INTRODUCERE ÎN ȘTIINȚA DATELOR

Suport de curs pentru studenți

Semestrul iarnă, 2021

Cuprins

3. Vizualizarea datelor

"The simple graph has brought more information to the data analyst's mind than any other device." — **John Tukey**

3.1. Introducere

Acest capitol vă va învăța să vizualizați datele folosind ggplot2. R are mai multe sisteme pentru realizarea vizualizărilor/ploturilor, însă ggplot2 reprezintă unul dintre pachetele cele mai elegante și versatile. ggplot2 implementează gramatica graficii, un sistem coerent pentru descrierea și construirea reprezentărilor grafice.

3.1.1. Cerințe

Această secțiune se concentrează pe ggplot2, unul dintre membrii de bază din tidyverse. Pentru a accesa seturile de date, paginile de ajutor și funcțiile pe care le vom folosi în acest capitol, încărcați versiunea tidyverse executând următorul cod:

```
library(tidyverse)
```

Vom utiliza pachetul ***tidyverse*** în aproape toate analizele a datelor. De asemenea sunt indicate conflictele cu denumiri a funcțiilor din alte pachete pe care le aveți instalate sau le veți instala.

Dacă rulați acest cod și primiți un mesaj de eroare *`there is no package called "tidyverse"`*, va trebui mai întâi să instalați pachetul apelând la funcțiile *`install.packages`* și *`library()`*.

```
install.packages("tidyverse")
library(tidyverse)
```

Trebuie să instalați un pachet o singură dată, însă trebuie să îl reîncărcați de fiecare dată când începeți o nouă sesiune.

Pentru a indica de unde provine o funcție (pachetul) sau un set de date, vom folosi formularea *`pachet::funcția()`*. De exemplu, *`ggplot2::ggplot()`* vă spune în mod explicit că folosim funcția *`ggplot()`* din pachetul *`ggplot2`*.

3.2. Primii pași

Să folosim o reprezentare grafică pentru a răspunde la întrebarea: dacă mașinile cu capacitatea motorului mai mare consumă mai mult combustibil ca mașinile cu motoare mici? Probabil că aveți deja un răspuns, însă de data asta vom oferi un răspuns precis. Cum arată relația dintre dimensiunea motorului și consumul de combustibil? Este o relație pozitivă? Negativă? Liniară? Ne-liniară?

3.2.1. Setul de date *mpg*

Puteți testa răspunsul setul de date *mpg*, pe care îl găsiți în pachetul ***ggplot2*** (*ggplot2::mpg*). Setul de date reprezintă un ***data frame***, o colecție rectanugulară de variabile (în coloane) și observații (în rânduri). *mpg* conține observații colectate de Agenția SUA pentru Protecția Mediului pentru 38 modele de mașini.

```
library(ggplot2)
mpg
```

```
## # A tibble: 234 x 11
##    manufacturer model    displ  year   cyl trans   drv     cty   hwy fl    class
##    <chr>        <chr>    <dbl> <int> <int> <chr>   <chr> <int> <int> <chr> <chr>
##  1 audi         a4         1.8  1999     4 auto(l~ f        18    29 p     comp~
##  2 audi         a4         1.8  1999     4 manual~ f        21    29 p     comp~
##  3 audi         a4         2    2008     4 manual~ f        20    31 p     comp~
##  4 audi         a4         2    2008     4 auto(a~ f        21    30 p     comp~
##  5 audi         a4         2.8  1999     6 auto(l~ f        16    26 p     comp~
##  6 audi         a4         2.8  1999     6 manual~ f        18    26 p     comp~
##  7 audi         a4         3.1  2008     6 auto(a~ f        18    27 p     comp~
##  8 audi         a4 quat~   1.8  1999     4 manual~ 4        18    26 p     comp~
##  9 audi         a4 quat~   1.8  1999     4 auto(l~ 4        16    25 p     comp~
## 10 audi         a4 quat~   2    2008     4 manual~ 4        20    28 p     comp~
## # ... with 224 more rows
```

Printre variabilele *mpg* se numără:

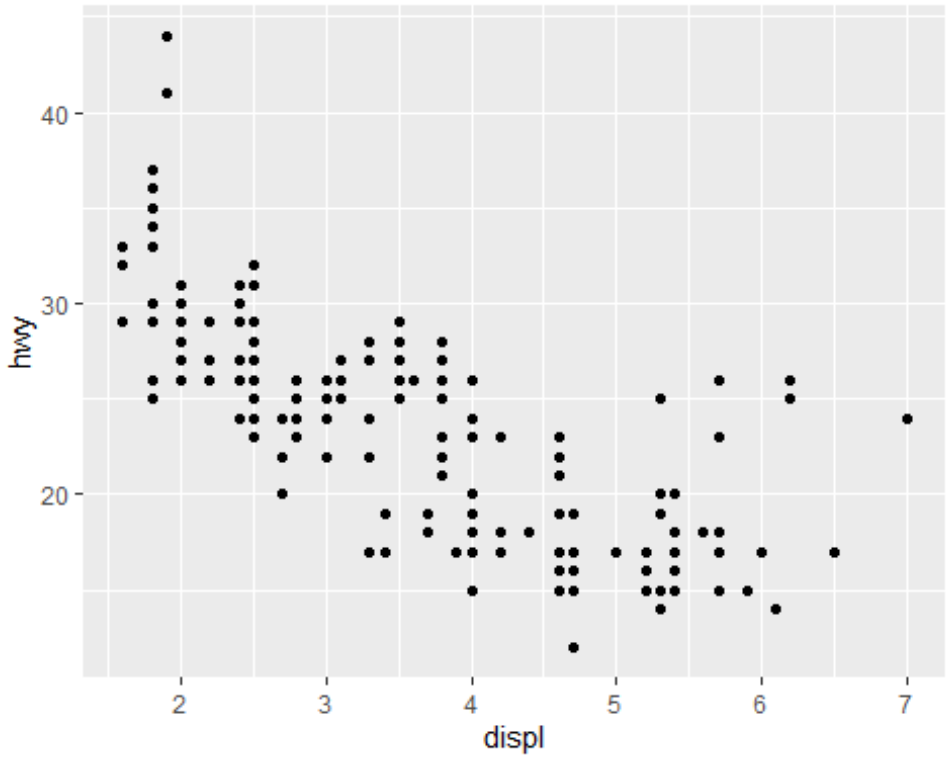
1. *displ*, dimensiunea motorului în litri;
2. *hwy*, randamentul consumului al unei mașini pe autrostradă, în mile pe galon (mpg). O mașină cu randamentul energetic mic consumă mai mult carburant față de o mașină cu randamentul energetic ridicat, odată ce parcurg aceeași distanță.

Pentru a afla mai multe despre setul de date mpg, puteți apela la comanda de ajutor, executând *?mpg*.

3.2.2. Crearea obiectelor *ggplot*

Pentru a reprezenta datele de interes din setul *mpg*, rulați următorul cod pentru a reprezenta variabilele *displ* pe axa *x* și *hwy* pe axa *y*:

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



Reprezentarea arată o relație negativă între dimensiunea motorului (`displ`) și randamentul de consum al carburanților (`hwy`). Cu alte cuvinte, autoturismele cu motoare mai mari consumă mai mult combustibil. Vă confirmă sau vă infirmă ipoteza despre consumul de combustibil și dimensiunea motorului?

Utilizând pachetul ggplot2, începem reprezentarea grafică prin rularea funcției `ggplot()`. `ggplot()` creează un sistem de coordonate la care puteți adăuga straturi. Primul argumet `ggplot()` este setul de date utilizat în grafic, deci `ggplot (data=mpg)` creează o reprezentare goală.

În continuare, completăm reprezentarea adăugând unul sau mai multe straturi la `ggplot()`. Funcția `geom_point()` adaugă un strat de puncte a graficului, ceea ce creează diagrama de dispersie din figură (scatterplot). ***ggplot2*** vine cu numeroase funcții geom care adaugă fiecare un tip diferit de staturi unei reprezentări la care se lucrează.

Fiecare funcție geom din ggplot2 ia un argumet pentru mapare/cartare. Aceasta definește modul în care variabilele din setul de date sunt cartate la proprietățile vizuale. Argumentul de cartare este întotdeauna asociat cu `aes()`, iar argumentele *x* și *y* ale lui `aes()` specifică ce variabile trebuie cartate pe axele x și y. ggplot2 caută pentru variabilele cartate în argumentele datelor, în aces caz, `mpg`.

3.2.3. Șablonul grafic

Să transformăm acest cod într-un șablon reutilizabil pentru realizarea reprezentărilor cu ggplot2. Pentru a crea un grafic, înlocuiți secțiunile parantezate din codul de mai jos cu un set de date, o funcție geom sau o colecție de cartări.

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

În restul acestei secțiuni vă vom arăta cum să completați și să extindeți acest șablon pentru a crea diferite tipuri de grafice. Vom începe cu componenta *<MAPPINGS>*.

3.2.4. Exerciții

1. Rulați codul *ggplot(data=mpg)*. Ce obțineți?

```
ggplot(data=mpg)
```


Acest cod creează un grafic gol. Funcția *ggplot()* creează fundalul graficului, dar din moment ce nu au fost specificate straturile prin funcția *geom*, nu se afișează nimic.

2. Câte rânduri și coloane are setrul de date *mpg*?

```
nrow(mpg)

## [1] 234

ncol(mpg)

## [1] 11
```

```
glimpse(mpg)
```

```
## Rows: 234
## Columns: 11
## $ manufacturer <chr> "audi", "audi", "audi", "audi", "audi", "audi", "audi"...
## $ model        <chr> "a4", "a4", "a4", "a4", "a4", "a4", "a4", "a4 quattro"...
## $ displ        <dbl> 1.8, 1.8, 2.0, 2.0, 2.8, 2.8, 3.1, 1.8, 1.8, 2.0, 2.0,...
## $ year         <int> 1999, 1999, 2008, 2008, 1999, 1999, 2008, 1999, 1999, ...
## $ cyl          <int> 4, 4, 4, 4, 6, 6, 6, 4, 4, 4, 4, 6, 6, 6, 6, 6, 6, 8, ...
## $ trans        <chr> "auto(l5)", "manual(m5)", "manual(m6)", "auto(av)", "a...
## $ drv          <chr> "f", "f", "f", "f", "f", "f", "f", "4", "4", "4", "4",...
## $ cty          <int> 18, 21, 20, 21, 16, 18, 18, 18, 16, 20, 19, 15, 17, 17...
## $ hwy          <int> 29, 29, 31, 30, 26, 26, 27, 26, 25, 28, 27, 25, 25, 25...
## $ fl           <chr> "p", "p", "p", "p", "p", "p", "p", "p", "p", "p", "p",...
## $ class        <chr> "compact", "compact", "compact", "compact", "compact",...
```

```
dim(mpg)
```

```
## [1] 234  11
```

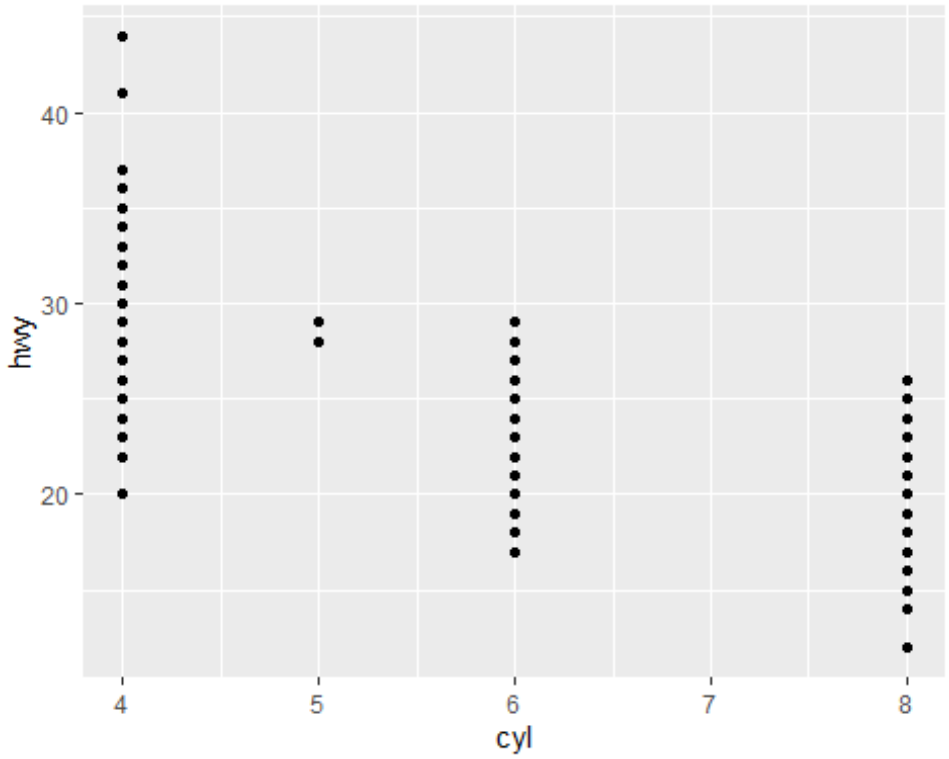
3. Ce descrie variabila *drv*? Apelați la ajutorul funției *?mpg*.

Variabila *drv* este o variabilă categorică care clasifică mașinile după tracțiune din față, spate și cea integrală.

Valoarea	Descrierea
“f”	tracțiunea din față
“r”	tracțiunea din spate
“4”	tracțiune integrală

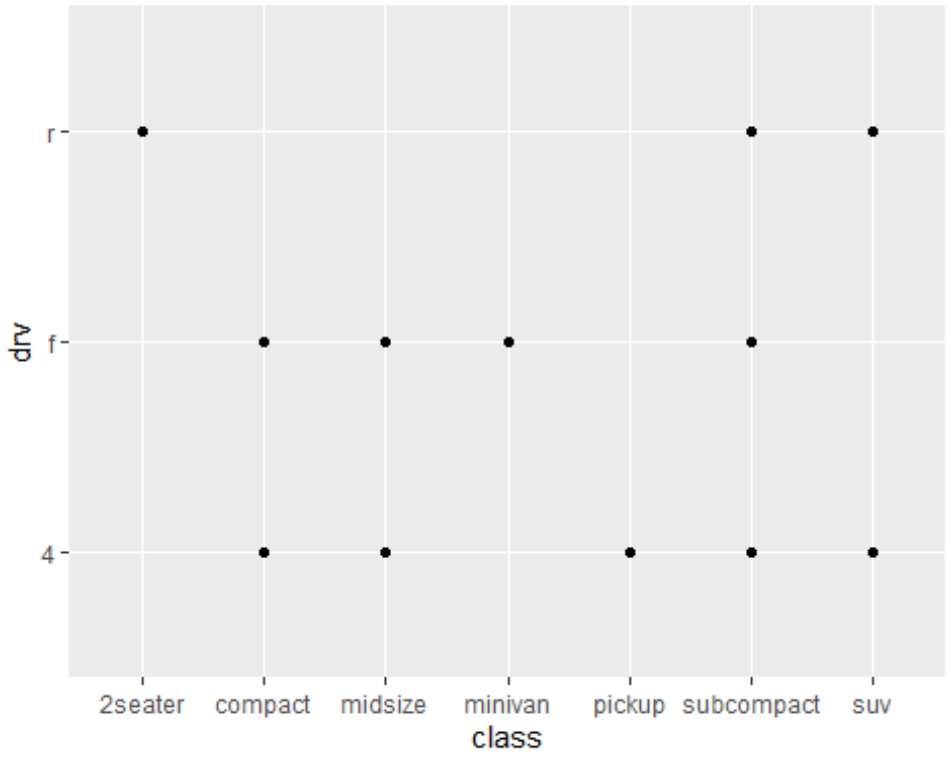
4. Reprezentați un scatterplot pentru *hwy* vs *cyl*.

```
ggplot(mpg, aes(x=cyl, y=hwy)) +
  geom_point()
```



5. Ce se va întâmpla dacă efectuăm un scatterplot pentru `class` vs `drive`? De ce reprezentarea nu este utilă?

```
ggplot(mpg, aes(x=class, y=drv)) +
  geom_point()
```



Un scatterplot nu reprezintă o expresie utilă pentru aceste variabile, deoarece atât *drv*, cât și *class* sunt variabile categorice. Deoarece varibilele categorice iau de obicei nu număr mic de valori, există un număr limitat de combinații unice de valori *(x,y)* care pot fi afișate. În datele noastre, *drv* ia trei valori, iar *class* ia șapte valori, ceea ce înseamnă că există doar 21 de valori care ar putea fi trasate într-o diagramă de dispersie pentru *drv* vs *class*. În datele noastre obținem 12 valori pentru *(drv, class)*.

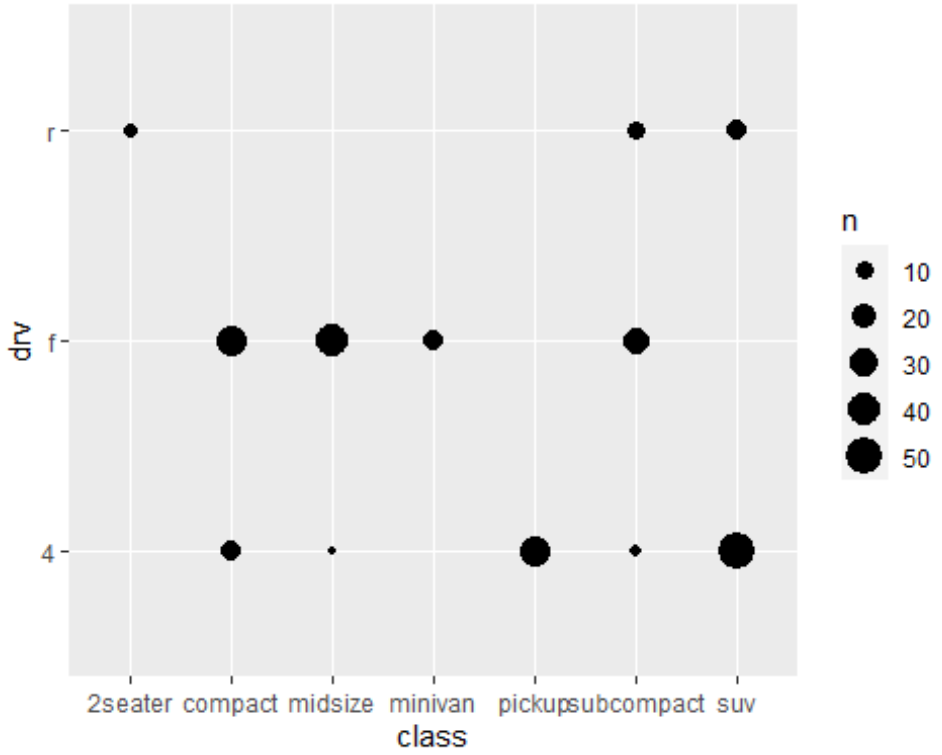
```
count(mpg, drv, class)
```

```
## # A tibble: 12 x 3
##    drv   class          n
##    <chr> <chr>      <int>
##  1 4     compact       12
##  2 4     midsize        3
##  3 4     pickup        33
##  4 4     subcompact     4
##  5 4     suv           51
##  6 f     compact       35
##  7 f     midsize       38
##  8 f     minivan       11
##  9 f     subcompact    22
## 10 r     2seater        5
## 11 r     subcompact     9
## 12 r     suv           11
```

Un scatterplot nu arată câte observații există pentru fiecare valoare *(x,y)*. Astfel, reprezentările pentru dispersie funcționează cel mai bine pentru valori constinue pentru *x* și *y*, precum și când toate valorile *(x,y)* sunt unice.

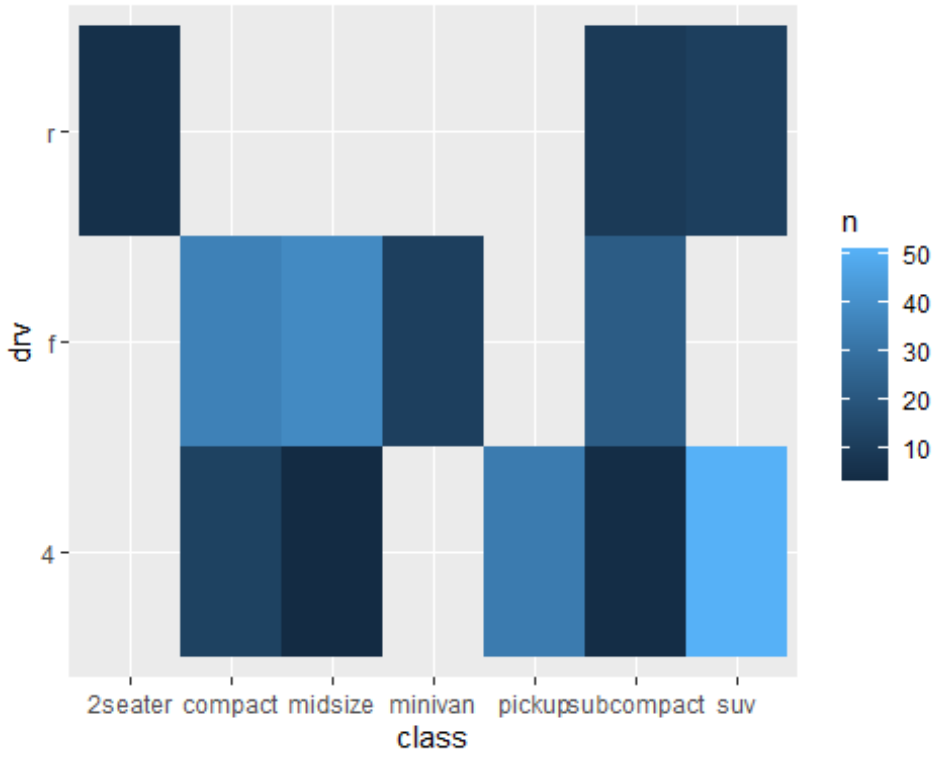
Pentru a fi mai informativ, în cazul a două variabile categorice, introducem funcția *geom_tile()*, care este similară scatterplotului, însă utilizează mărimimea punctului pentru a arăta numărul de observații pentru un punce *(x,y)*.

```
ggplot(mpg, aes(x=class, y=drv)) +
  geom_count()
```



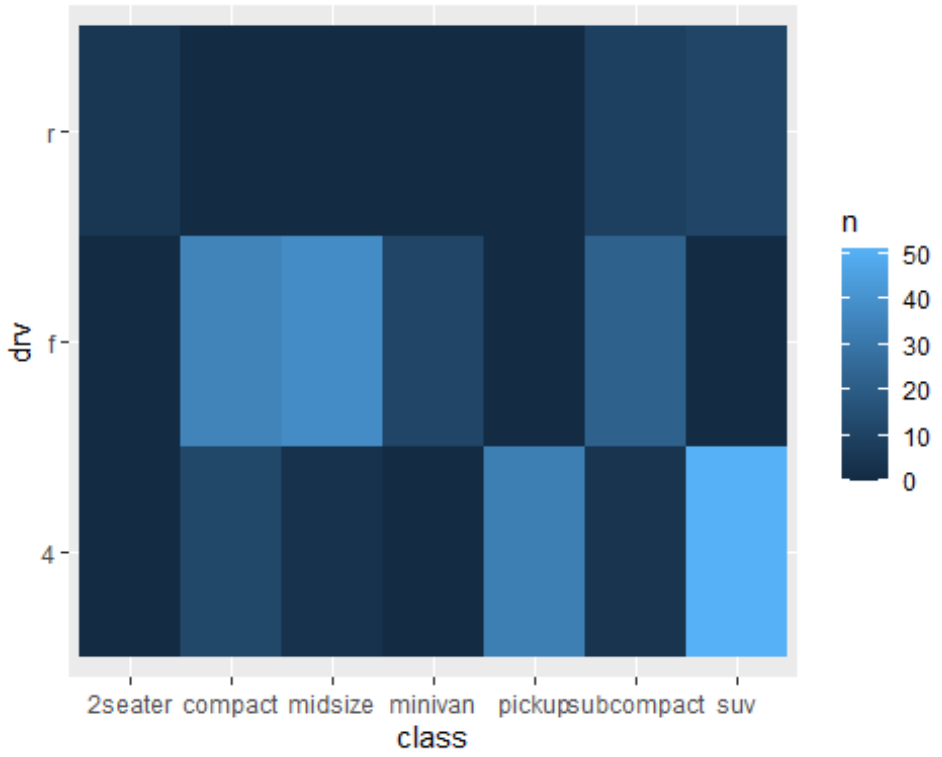
O altă funcție este *geom_tile()*, care utilizează o scară coloră pentru a arăta numărul de observații pentru fiecare valoare *(x,y)*.

```
mpg %>%
  count(class, drv)%>%
  ggplot(aes(x=class, y=drv))+
  geom_tile(mapping=aes(fill=n))
```



În graficul anterior observați lipsa a numeroase date. Lipsa culorii reprezintă combinații neobservate de valori pentru valorile *class* și *drv*. Valorile lipsă nu sunt necunoscute, doar reprezintă valori pentru *(class, drv)*, unde *n=0*. Funcția *complete()* din pachetul *tidyr* adaugă rânduri noi în setul de date pentru combinațiile lipsă de coloane. Următorul cod adaugă rânduri pentru combinațiile lipsă în *class* și *drv* și folosește argumentrul de umplere pentru a seta *n=0* pentru aceste rânduri noi.

```
mpg %>%
  count(class, drv)%>%
  complete(class, drv, fill=list(n=0)) %>%
  ggplot(aes(x=class, y=drv))+
  geom_tile(mapping=aes(fill=n))
```



3.3. Cartarea estetică

"The greatest value of a picture is when it forces us to notice what we never expected to see."
— John Tukey

În graficul de mai jos, un grup de puncte (evidențiate cu roș) pare să se încadreze în afara tendinței liniare. Aceste mașini au un parcurs mai mare decât cel așteptat. Cum putem explica aceste lucruri? Să presupunem că mașinile usnt hibride. O modalitate de a testa această ipăteză este să analizăm valoarea clasei pentru fiecare autovehicul. Variabila class a setului de date mpg clasifică autovehiculele în grupuri de tip compacte, mijlocii și SUV.

Dacă punctele periferice sunt hibride, acestea trebuie să fie clasificate în autovehicule de tip compacte sau, poate, subcompacte (aceste date au fost colectate înainte ca camioanele și SUV-urile hibride să devină populare).

Astfel, putem adăuga o a treia variabilă, cum ar fi clasa (class), într-un scatterplot bidimensional, prin cartarea estetică. Prelucrările estetice dunt proprietăți vizuale a obiectelor din reprezentările sau ploturile noastre. Esteticile includ lucruri precum dimensiuni, forme sau culori ale entităților reprezentate (punctelor în cazul nostru):

Putem afișa un punct (ca cel de mai jos) în moduri deiferite, schimbând valorile proprietăților sale estetice. Întrucât folosim deja cuvântul „valoare” pentru a descrie datele, să mai utilizăm și cuvântul „nivel” pentru a descrie proprietățile sale estetice. Aici schimbăm nivelurile dimenilor, formelor sau culorile punctelor pentru a face punctele mai mici, mari, de le trasforma în triunghiuri patrate sau a le schimba cu locurile.

Putem transmite informații despre datele nostre prin cartări estetice din grafic la variabilele de setul de date. De exemplu, putem carta culorile punctelor la variabiele clasei pentru a dezvălui clasa fiecărui autovehicul.

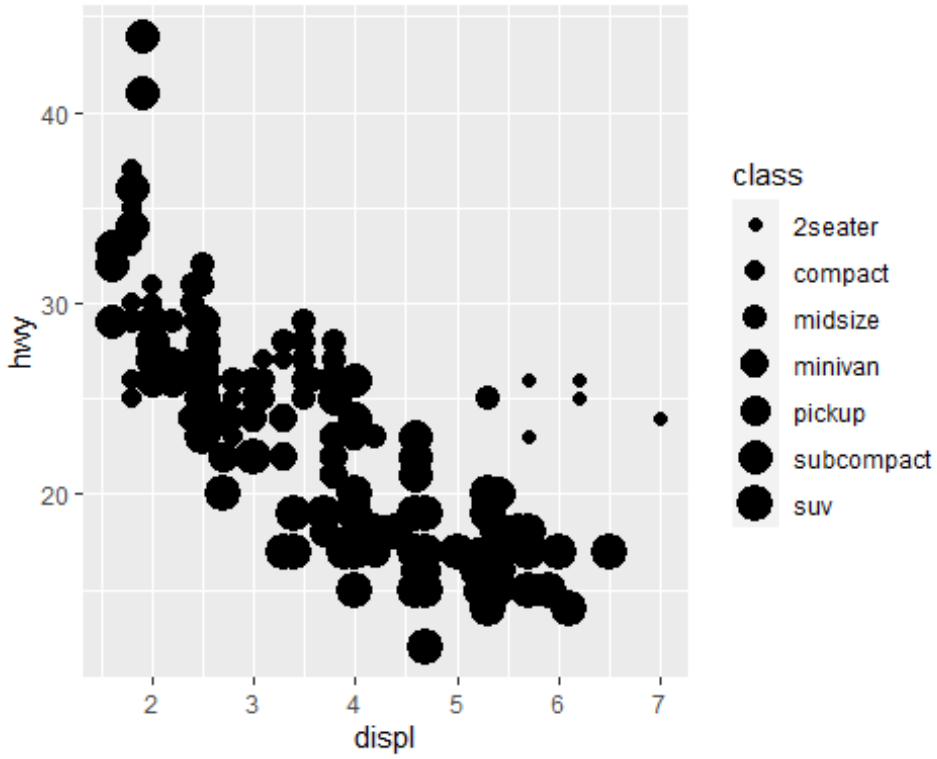
```
ggplot(data=mpg) +
  geom_point(mapping = aes(x=displ, y=hwy, color=class))
```

Pentru a asocia o estetică la o variabilă, asociți numele esteticului cu numele variabilei din interiorul funcției `aes()`. ggplot2 va atribui automat un nivel unic al esteticului (aici o culoare unică) fiecărei valori unice a variabilei, un proces cunoscut sub numele de scalare. ggplot2 va adăuga, de asemenea, o legendă care explică ce nivele corespund valorilor.

Culorile dezvăluie că multe dintre punctele neobișnuite (cele roșii) sunt autovehicule cu două locuri. Aceste mașini nu par hibride, și sunt, de fapt, mașini sport! Mașinile sport au motoare mari, cum ar fi la SUV-uri și camionete, însă corpurile mici, cum ar fi la mașinile mijlocii și compacte, ceea ce mărește semnificativ parcursul acestora. În retrospectivă, este puțin probabil ca aceste autovehicule să fie hibride, deoarece au motoare mari.

În exemplul de mai sus, am asociat clasa (`class`) la estetica culorii, însă am fi putut asocia clasa la estetica mărimii în același mod. În acest caz, dimensiunea exactă a fiecărui punct ar dezvălui apartenența la clasă. Aici vom primi un avertisment, deoarece cartarea unei variabile neordonate (`class`) la o estetică ordonată (`size`) nu este o idee bună.

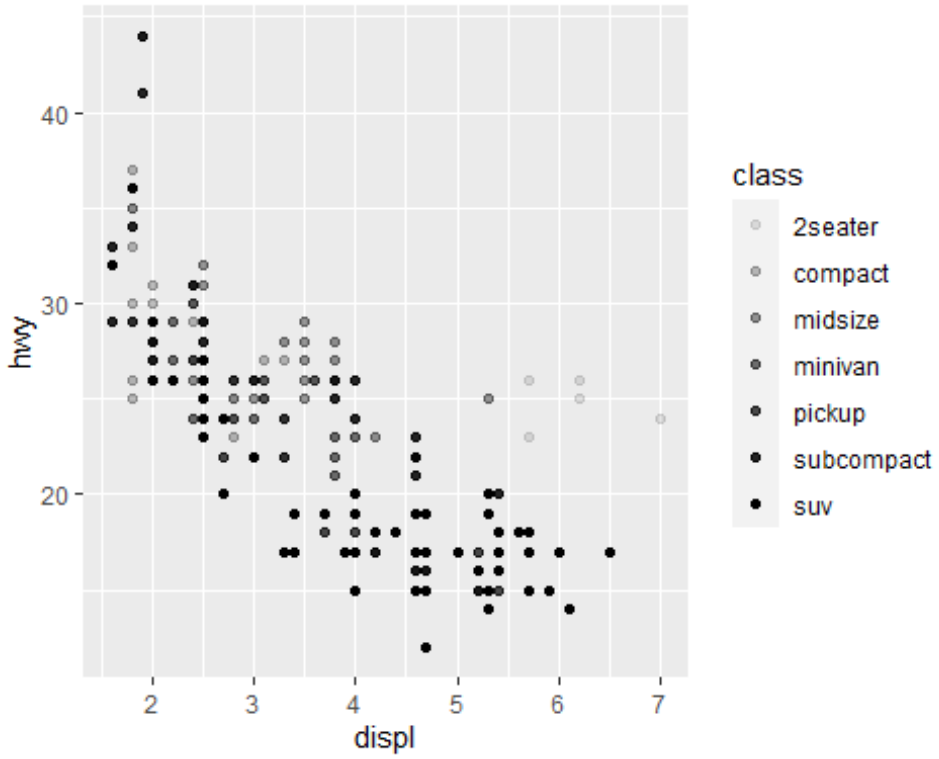
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, size = class))

## Warning: Using size for a discrete variable is not advised.
```



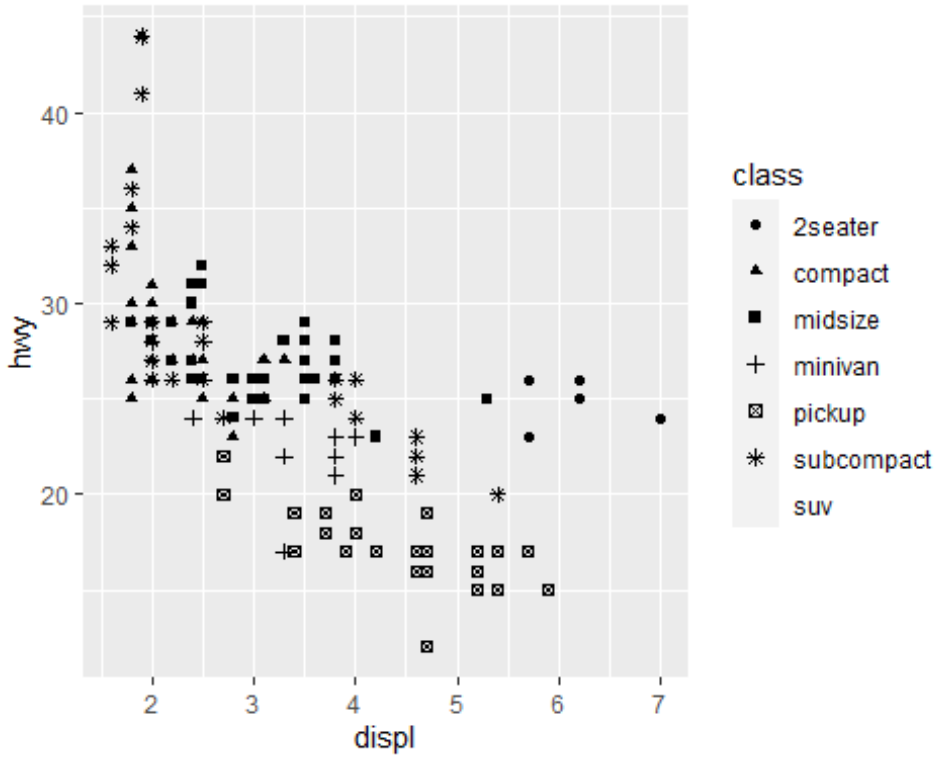

```
#> Warning: Using size for a discrete variable is not advised.
```

Sau am fi putut carta clasa la estetica alfa, care controlează transparența punctelor, la la esteticul formei, care controlează forma punctelor.

```
# Stânga
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, alpha = class))

## Warning: Using alpha for a discrete variable is not advised.
```




```
# Dreapta
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, shape = class))

## Warning: The shape palette can deal with a maximum of 6 discrete values be
cause
## more than 6 becomes difficult to discriminate; you have 7. Consider
## specifying shapes manually if you must have them.

## Warning: Removed 62 rows containing missing values (geom_point).
```

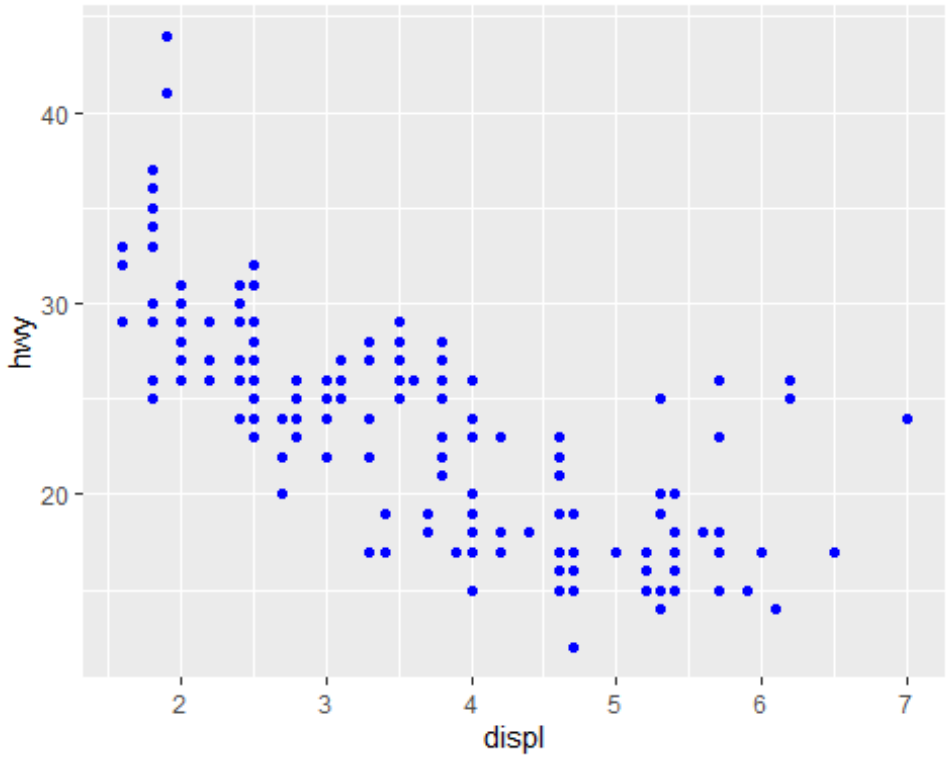


Ce s-a întâmplat cu SUV-urile? ggplot2 folosește doar șase forme o dată. În mod implicit, grupurile suplimentare vor rămâne ne-cartate atunci când utilizăm estetica formei.

Pentru fiecare estetică utilixăm *aes()* pentru a asocia numele esteticului cu o variabilă de afișat. Funcția *aes()* reunește fiecare dintre cartările estetice utilizate de un strat și le trece la argumentrul de cartare al stratului. Sintaxa evidențiază o perspectivă utilă despre x și y: locațiile x și y ale unui punct sunt și estetice, proprietăți vizuale pe care le putem asocia la variabile pentru a afișa informații despre date.

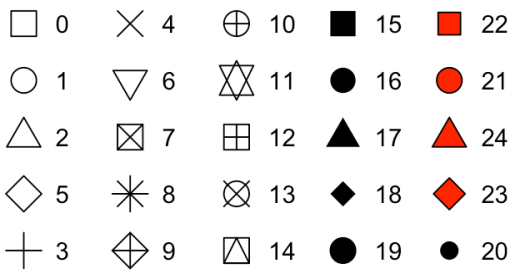
Odată ce am cartat un estetic, ggplot2 se ocupă de restul. Selectează o scară rezonabilă de utilizare cu esteticul utilizat și construiește o legendă care explică cartarea între nivele și valori. Pentru x și y, ggplot2 nu creează o legendă, dar creează o linie de axă cu etichetă. Linia axei acționează ca o legendă: explică cartarea între locații și valori. De asemenea, putem seta manual proprietățile estetice pentru geom. De exemplu, putem face toate punctele din reprezentare să fie albastre:

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



Aici, culoarea nu transmite informații despre o variabilă, ci doar modifică aspectul reprezentării. Pentru a seta manual o estetică, setați estetica după nume ca argument al funcției geom, adică iese în afara la `aes()`. Va trebui să alegem un nivel care să aibă sens pentru această estetică:

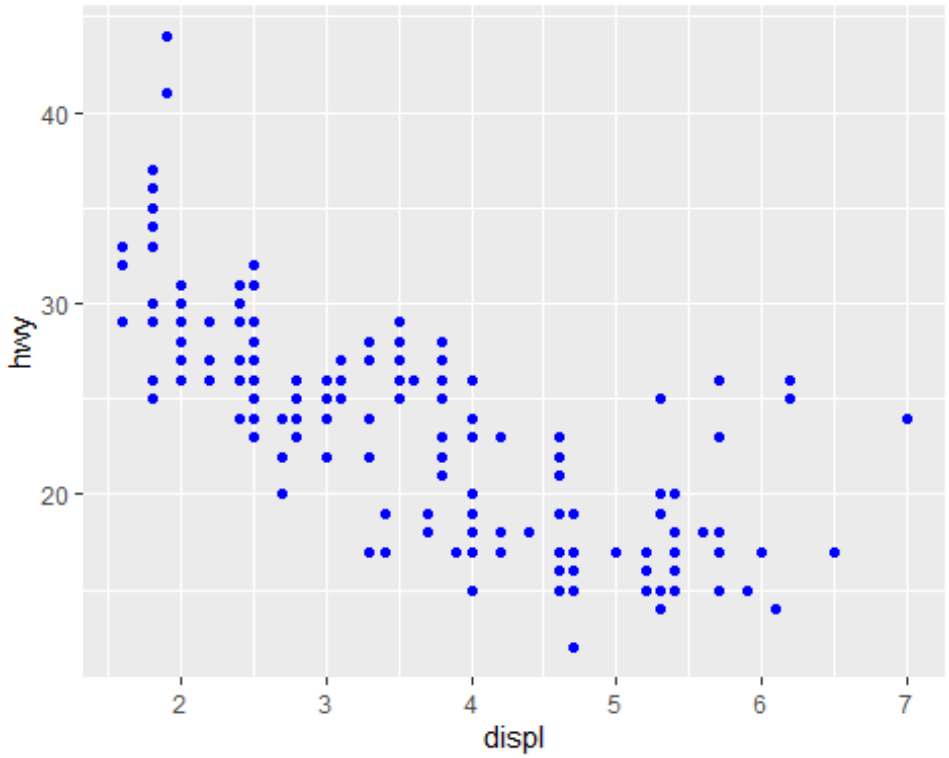
- Numele culorii ca șir de caractere.
- Dimensiunea unui punct în mm.
- Forma punctelor codate în cifre:



3.3.1. Exerciții

1. Ce a mers greșit în acest cod? De ce punctele nu sunt albastre?

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



2. Care variabile din `mpg` sunt categorice? Care variabile sunt continue? Cum vedeți aceste informații când rulați `mpg`?

3. Cartați o variabilă continuă la culoare, formă și dimensiune. Care esete diferența de comportament dintre aceste estetice pentru variabile categorice și cele continue?

4. Ce se întâmplă dacă folosiți aceeași variabilă la mai multe estetice?

5. Ce dacă esteticul `stroke`? Cu ce forme funcționează (utilizați `?geom_point`).

6. Ce se întâmplă dacă asociați o estetică cu altceva decât numele unei variabile, cum ar fi `aes(color = displ<5)`? Rețineți, va trebui de asemenea, se specificat `x` și `y`.

3.4. Probleme comune

Pe măsură ce începeți să rulați R-ul, este posibil să întâmpinați probleme. Nu vă faceți griji - se întâmplă tuturor. Personal, scriu coduri R de ani de zile și în fiecare zi scriu coduri care nu funcționaează.

Începeți prin a compara cu atenție codurile pe cale le executați cu cele din suportul de curs. R este extrem de pretențios, iar un caracter deplasat poate dă vă dea bătăi de cap. Uneori veți rula codul și nu se va întîmpla nimic. Verificați partea stângă a consolei: dacă este un +, înseamnă R nu crede că ați scris o expresie completă și așteaptă să o teminați. În acest caz, este bine de obicei să porniți de la zero apăsând pe ESCAPE pentru a întrerupe procesarea comenzii curente.

O problemă comună atunci când creați reprezentări cu ggplot2 este să puneți + în locul greșit: trebuie să vină la sfârșitul liniei, nu la început. Cu alte cuvinte, asigurați-vă că nu ați scris din greșeală linii de cod de genul acesta:

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

Dacă sunteți în continure deurat(ă), încercați ajutorul. Puteți obține ajutor privind orice funcție în R executând *?Function_name* în consolă sau selectând numele funcției și apăsând pe F1 în RStudio. Nu vă faceți griji dacă ajutorul nu pare atât de util - în schimb, treceți la exemple și căutați coduri care să se potrivească cu ceea ce încercați să faceți.

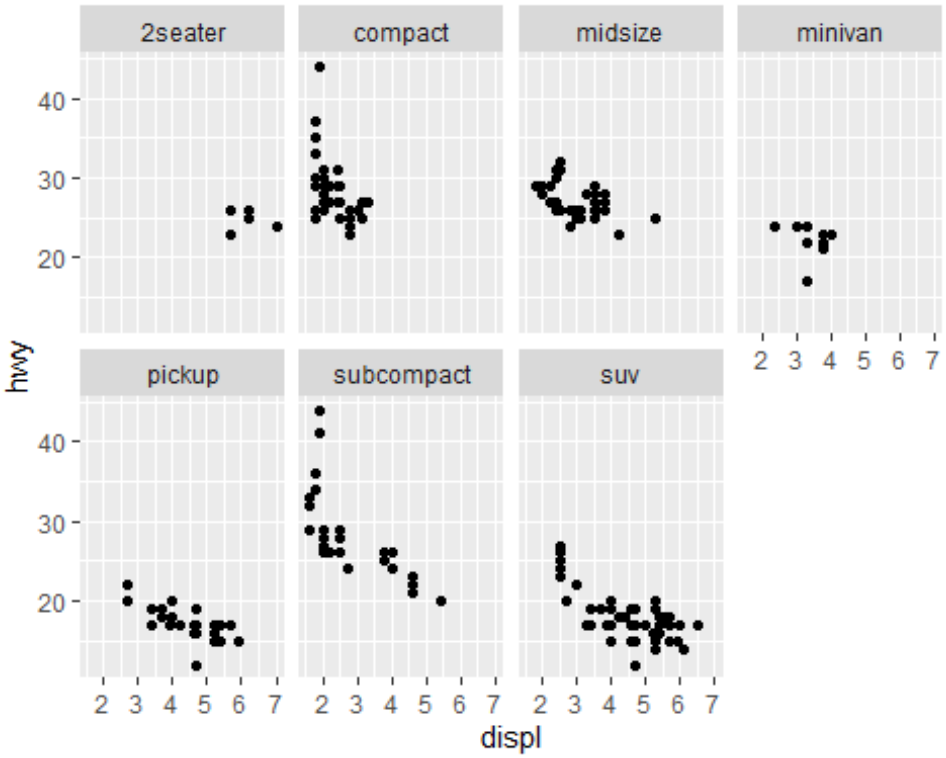
Dacă acest lucru nu ajută, citiți cu atenție mesajul de eroare. Uneori, răspunsul este masact anume aici! Însă când sunteți nou în R, răspunsul este în mesajul de eroare, și nu știți încă cum să-l înțelegeți, utilizați un alt instrument excelent - Google! Încercați să căutați mesajul de eroare, deoarece este foarte posibil că altcineva a avut aceeași problemă și a primit ajutor online.

3.5. Fațete

O modalita de a adăuga varibile suplimentare este utilizarea esteticilor. O altă modalitate, deosebit de utilă pentru variabilele categorice este împărțirea graficului în fațete, sub-ploturi care afișează fiecare câte un subset de date.

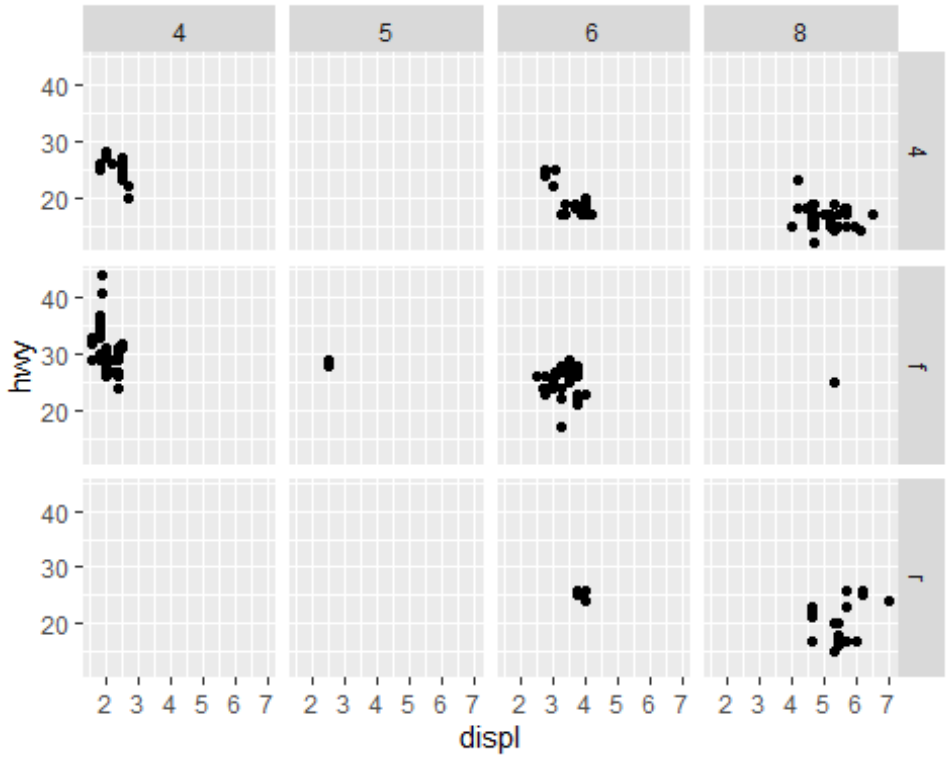
Pentru a fațeta reprezentarea pentru o singură variabilă, utilizați *facet_wrap()*. Primul argument din *facet_wrap()* trebuie să fie o formulă, pe care o creați cu ~ urmată de numele variabilei (aici „formula” este numele unei structuri de date în R, nu un sinonim pentru „ecuație”). Variabila pe care o treceți către *facet_wrap()* trebuie să fie discretă.

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



Pentru a fațeta graficul pentru combinația a două variabile, adăugați `facet_grid()` la apelul pentru vizualizare. Primul argument al `facet_grid()` este, de asemenea, o formulă. De data aceasta, formula ar trebui să conțină două nume de variabile separate printr-un simbol `~`.

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

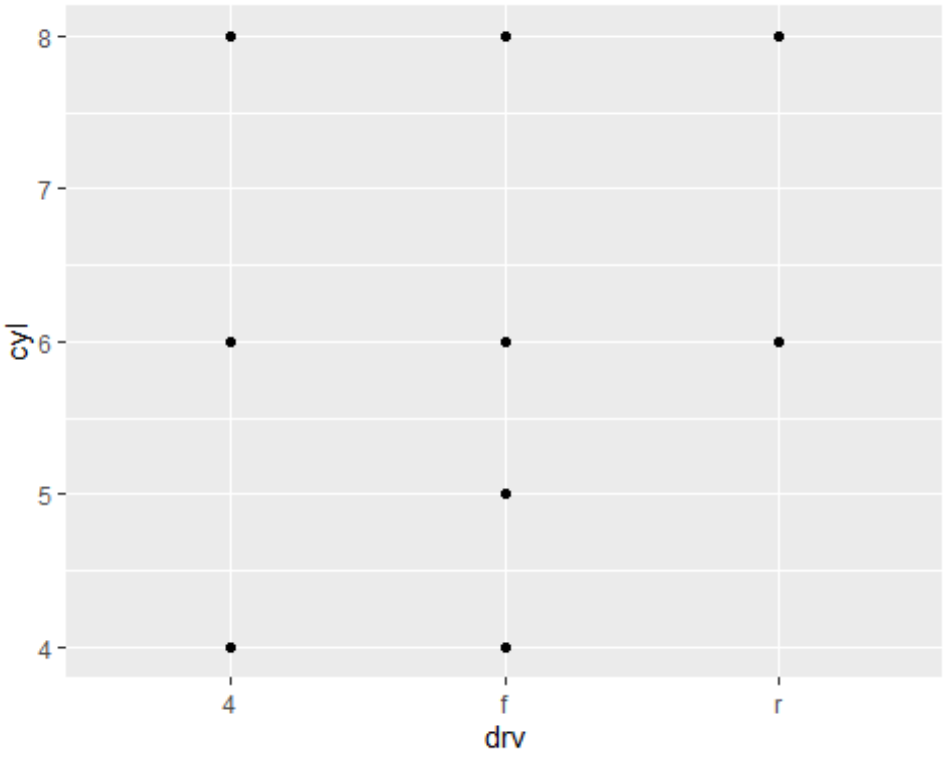


3.5.1. Exerciții

1. Ce sa întâmplă dacă fațetați asupra variabilelor continue?

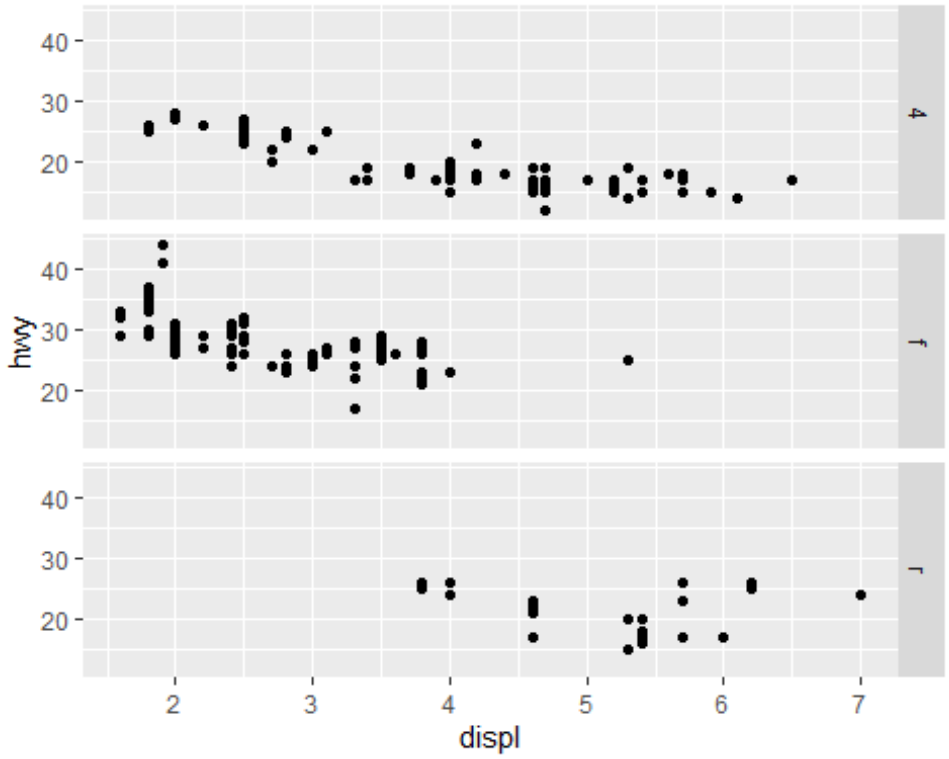
2. Ce înseamnă celulele goale din reprezentarea cu *facet_grid(drv~cyl)*? Cum le explicați în contextul reprezentării?

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

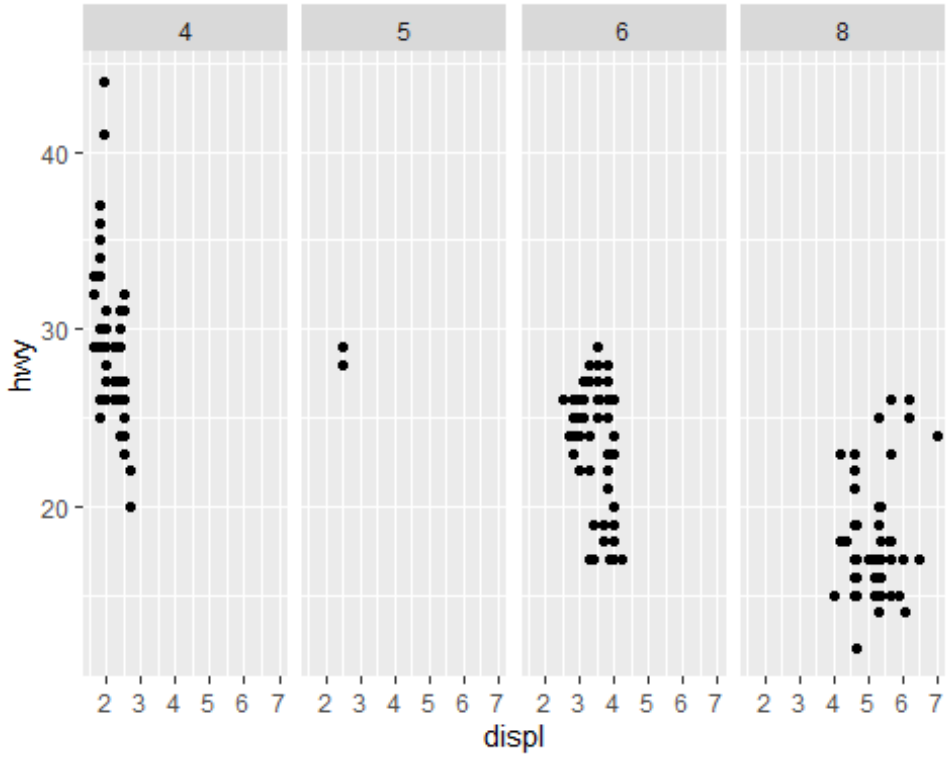


3. Ce va reprezenta următorul cod? Care este rolul .-ului?

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

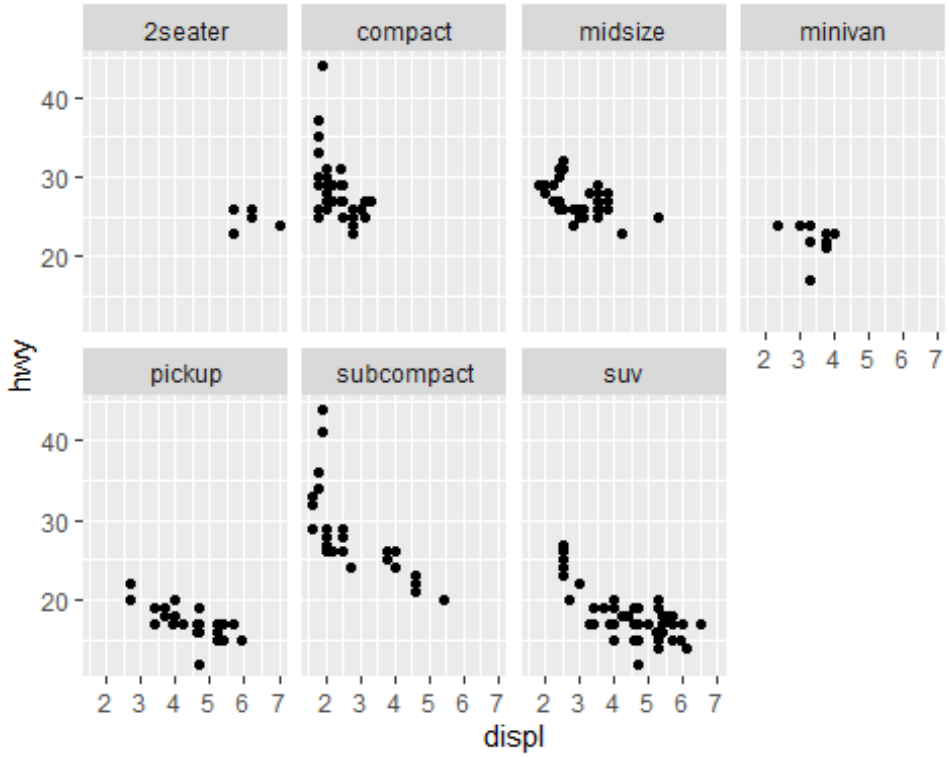



```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  facet_grid(. ~ cyl)
```



4. Efectuați următoarea fațetare:

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



Care sunt avantajele utilizării fațetării față de estetica coloristică? Care sunt dezavantajele? Cum s-ar putea schimba balanța dacă ați avea un set de date mai mare?

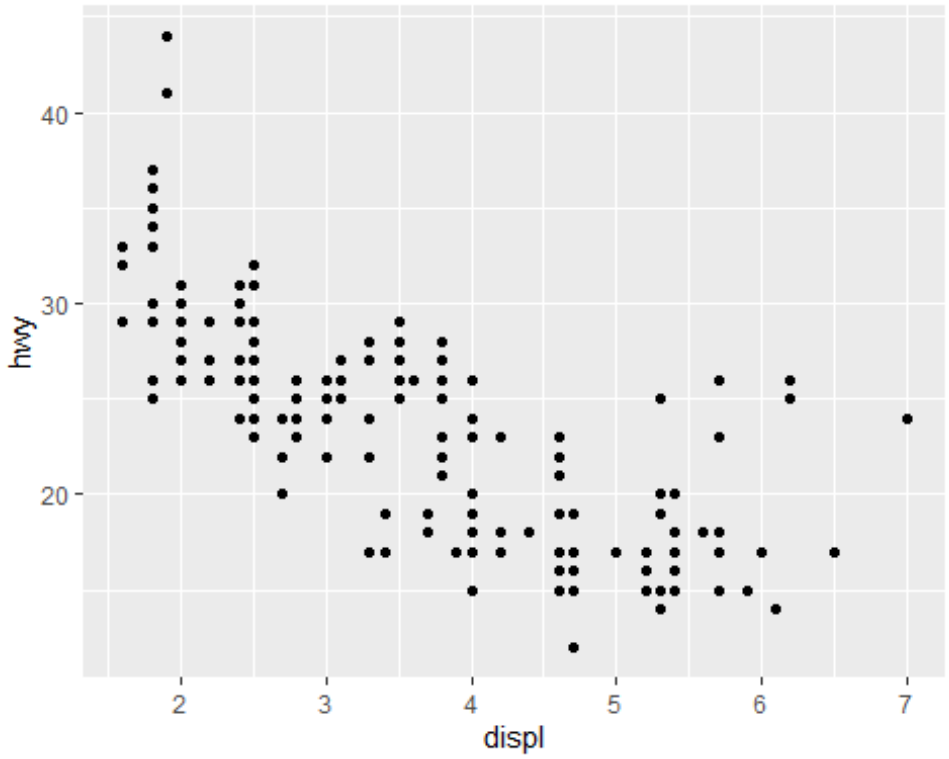
5. Executați `?face_wrap`. Ce face `nrow`? Ce face `ncol`? Ce alte opțiuni controlează aspectul panourilor individuale? De ce `facet_grid()` nu are argumentele `nrow` și `ncol`?

6. Când utilizăm `facet_grid()` ar trebui să punem variabila cu mai multe nivele unice în coloane. De ce?

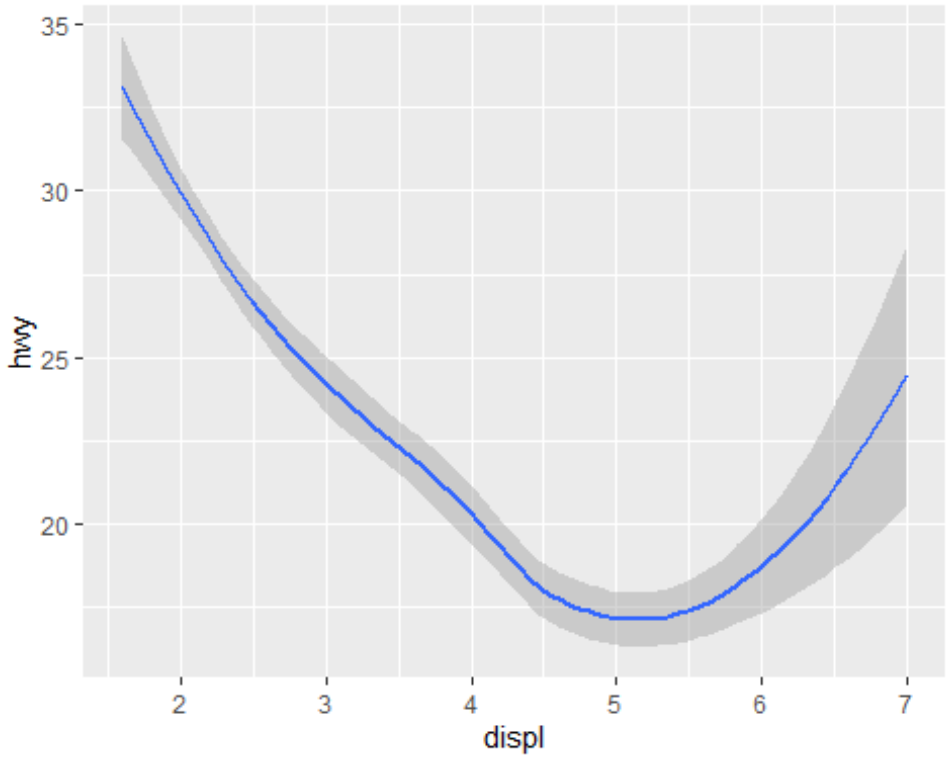
3.6. Obiecte geometrice

Prin ce seamănă următarele două ploturi?

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




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

## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



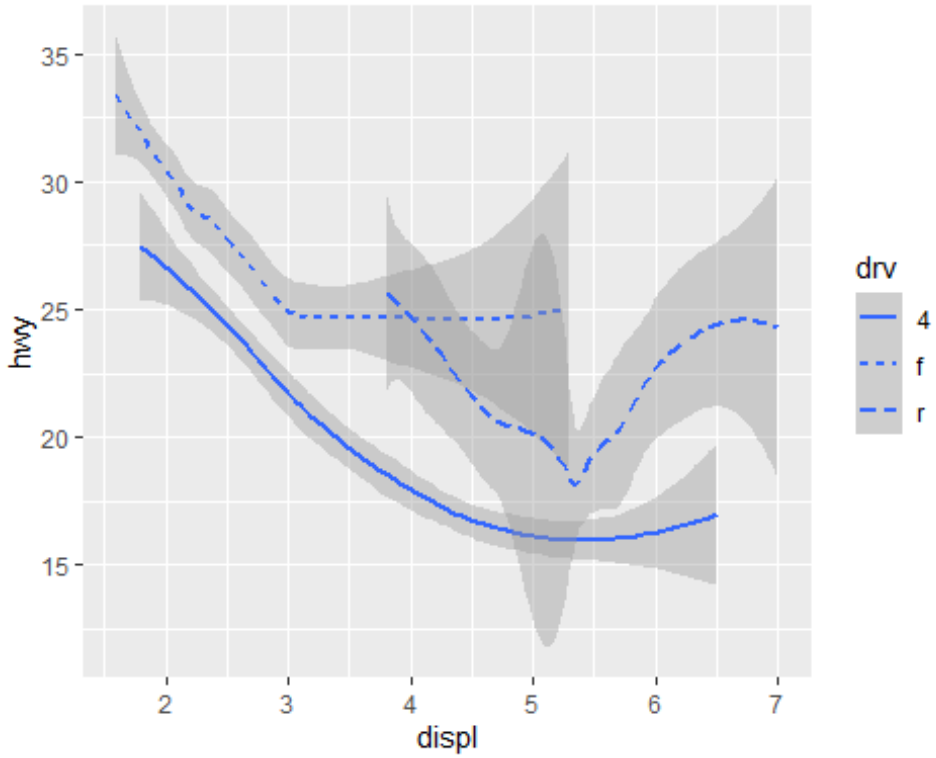
Ambele ploturi conțin aceeași variabilă x și aceași variabilă y, ambele descriind aceleași date. Însă ploturile nu sunt identice. Fiecare din ele folosește un obiect vizual diferit pentru a reprezenta datele. În sintaxa ggplot2 spunem că utilizează geomuri diferite.

Un geom este un obiect geometric pe care un grafic îl folosește pentru a reprezenta datele. Oamenii descriu adesea ploturile după tipul de geom utilizat. De exemplu, bar-ploturile utilizează geomuri bare, ploturile cu linii utilizează geomuri linii, boxploturile utilizează geomuri boxplot și așa mai departe. Scaterplotul rupe tendința, acesta folosește drept geom punctul. Așa cum e reprezentat mai sus putem utiliza geoame diferite pentru a reprezenta aceleași date. Graficul din stânga folosește punctul geom, iar graficul din dreapta folosește geomul neted, o linie netedă adaptată datelor.

Pentru a modifica gemul din grafic, se modifică funcția geom pe care o adăugăm la *`ggplot()`*.

Fiecare funcție geom din ggplot2 ia un argument de cartare. Cu toate acestea, nu orice estetică funcionează cu fiecare geom. Ați putea seta forma unui punct, însă nu ați putea seta forma unei linii. Pe de altă parte, puteți seta linetype pentru linie. *`geom_smooth()`* va trasa o linie diferită, cu un linetype diferit pentru fiecare valoare unică a variabilelor pe care le cartăm la *`linetype`*.

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

## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



Aici `geom_smooth()` separă mașinile în trei linii în funcție de valoarea lor `drv`, care descrie tipul de tracțiune a mașinii. O linie descrie toate punctele cu o valoare de `4`, o linie descrie toate punctele cu valoare `f` și o linie descrie toate punctele cu o valoare `r`. Aici, `4` reprezintă tracțiunea integrală, `f` pentru tracțiunea față și `r` pentru tracțiunea spate.

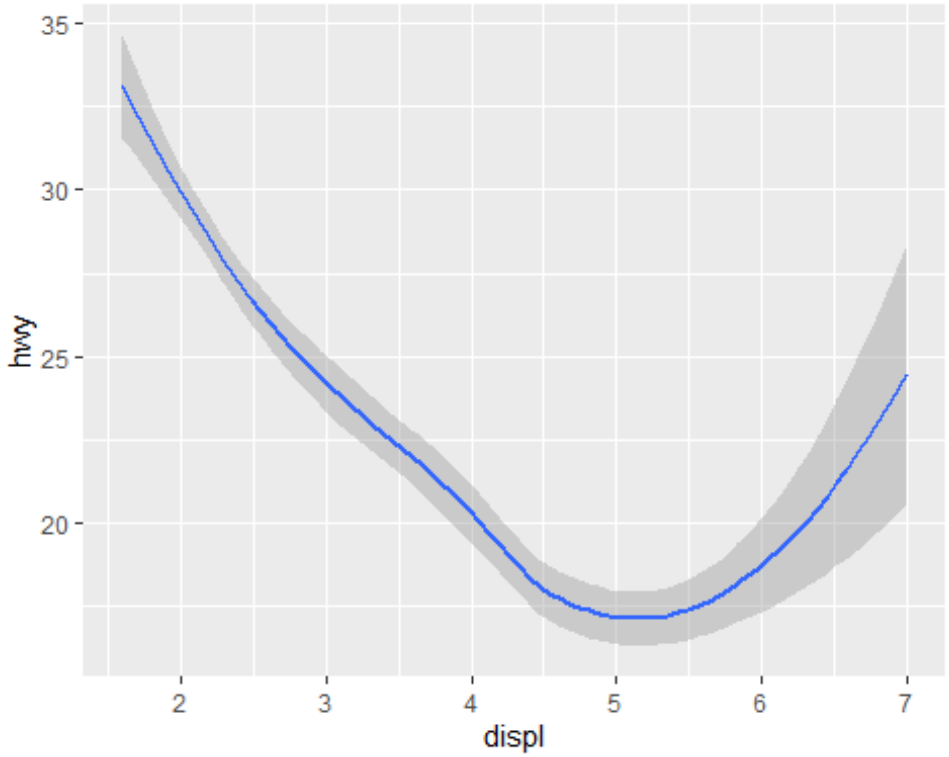
Dacă acestea arată cam bizar, putem să clarificăm lucrurile prin suprapunerea liniilor deasupra datelor brute și apoi să le colorăm în funcție de drv. Observăm că această reprezentare conține două geoame în același grafic! Foarte curând vă voi arăta cum să plasați mai multe geoame în același plot.

ggplot2 oferă peste 40 de geoame, iar pachetele de extensie și mai multe (vedeți *https://exts.ggplot2.tidyverse.org/gallery/*). Cel mai bun mod de a obține o imagine de ansamblu pentru ggplot2, găsiți ggplot2 pe *http://rstudio.com/cheatsheets*. Pentru a afla mai multe despre un singur geom, utilizați ?geom_smooth.

Numeroase geomuri, cum ar fi `geom_smooth()`, folosesc un singur obiect geometric pentru a afișa mai multe rânduri de date. Pentru aceste geomuri putem să setăm estetica grupului la o variabilă categorică pentru a reprezenta mai multe obiecte. ggplot2 va reprezenta un obiect separat pentru fiecare valoare unică a variabilei de grupare. În practică, ggplot2 va grupa automat datele pentru aceste geoame ori de câte ori asociați o estetică la o variabilă discretă (ca în exemplul `linetype`). Este convenabil să ne bazăm pe această caracteristică, deoarece estetica grupului, prin ea însăți nu adaugă o legendă sau trăsături distinctive geomurilor.

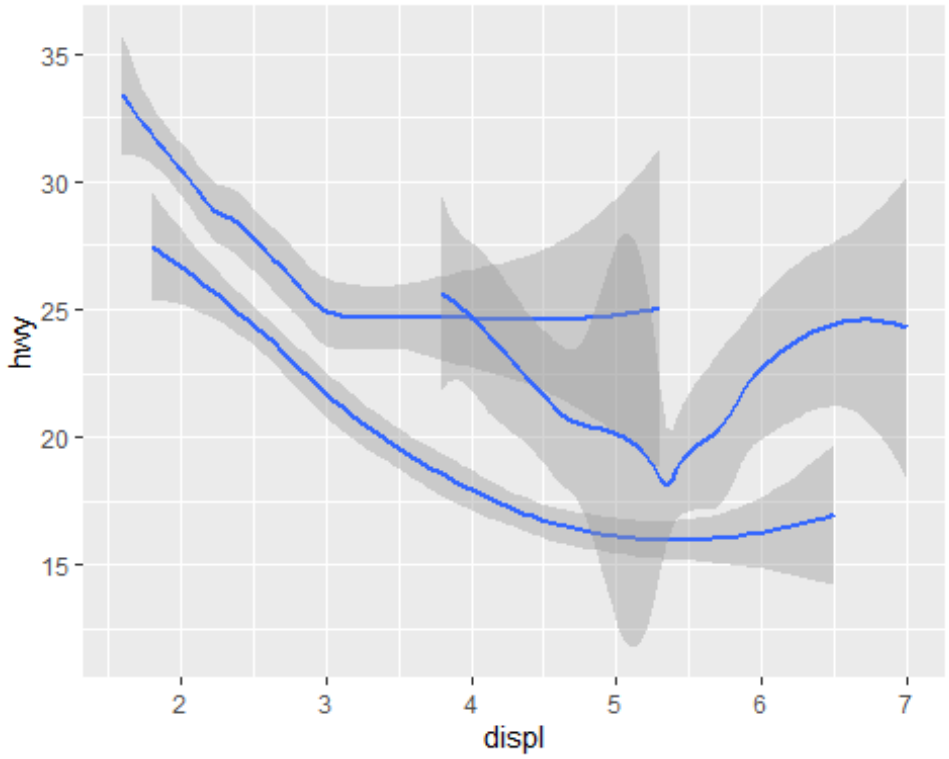
```
ggplot(data = mpg) +
  geom_smooth(mapping = aes(x = displ, y = hwy))
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

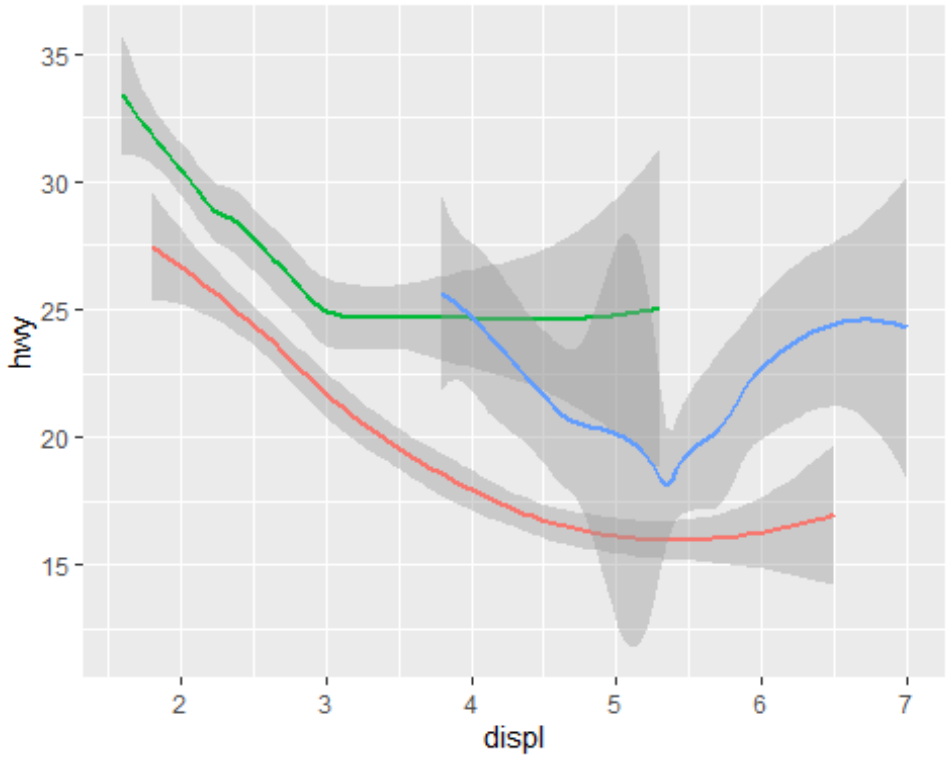



```
ggplot(data = mpg) +
  geom_smooth(mapping = aes(x = displ, y = hwy, group = drv))
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



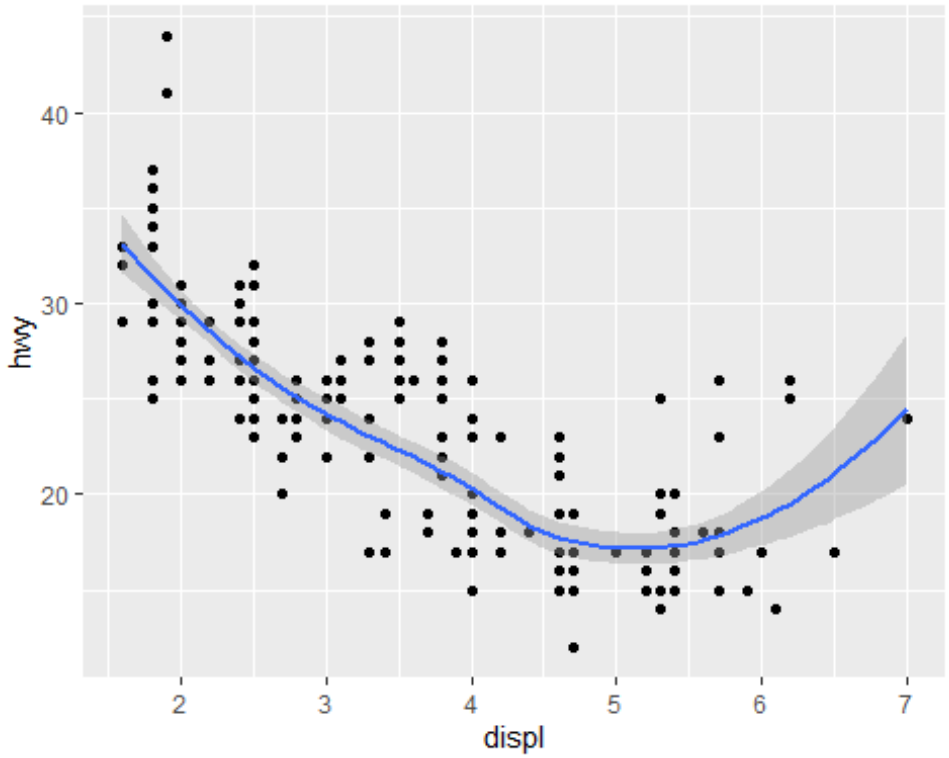

```
ggplot(data = mpg) +
  geom_smooth(
    mapping = aes(x = displ, y = hwy, color = drv),
    show.legend = FALSE
  )

## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



Pentru a afișa mai multe geoame în aceeași reprezentare, adăugați mai multe funcții geom la *ggplot()*.

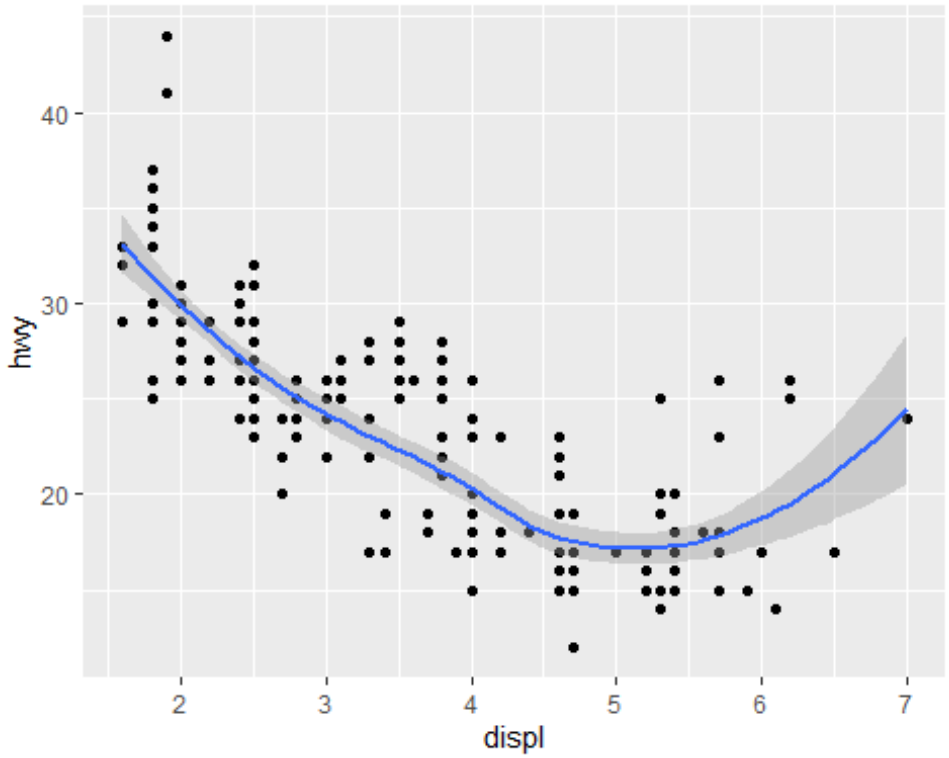
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  geom_smooth(mapping = aes(x = displ, y = hwy))
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



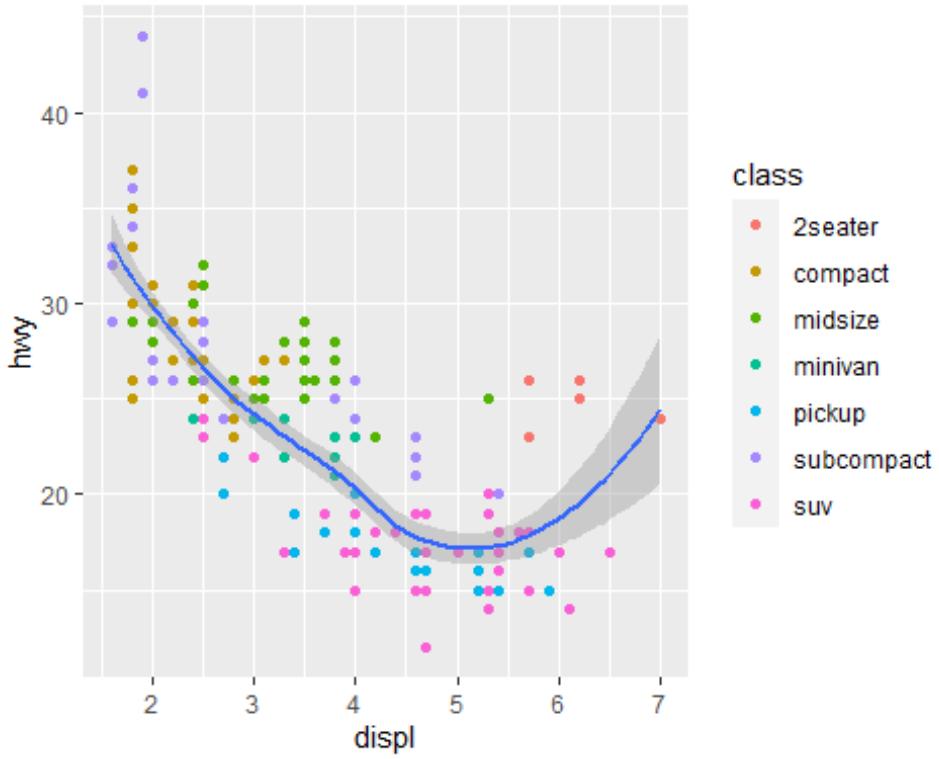
Totuși, acest lucru introduce o anumită duplicare în codul nostru. Imaginați-vă dacă ați dori să schimbați axa *y* pentru a afișa *cty* în loc de *hwy*. Ar trebui să modificați variabila în două locuri și s-ar putea să uitați să actualizați una. Puteți evita acest tip de repetare trecând un set de cartări la *ggplot()*. ggplot2 va trata aceste cartări drept globale, care se aplică fiecărui geom din grafic. Cu alte cuvinte, acest cod va produce același grafic ca și codul anterior:

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

## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



Dacă plasăm cartări într-o funcție geom, ggplot2 le va trata ca cartări locale pentru strat. Acesta va utiliza cartările pentru a extinde sau a suprascrie cartările globale numai pentru acel strat. Acest lucru face posibilă afișarea esteticilor diferite în diferite straturi.

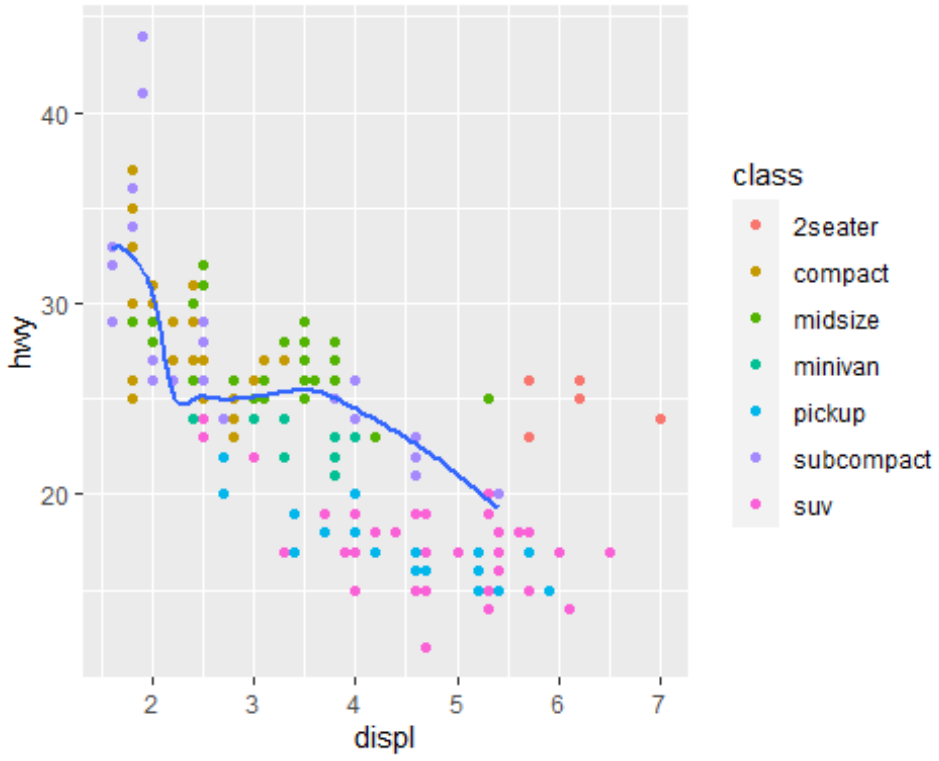
```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point(mapping = aes(color = class)) +
  geom_smooth()

## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



Putem utiliza aceeași idee pentru a specifica date diferite pentru fiecare strat. Aici, linia noastră afișează doar un subset al setului de date mpg, mașinile subcompacte. Argumentul de date locale din geom_smooth() înnlocuiește argumentul de date globale din ggplot() numai pentru acel strat.

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

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

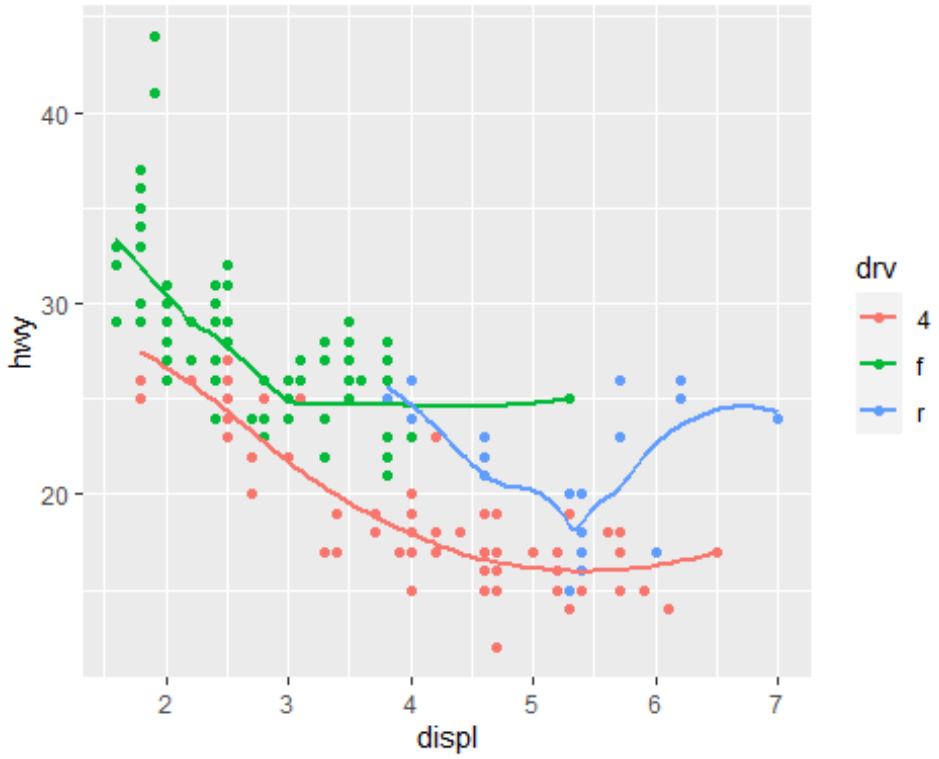


3.6.1. Exerciții

1. Ce geom veți folosi pentru a reprezenta o diagramă liniară? Un boxplot? O histogramă? O diagramă de zone?

2. Rulați următorul cod în imaginația dumneavostră și preziceți rezultatul. Apoi rulați codul în R și verificați predicțiile.

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

## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

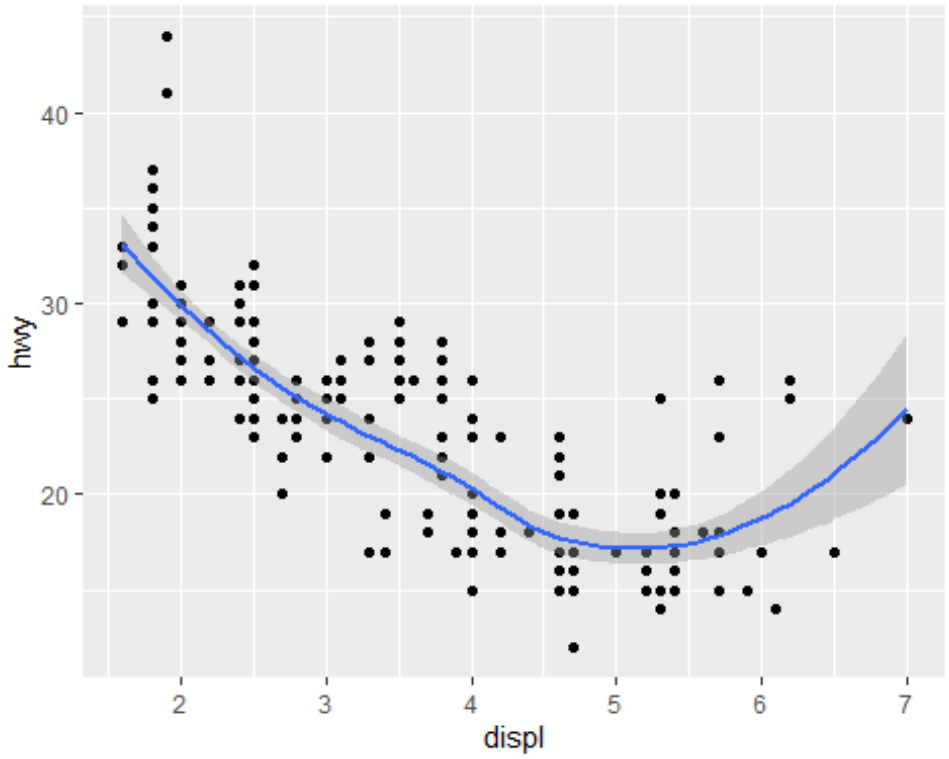


3. Ce face codul *show.legend=FALSE*? Ce se întâmplă dacă nu o utilizăm? Cum credeți, de ce am utilizat-o anterior în această secțiune?

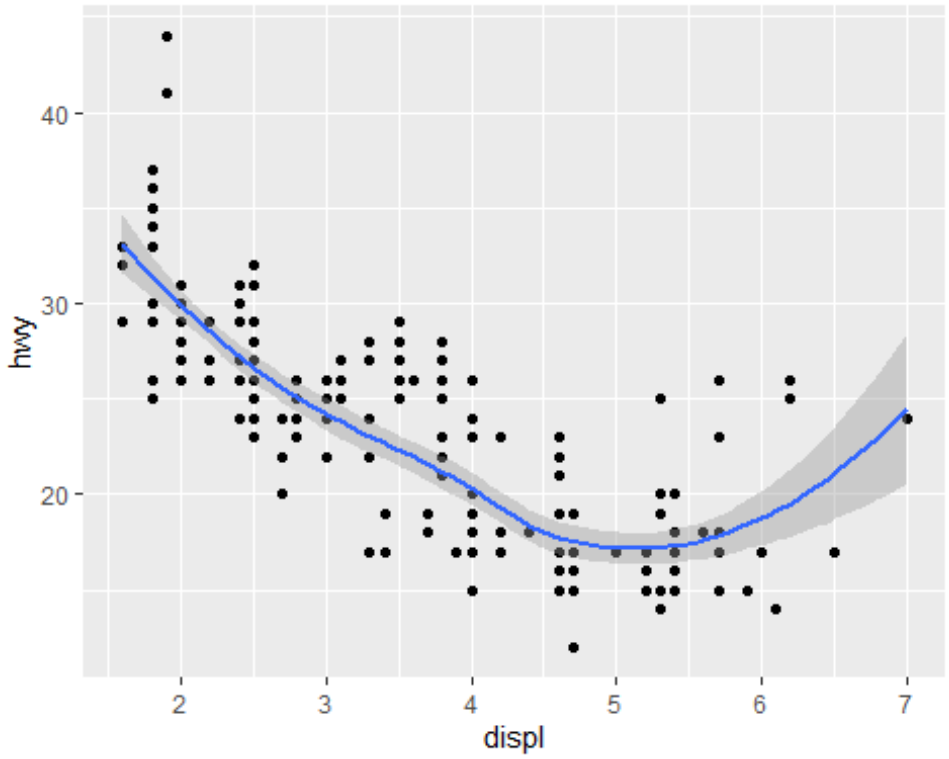
4. Ce face codul argumentului *se* pentru funcția *geom_smooth()*?

5. Aceste două reprezentări vor arăta diferit? De ce? De ce nu?

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

## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




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

## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



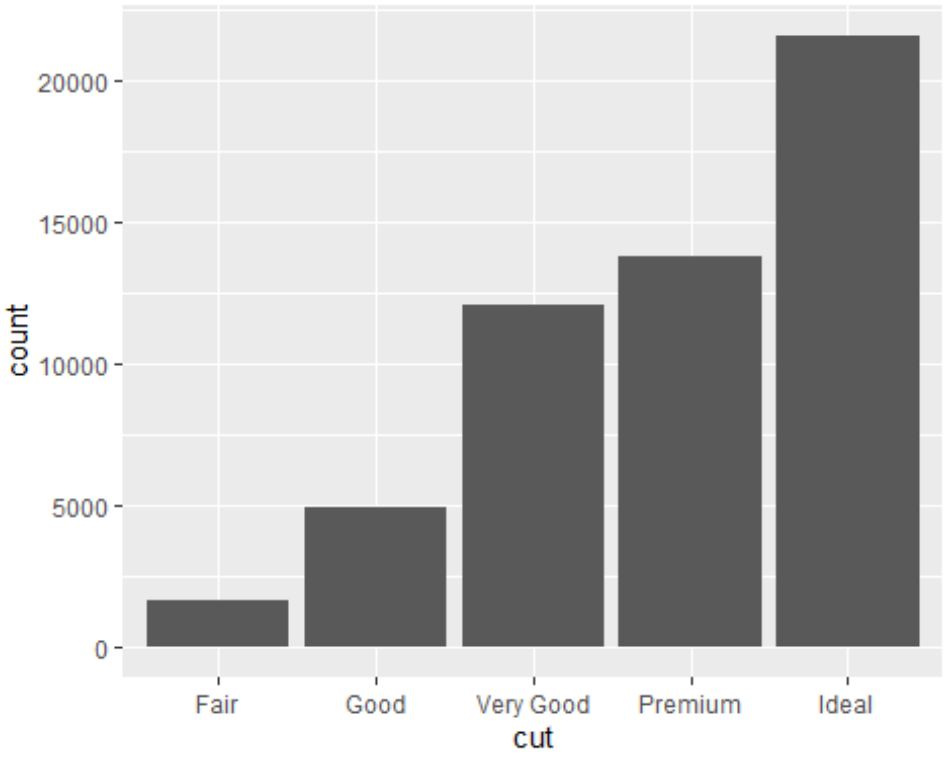
6. Creați codurile R care vor genera următoarele reprezentări: Se dau 6 Grafice.

3.7. Transformări statistice

În continuare să aruncăm o privire asupra diagramelor cu bare. Diagramele cu bare aparent par simiple, dar sunt destul de interesante și informative, deoarece dezvăluie ceva subtil despre ploturi. Să considerămo diagramă de bare simplă, așa cum o generează *`geom_bar()`*.

Următorul grafic afișează numărul total de diamante din setul de date *`diamonds`*, grupate după *`cut`* (tăietura). Setul de date *`diamonds`* se conține în ***ggplot2*** și conține informații privind ~54,000 de diamante, inclusiv prețul, caratele, culoarea, claritatea și tăietură fiecărui diamant. Graficul arată că sunt mai multe diamante cu tăieturi dă înaltă calitate decât diamante cu tăieturi de calitate scăzută.

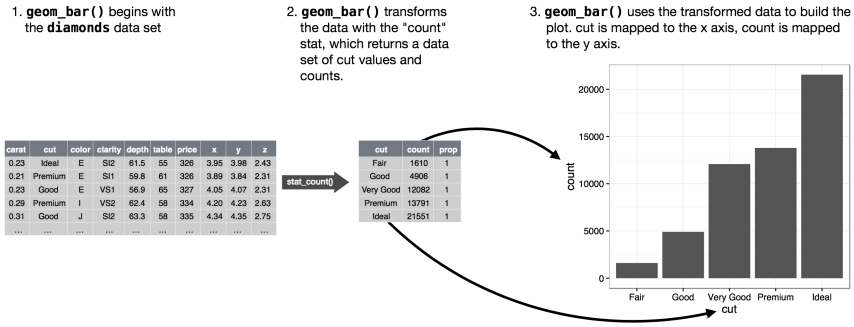
```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut))
```



Pe axa `x`, graficul afișează calitatea tăieturilor (`cut`), o variabilă de estimare a diamantelor. Pe axa y este afișat numărul de diamante, însă numerele date nu sunt variabile ce se conțin în setrul de date `diamonds`. De unde vin aceste numere? Multe grafice, cum ar fi diagramele de dispersie (scatterplots), cartografiază valorile brute ale setului de date. Alte tipuri de reprezentări, cum ar fi diagramele cu bare, calculează noi valori pentru a reprezenta:

- diagramele cu bare, histogramele și poligoanele de frecvență colectează datele din set și apoi cartează valorile colectărilor, numărul de puncte conținute în fiecare colecție.
- smoothie-urile potrivesc un model la datele noastre, apoi prezintă predicțiile dinmodel.
- boxploturile calculează un sumar sobust al distribuției și apoi afișează „boxele” special formatate.

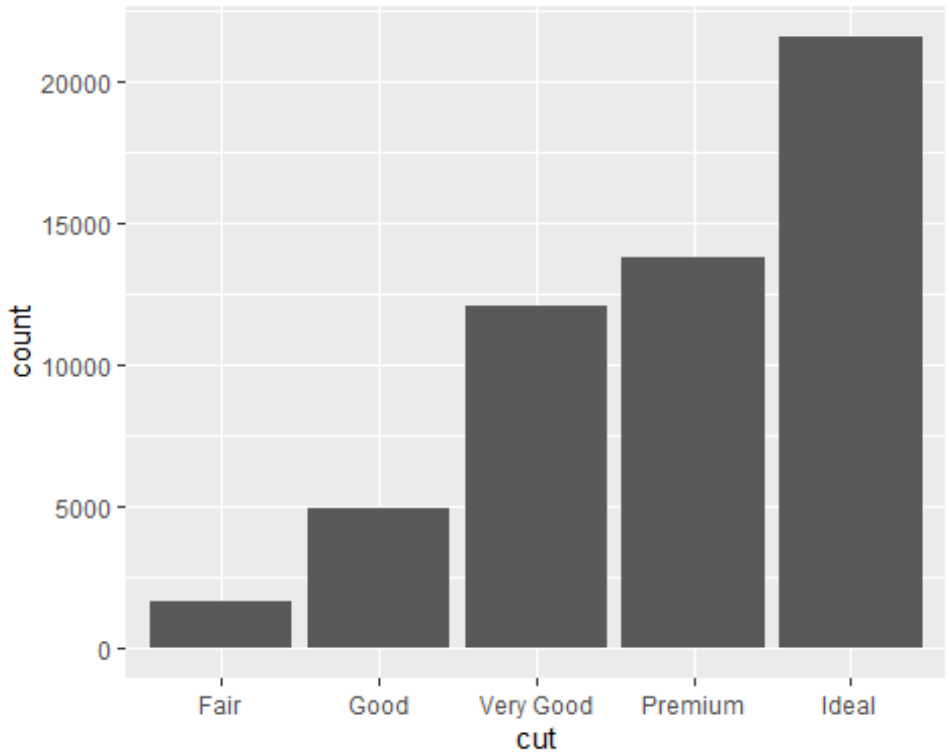
Algoritmul utilizat pentru calcularea noilor valori pentru un grafic/reprezentare se numește stat, scurt de la transformarea statistică. Figura de mai jos descrie modul în care funcționează acest proces cu `geom_bar()`.



Puteți afla ce stat folosește geomul inspectând valoare implicită pentru argumentul stat. De exemplu, *?geom_bar* arată că valoare implicită pentru stat este „*count*”, ceea ce înseamnă că *geom_bar()* folosește *stat_count()*. *stat_count()* este documentat pe aceeași pagină cu cu *geom_bar()* și mai jos putem observa o secțiune numită **„Computed variables”**. Aceasta descrie modul în care calculează două noi variabile: *count* și *prop*.

În general, putem utiilza gomurile și staturile într-un mod interschimbabil. De exemplu, putem recrea graficul anterior folosind *stat_cout()* în loc de *geom_bar()*:

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



Acest lucru funcționează deoarece fiecare geom are un stat implicit și fiecare stat are un geom implicit. Acest lucru înseamnă că putem utiliza în mod obișnuit goemurile fără să ne facem griji

cu privire la transformarea statistică de bază. Există trei motive pentru care ar putea fi necesar să utilizăm în mod explicit o statistică:

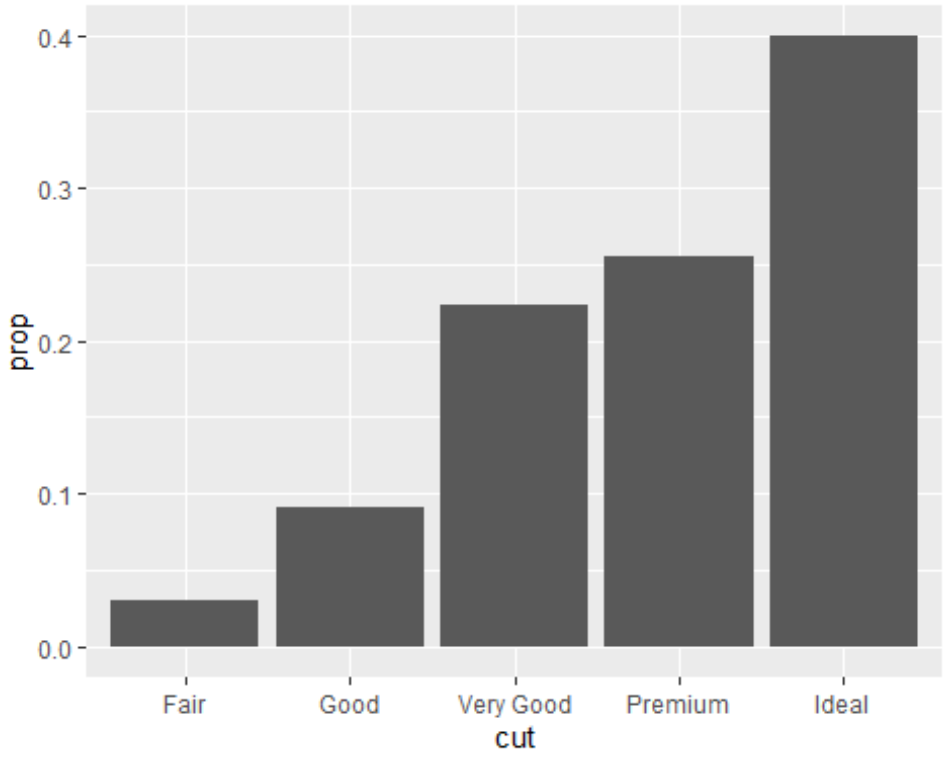
1. S-ar putea să dorim să înlocuim statul implicit. În codul de mai jos, schimbăm statul *geom_bar()* din count (implicit) în identitate. Acest lucru ne permite să cartăm înălțimea barelor la valorile brute ale unei variabile y. Din păcate, atunci când se vorbește despre diagramele cu bare, unde înălțimea barei este deja prezentă în date sau la diagrama de bare anterioară, în care înălțimea barei este generată prin numărarea rîndurilor.

```
demo <- tribble(
  ~cut,          ~freq,
  "Fair",        1610,
  "Good",        4906,
  "Very Good",   12082,
  "Premium",     13791,
  "Ideal",       21551
)
```

Nu vă faceți griji dacă nu ați mai întâlnit *<-* sau *tribble()* până acum. S-ar putea să le ghiciți semnificația din context și veți afla în curând exact ce fac exact acestea.

2. S-ar putea să dorim să înlocuim cartarea implicită de la variabile transformate la estetici. De exemplu, am putea dori să afișăm o diagramă cu bare de proporție, mai degrabă decât numărul lor:

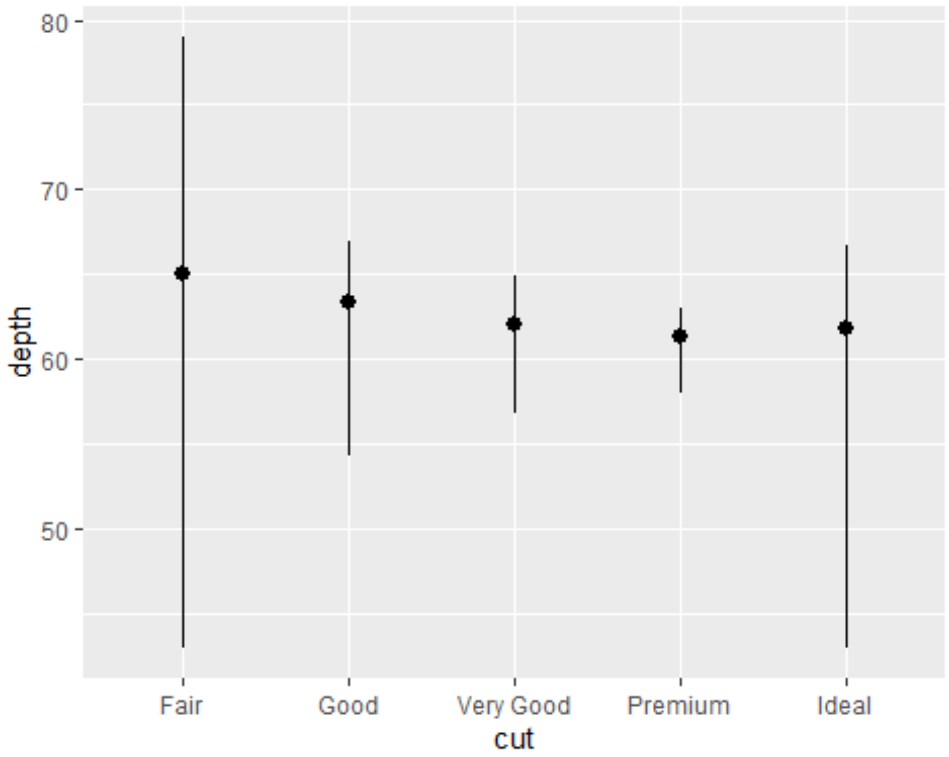
```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, y = stat(prop), group = 1))
```



Pentru a găsi variabilele calculate de stat, căutați secțiunea de ajutor intitulată „computed variables”.

3. S-ar putea să dorim să atragem o atenție mai mare asupra transformării statistice a codului nostru. De exemplu, putem utiliza *stat_summary()*, care rezumă valorile y pentru fiecare valoare x unică, pentru a atrage atenția asupra sumarului pe care îl calculăm:

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



ggplot2 oferă peste 20 de staturi (statistici) pe care le putem utiliza. Fiecare stat este o funcție, deci putem obține ajutor în mod obișnuit, de exemplu, *?stat_bin*. Pentru a vedea lista completă de statistici, putem vizita ggplot2 cheatsheet pe *https://rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf*.

3.7.1. Exerciții

1. Care este geomul implicit asocit cu *stat_summary()*? Cum am putea rescrie repreazentarea anterioară pentru a utiliza acea funcție geom în loc de funcția stat?

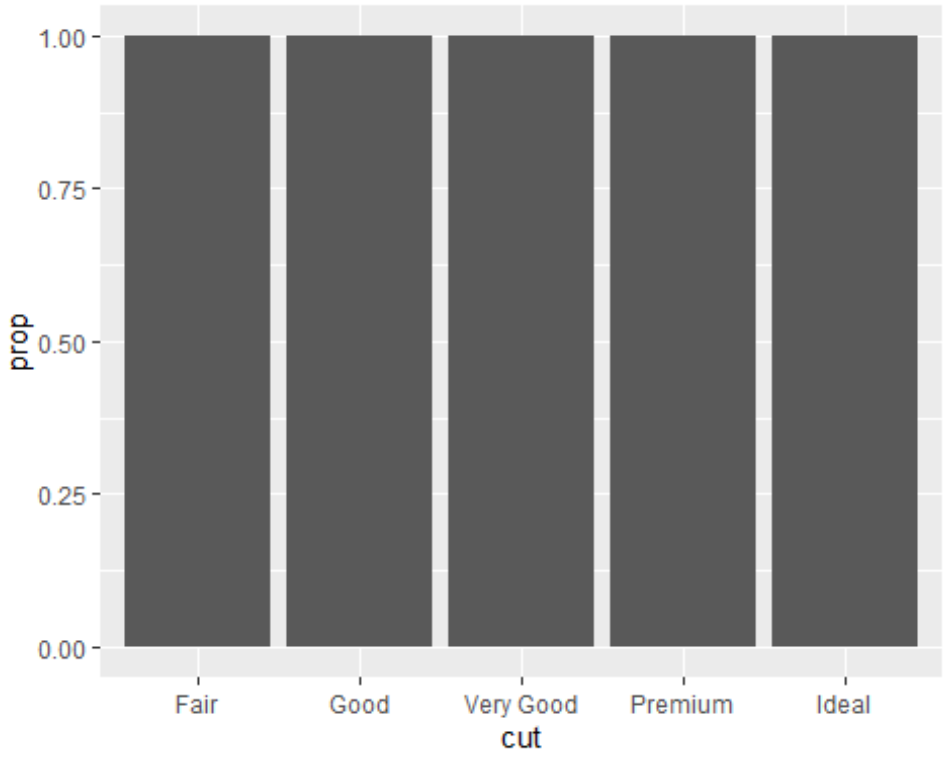
2. Ce face *geom_col()*? Prin ce este diferită de *geom_bar()*?

3. Cele mai multe geomuri și statistici vin în perechi care sunt aproape întotdeauna utilizate împreună. Citiți documentația și faceți o listă cu toate perechile. Ce au în comun?

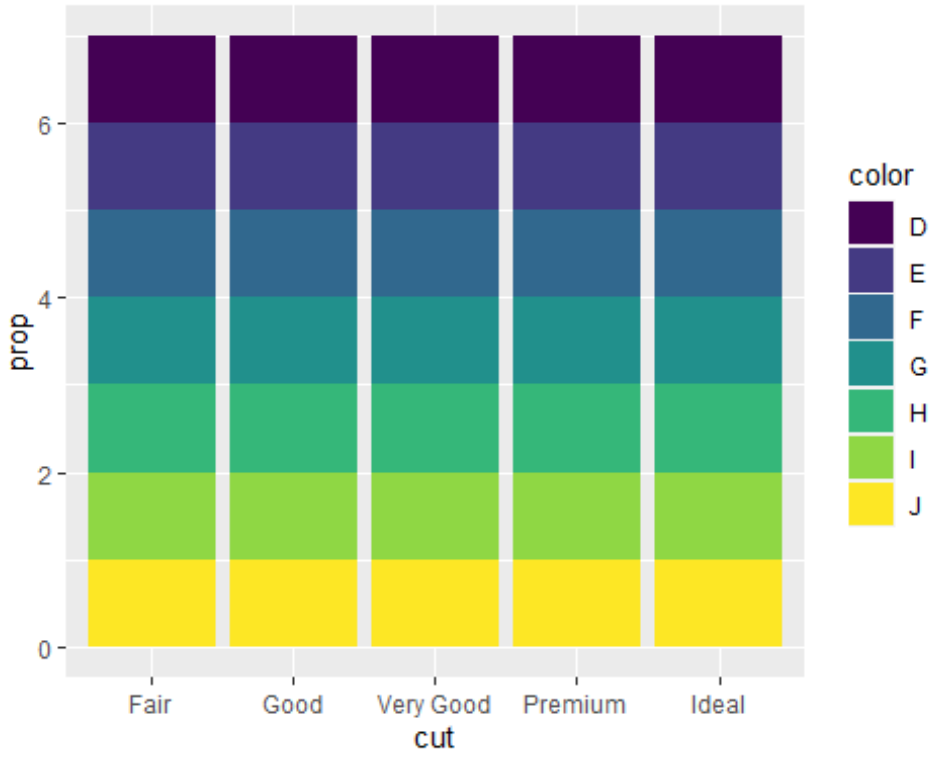
4. Ce variabile calculează *`stat_smooth()`*? Ce parametri îi controlează comportamentrul?

5. În diagrama de mai jos cu bare proporționale, trebuie să setăm group=1. De ce? Cu alte cuvinte, care este problema cu aceste două grafice?

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



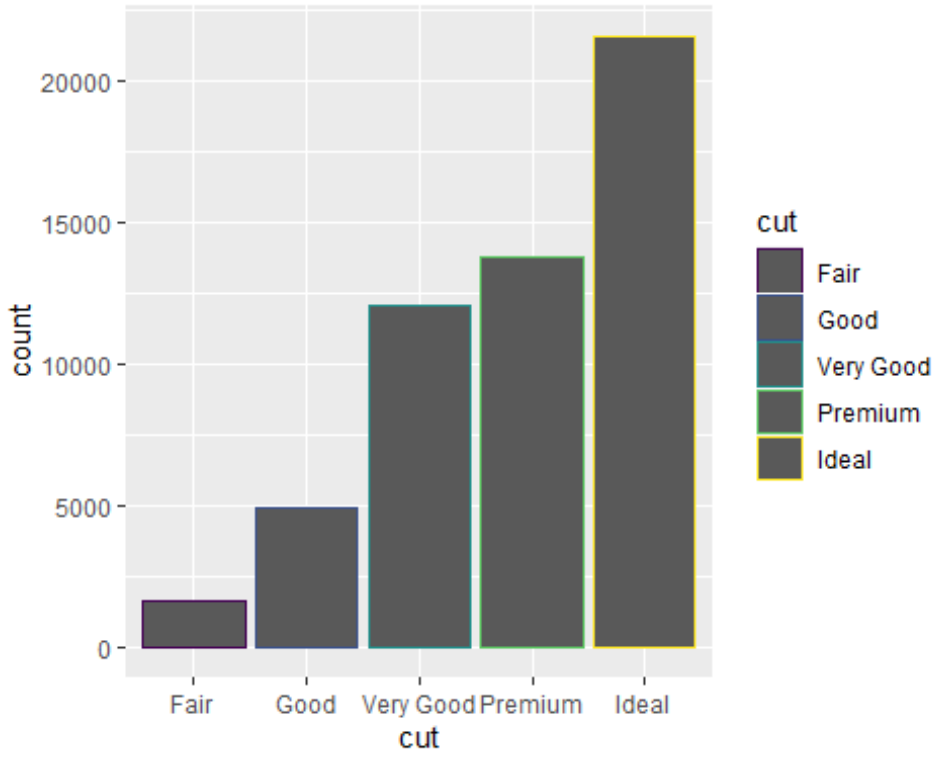

```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, fill = color, y = after_stat(prop)))
```



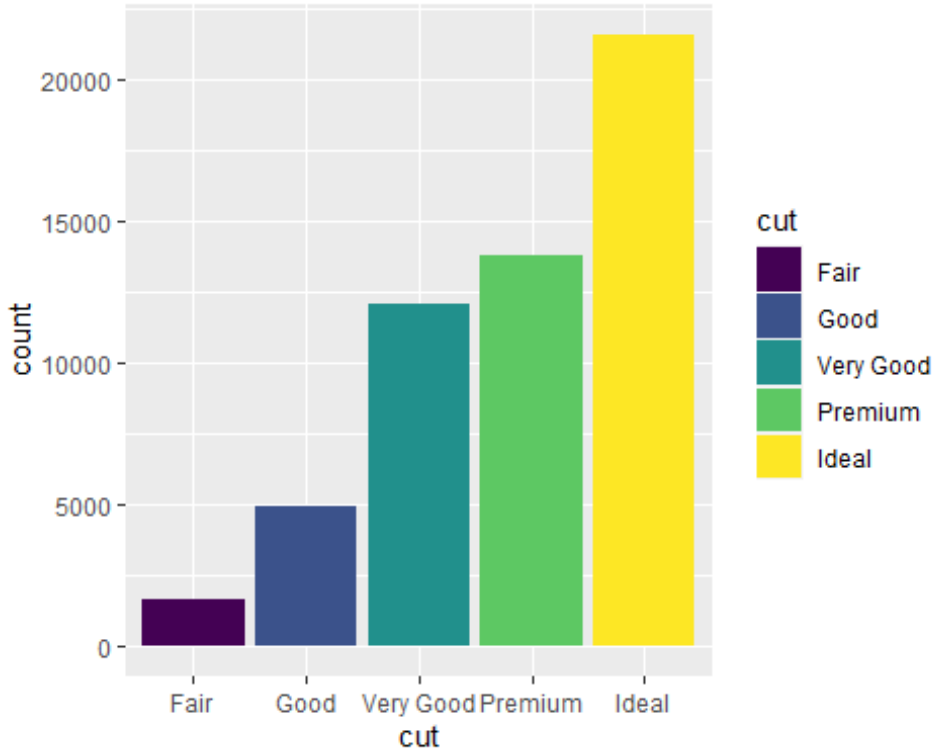
3.8. Ajustarea pozițiilor

Există încă o parte interesantă asociată cu diagramele cu bare. Putem colora o diagramăcu bare utilizând fie esteticul *colour*, fie, mai util, *fill*:

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

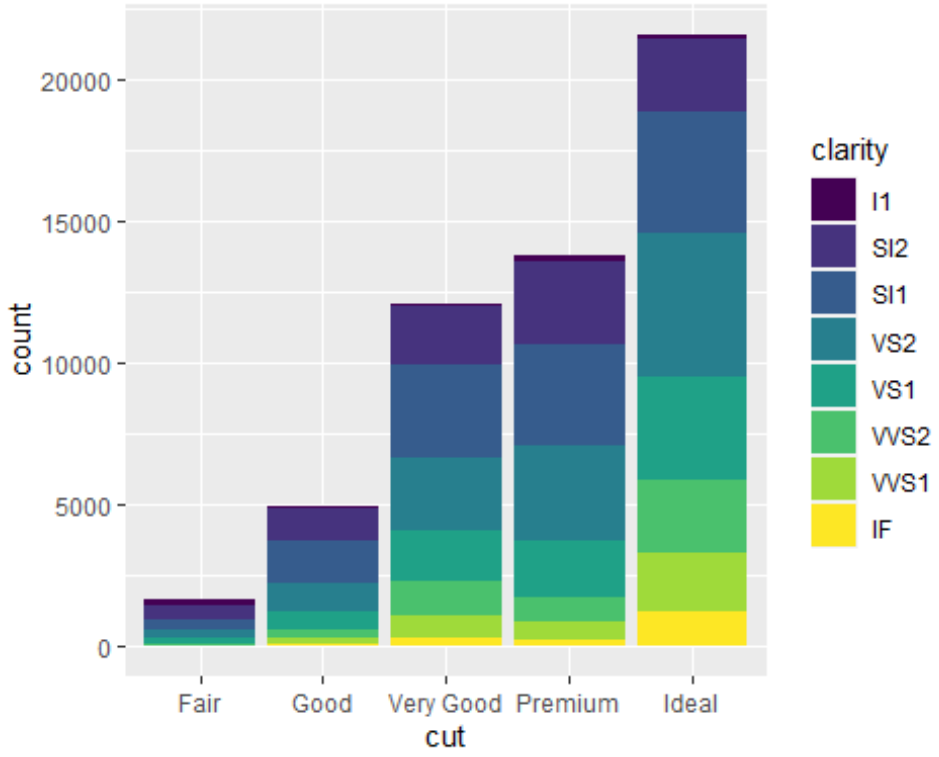



```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, fill = cut))
```



Este de reținut ce se întâmplă dacă asociem esteticul de umplere la o altă variabilă, precum ar fi *clarity*: barele sunt stivuite automat. Fiecare dreptunghi colorat reprezinită o combinație de dintre *cut* și *clarity*.

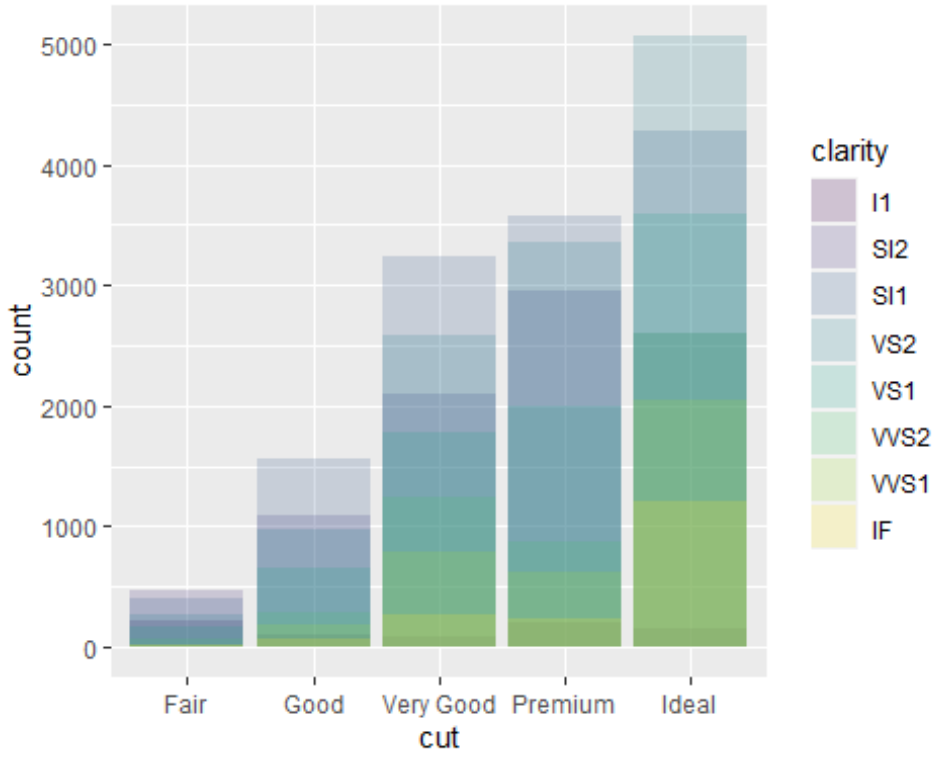
```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, fill = clarity))
```



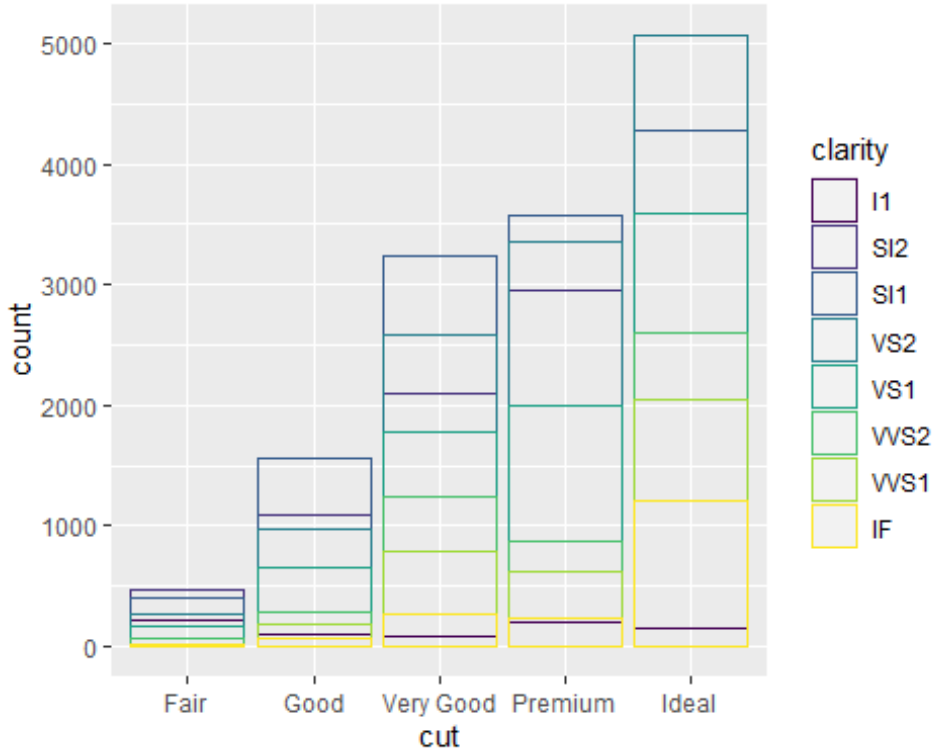
Stivuire (aranjarea) se realizează automat prin ajustarea poziției specificată de argumentul position. Dacă nu doriți o diagramă cu bare stivuite, puteți utiliza una dintre cele alte trei opțiuni: *"identity"*, *"dodge"* sau *"fill"*.

- *position="identity"* va plasa fiecare obiect exact acolo unde se încadrează în contextul graficului. Acest lucru nu este foarte util pentru bare, deoarece le suprapune. Pentru a vedea suprapunerea, fie trebuie să facem bare ușor transparente prin setarea *alpha* la o valoarea mică, fie complet transpartente prin setarea *fill=NA*.

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



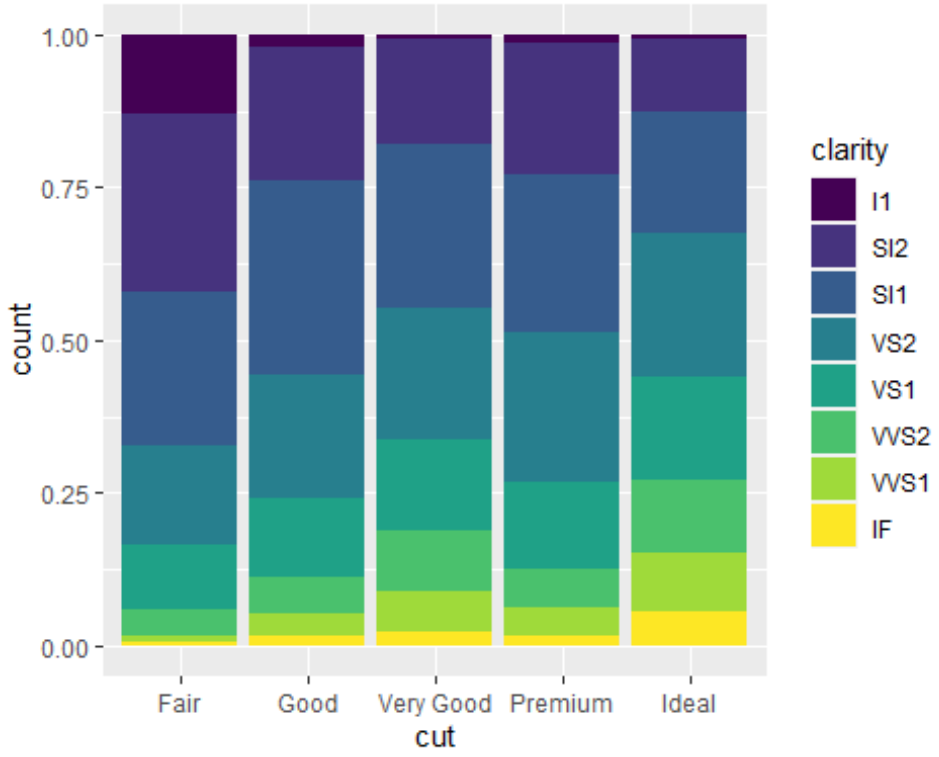

```
ggplot(data = diamonds, mapping = aes(x = cut, colour = clarity)) +
  geom_bar(fill = NA, position = "identity")
```



Reglarea poziției identității este mai utilă pentru geomurile 2d, cum ar fi punctele, unde valoarea este implicită.

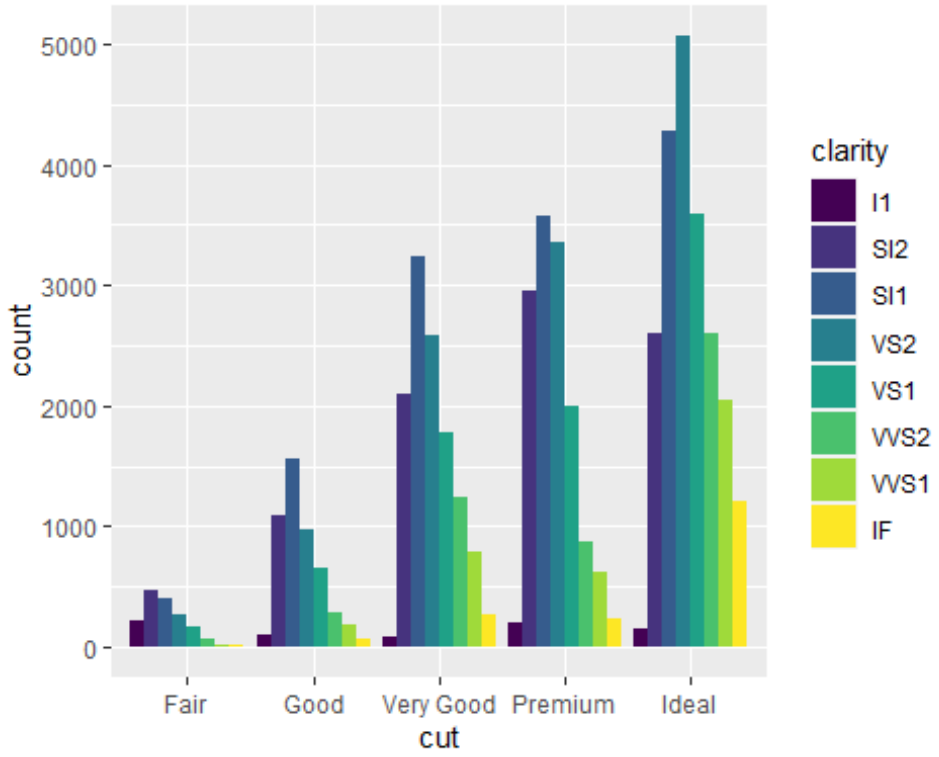
- `position = "fill"` funcționează cu stivuirea, dar face ca fiecare set de bare stivuite să aibă aceeași înălțime. Acest lucru facilitează compararea proporțiilor între grupuri.

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



- `position= "dodge"` plasează obiectele suprapuse direct unul lângă altul. Acest lucru facilitează compararea valorilor individuale.

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



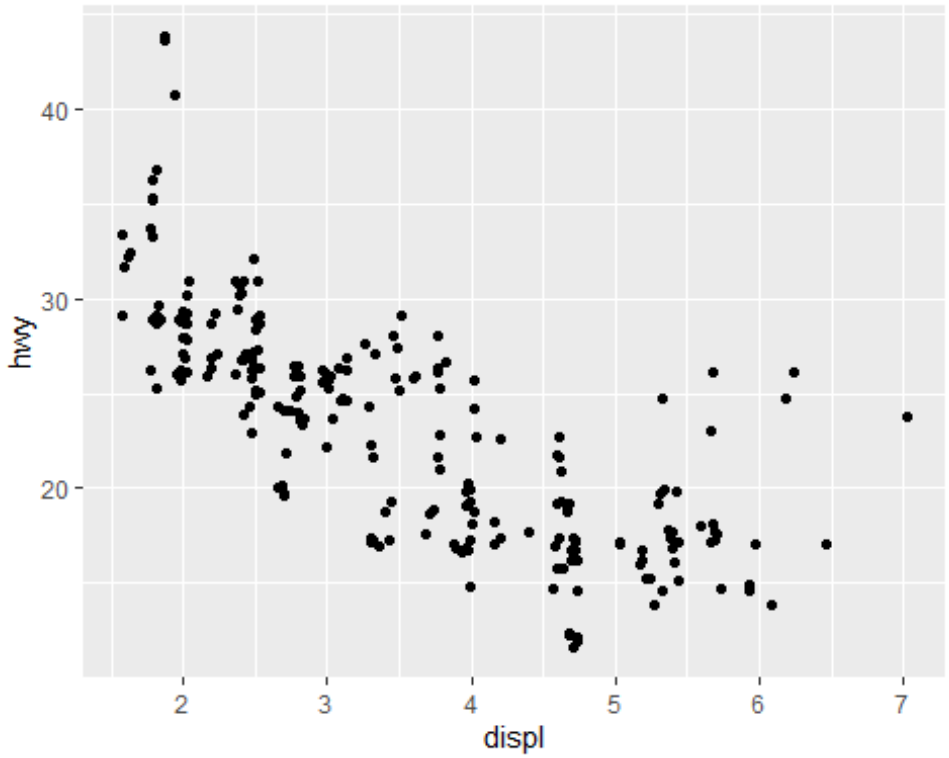
Există un alt tip de ajustare care nu este utilă pentru diagramele cu bare, dar poate fi foarte utilă pentru diagramele de dispersie (scatterplot). Ne reamintim de prima noastră diagramă de dispersie. Nu ați observat, însă scatterplotul afișează doar 126 de puncte, chiar dacă există 234 de observații în setul de date.

Valorile *hwy* și *displ* sunt rotungite, astfel încât punctele să apară o dată pe o grilă și mai multe puncte se suprapun. Această problemă este cunoscută sub numele de supra-plotare.

Acest aranjament face dificil de văzut unde este masa datelor. Punctele datelor sunt răspândite în mod egal în întregul grafic sau există o combinație specială pentru hwy și disp pentru 109 valori?

Putem să evităm această poziționare prin setarea ajustării poziției prin „jitter”. *position="jitter"* adaugă o cantitate se zgomot aleatoriu la fiecare punct. Acest lucru răspândește punctele, deoarece nu există două puncte care să primească aceeași cantitate de zgomot aleatoriu.

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



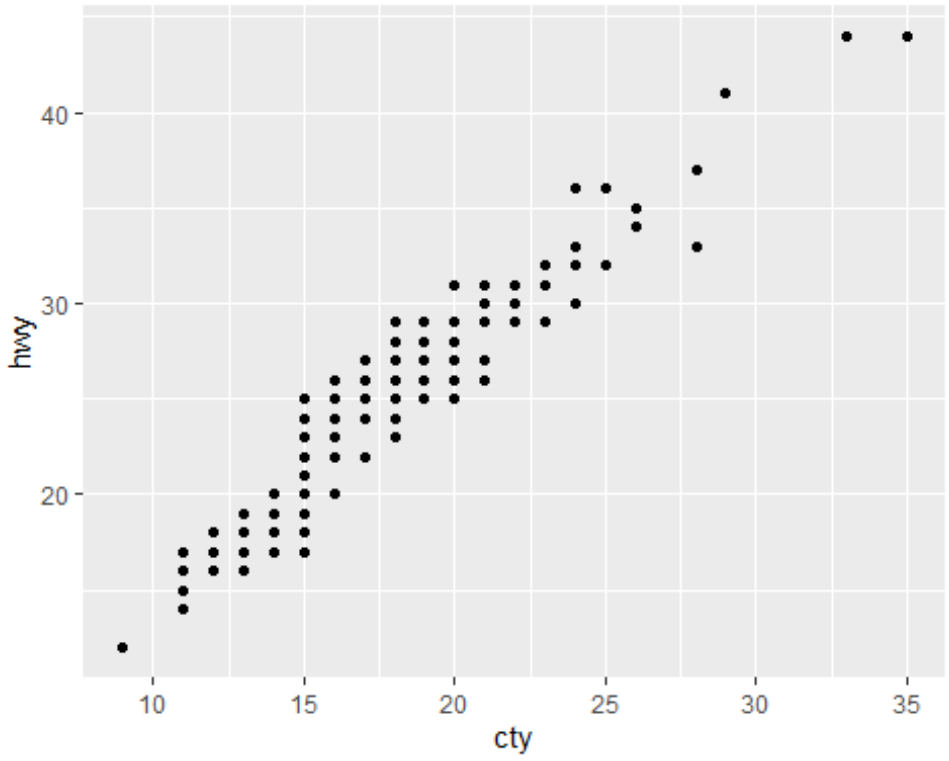
Adăugara caracterului aleatoriu pare a fi o modalitate ciudată de îmbunătățire a graficului, însă, deși face graficul mai puțin precis la scări extrem de mici, aceasta face ca graficul să fie mai relevant la scările mari. Deoarece aceasta este un procedeu util, ggplot2 vine cu o prescurtare pentru `geom_point(position="jitter"): geom_jitter()`.

Pentru a afla mai multe despre ajustarea pozițiilor, cautați paginile de ajutor asociate cu fiecare ajustare: `?position_dodge`, `?position_fill`, `?position_identity`, `?position_jitter`, și `?position_stack`.

3.8.1. Exerciții

1. Care este problema cu aceast grafic? Cum propuneți să-l îmbunătățiți?

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



2. Care parametri dini *geom_jitter()* controlează cantitatea de bruiaj?

3. Comparați *geom_jitter()* cu *geom_count()*.

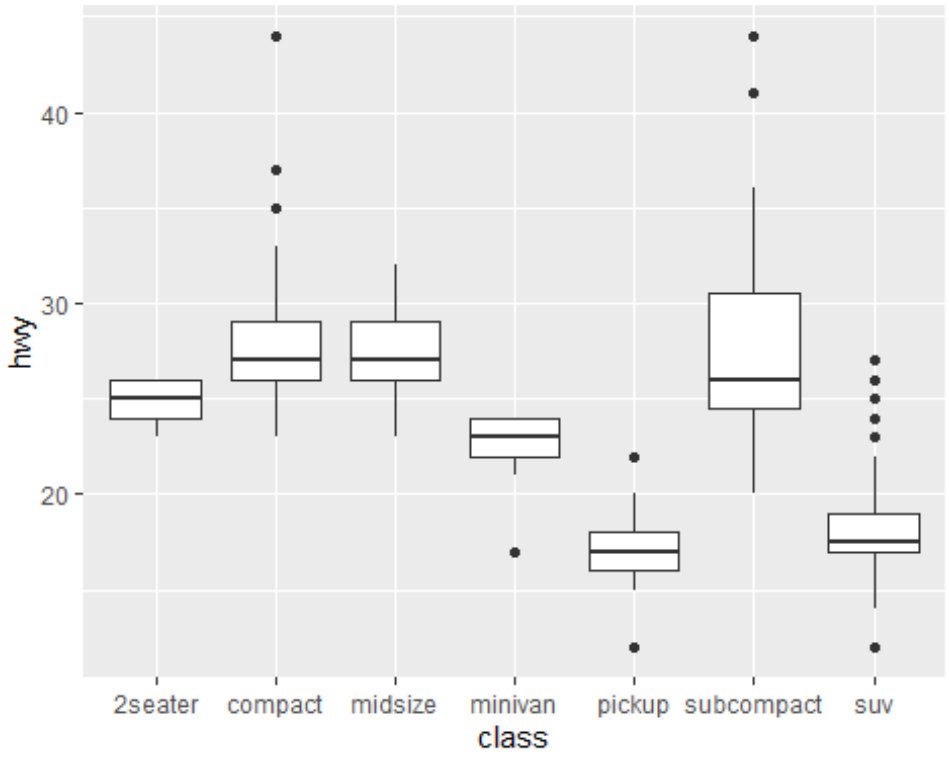
4. Care este ajustarea poziției implicită pentru *geom_boxplot()*? Creați o vizualizare a setului de date *mpg* care o demonstrează.

3.9. Sisteme de coordonate

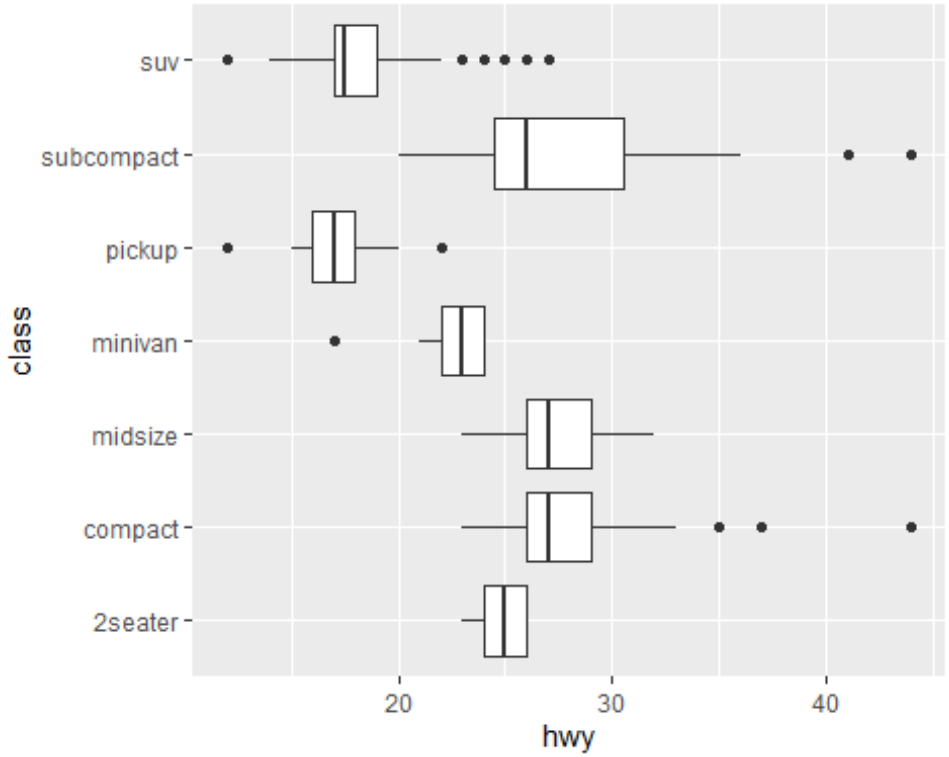
Sistemele de coordonate este probabil cea mai complicată parte a ggplot2. Sistemul de coordonate implicit este sistemul de coordonate cartezian în care pozițiile x și y acționează independent pentru a determina locația fiecărui punct. Există o serie de alte sisteme de coordonate care ocazional sunt utile.

- *coord_flip()* comută axele x și y. Acest lucru este util (de exemplu), dacă dorim boxploturi orizontale. De asemenea, comutarea este comodă pentru etichetele lungi, deoarece este greu să le potrivim fără să le suprapunem cu axa x.

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

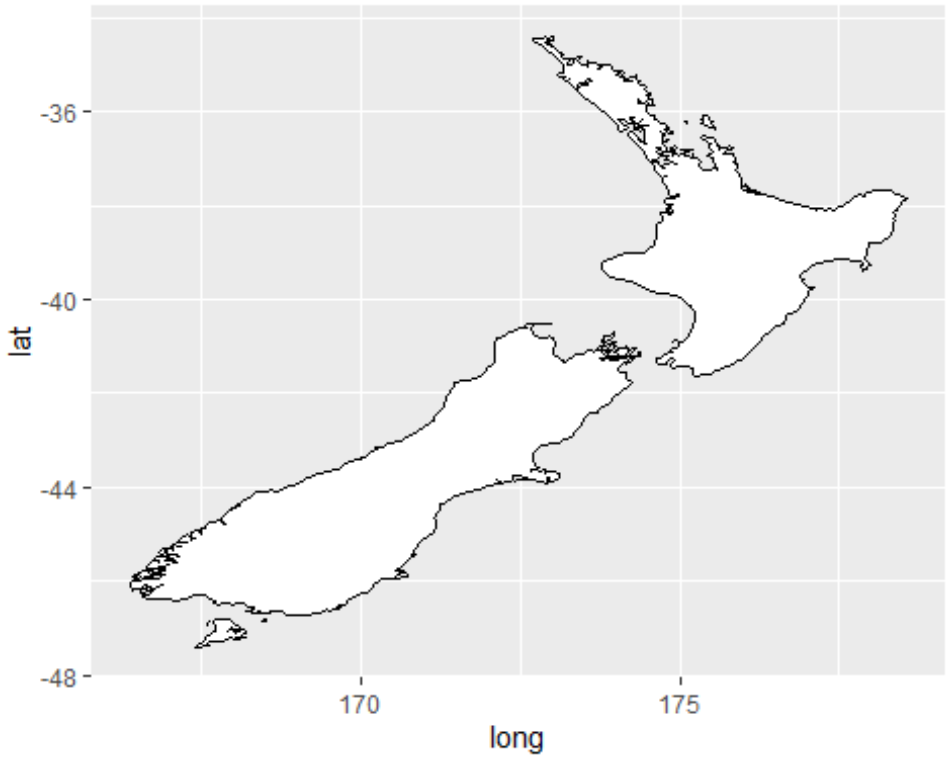



```
ggplot(data = mpg, mapping = aes(x = class, y = hwy)) +
  geom_boxplot() +
  coord_flip()
```

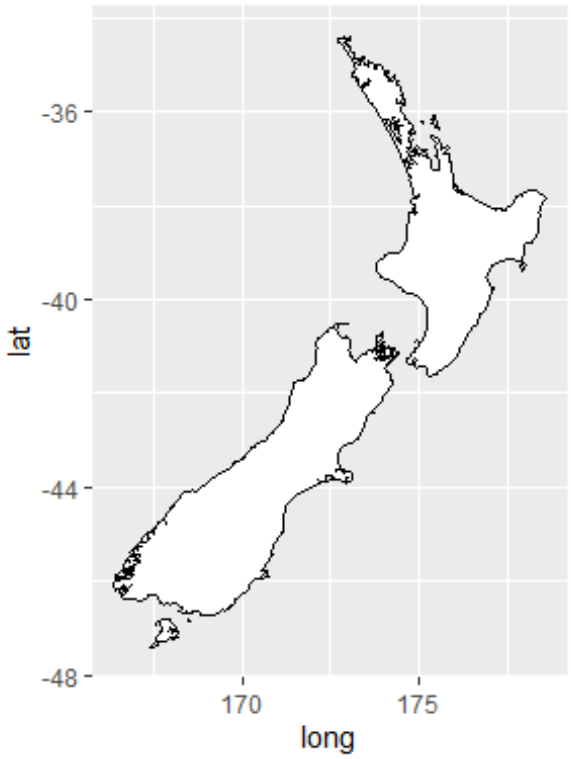


- *coord_quickmap()* setează corect raportul aspectului hărților. Acest lucru este foarte important dacă planifică să lucrpm cu date geospațiale în ggplot2.

```
nz <- map_data("nz")

ggplot(nz, aes(long, lat, group = group)) +
  geom_polygon(fill = "white", colour = "black")
```

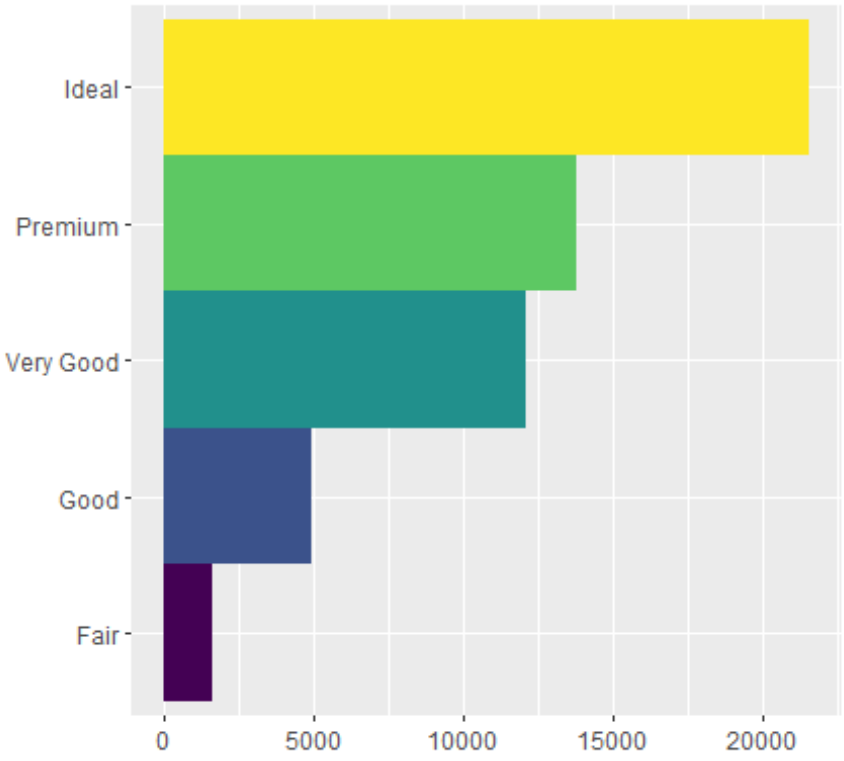



```
ggplot(nz, aes(long, lat, group = group)) +
  geom_polygon(fill = "white", colour = "black") +
  coord_quickmap()
```

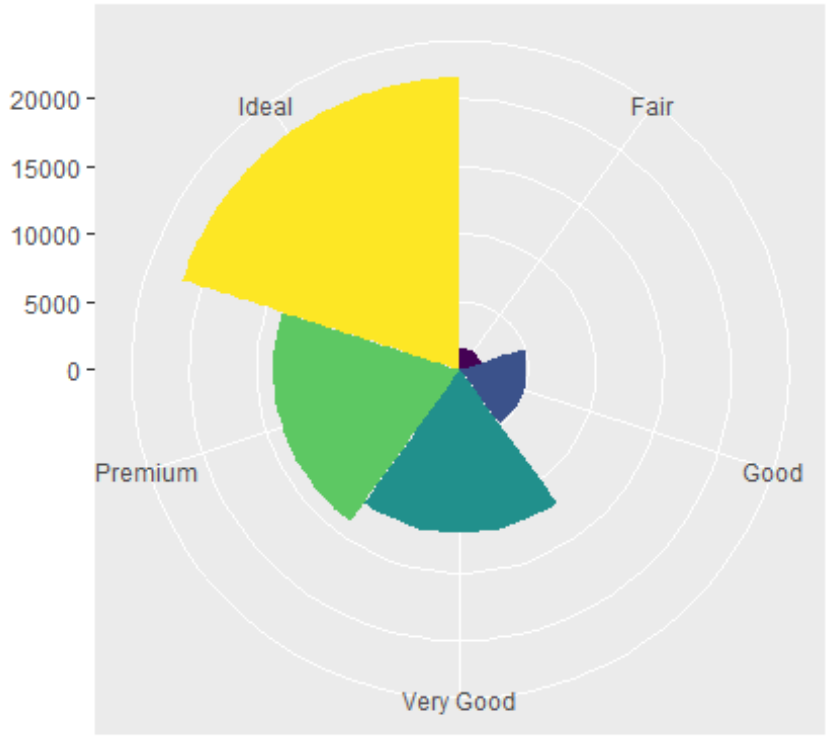


- *coord_polar()* utilizează coordonatele polare. Coordonatele polare dezvăluie o conexiune interesantă între o diagramă cu bare și o diagramă de tiă Coxcomb.

```
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)

bar + coord_flip()
```



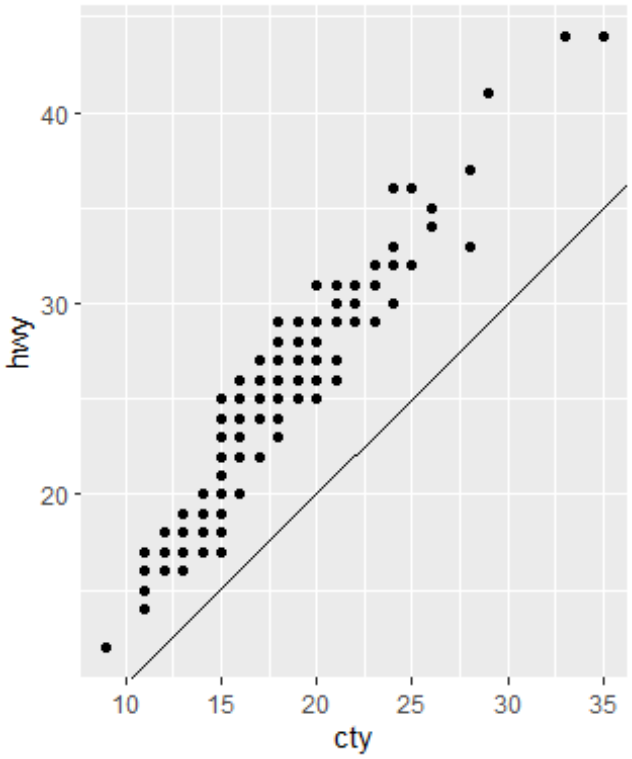

```
bar + coord_polar()
```



3.9.1. Exerciții

1. Transformați o diagramă cu bare stivuite într-un pie cart utilizând *coord_polar()*.
2. Ce face *labs()*? Citiți documentația.
3. Care este diferența dintre *coord_quickmap()* și *coord_map()*?
4. Ce fă spune graficul de mai jos despre relația dintre *city* și *highway* din setul de date *mpg*? De ce este important să utilizăm *coord_fixed()*? Ce face *geom_abline()*?

```
ggplot(data = mpg, mapping = aes(x = cty, y = hwy)) +
  geom_point() +
  geom_abline() +
  coord_fixed()
```



3.10. Gramatica stratificată a graficii

În secțiunile anterioare, am învățat mult mai mult decât cum să facem scatterploturi, barploturi și boxploturi. Am învățat bazele pentru a face orice tip de plot cu ggplot2. Pentru a vedea. Acest lucru, să adăugăm la șablonul de cod ajustări de poziție, statistici, sistemele de coordonate și fațetele:

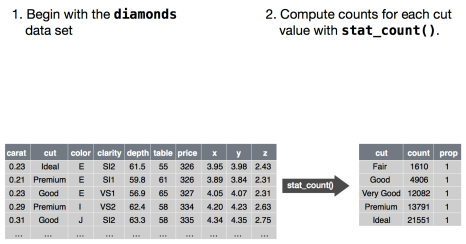
```
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(
     mapping = aes(<MAPPINGS>),
     stat = <STAT>,
```

```
    position = <POSITION>
) +
<COORDINATE_FUNCTION> +
<FACET_FUNCTION>
```

Noul nostru șablon are șapte parametri - cuvintele cheie indicate în parantezele din șablon. În practică, rareori utilizăm toți șapte parametri pentru a face un grafic, deoarece ggplot2 utilizează valori implicite utile pentru majoritatea cazurilor, cu excepția datelor, a cartărilor și a funcției geom.

Cei șapte parametri din șablon alcătuiesc gramatica graficii, un sistem formal pentru construirea ploturilor. Gramatica graficii de bazează pe ideea că puteți descrie în mod unic orice grafic ca o combinație a unui set de date, un geom, un set de cartări, o statistică, o ajustare a poziției, un sistem de coordonate și o schemă de fațetare.

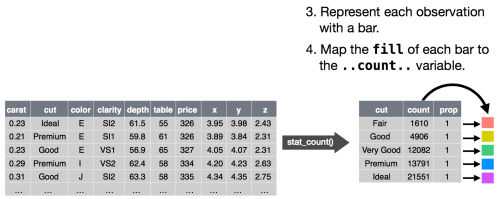
Pentru a vedea cum funcționează acest lucru, luați în considerație modul în care ați putea construi un plot de bază de la zero: puteți începe de la un set de date și apoi îl puteți transforma în informații pe care doriți să le afișați (cu o statistică).



carat	cut	color	clarity	depth	table	price	x	y	z
0.23	Ideal	E	SI2	61.5	55	326	3.95	3.98	2.43
0.21	Premium	E	SI1	59.8	61	326	3.89	3.84	2.31
0.23	Good	E	VS1	56.9	65	327	4.05	4.07	2.31
0.29	Premium	I	VS2	62.4	58	334	4.20	4.23	2.63
0.31	Good	J	SI2	63.3	58	335	4.34	4.35	2.75
...	...	...	...	...	...	...	...	...	...

cut	count	prop
Fair	1610	1
Good	4906	1
Very Good	12082	1
Premium	13791	1
Ideal	21551	1

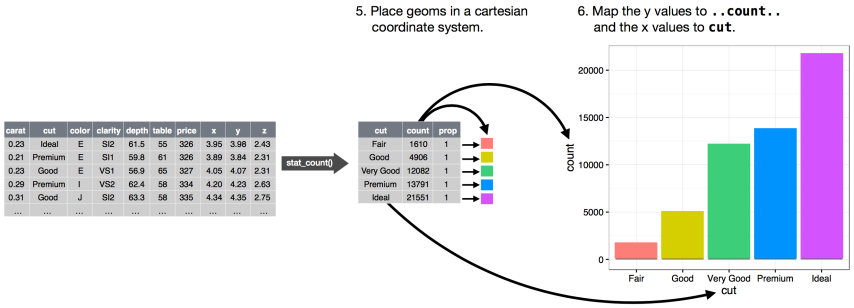
Apoi, puteți alege un obiect geometric pentru a reprezenta fiecare observație în datele transformate. Putem utiliza proprietățile estetice ale geomurilor pentru a reprezenta variabilele datelor. Trebuie să cartăm valorile fiecărei variabile la nivelurile unui esteteic.



carat	cut	color	clarity	depth	table	price	x	y	z
0.23	Ideal	E	SI2	61.5	55	326	3.95	3.98	2.43
0.21	Premium	E	SI1	59.8	61	326	3.89	3.84	2.31
0.23	Good	E	VS1	56.9	65	327	4.05	4.07	2.31
0.29	Premium	I	VS2	62.4	58	334	4.20	4.23	2.63
0.31	Good	J	SI2	63.3	58	335	4.34	4.35	2.75
...	...	...	...	...	...	...	...	...	...

cut	count	prop
Fair	1610	1
Good	4906	1
Very Good	12082	1
Premium	13791	1
Ideal	21551	1

După care trebuie să selectăm un sistem de coordonate în care să plasăm gemurile. Putem să utilizăm amplasarea obiectelor (care în sine este o proprietate estetică) pentru a afișa valorile variabilelor x și y. La această etapă vom avea un grafic complet, însă putem regla în continuare pozițiile geoamelor din cadrul sistemului de coordonate (o ajustare a poziției) sau putem împărți graficul în sub-ploturi (fațetări). Putem de asemenea extinde graficul adăugând unul sau mai multe straturi suplimentare, unde fiecare strat suplimentar folosește un set de date, un geom, un set de cartări, o statistică și o ajustare a poziției.



Putem folosi această metodă pentru a construi orice plot pe care ni-l imaginăm. Cu alte cuvinte, putem utiliza șablonul cod pe care l-am învățat în această secțiune pentru a construi sute de mii de ploturi unice.