Laborator 2-2
Vizualizare date în R

Pentru crearea de grafice avem pachetul graphics ce este în mod implicit inclus în R. Pe lângă acesta mai avem și alte pachete precum ggplot2, ggvis, lattice.

Funcția plot() inclusă în graphics este generică, adică poate fi folosită pentru diferite reprezentări în funcție de tipul de date ce îl primește. În general se dorește a fi reprezentate una sau mai multe coloane din setul de date însă pot fi reprezentate și modele liniare (evoluția unei variabile în raport cu altă variabilă).

Vom crea un grafic folosindu-ne de librăria ggplot2 (inclusă în pachetul tidyverse) și de setul de date **mpg** ce este furnizat odată cu librăria. mpg este o colecție de observații (234 observații sau rânduri) cu valori ale 11 variabile (coloanele). Putem vizualiza conținutul setului de date tastând mpg în consolă. (Nu uitați în prealabil să încărcați pachetul tidyverse cu comanda ***library(tidyverse)*).** Dacă dorim să citim mai multe despre acest set de date(de ex. documentația) tastăm ?mpg

Dintre cele 11 variabile vom folosi întâi displ (engine displacement) - dimensiunea motorului în litri și hwy (highway miles per gallon fl) – km ce pot fi parcurși pe autostradă cu un galon de carburant. Vom genera un grafic pentru a observa influența unei variabile asupra celeilalte, mai exact dacă dimensiunea motorului afectează numărul de km pe autostradă. *Graficul va arăta o relație negativă între cele două variabile, dată de tendința descrescătoare prezentată de grafic*.

Dorim să schimbăm numele etichetelor folosite la setul de date în română pentru le identifica mai ușor:

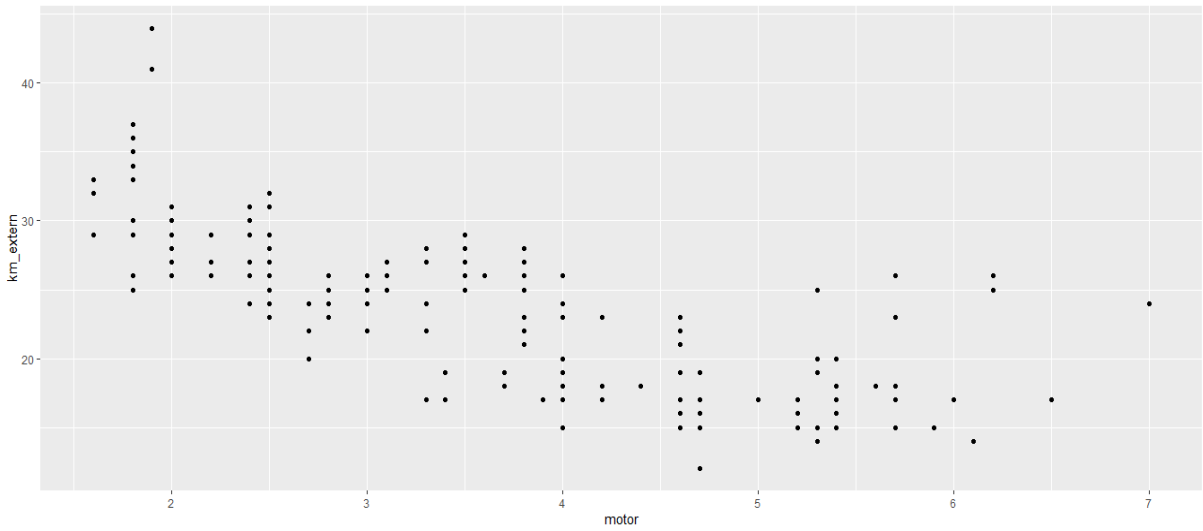
```
names (mpg)<-c("producator", "model", "motor", "an", "cyl", "transmisie", "tractiune", "km_oras", "km_extern", "carburant", "tip")
```

Grafice simple

Pentru crearea unui grafic folosim funcția ggplot() ce preia ca argument setul de date sursă ggplot(data=mpg). Totuși, această comandă nu va desena încă nimic – graficul inserat este gol. Adăugăm un strat cu un grafic de tip puncte (geom_point(). Fiecare funcție geom preia argumentul mapping. Acesta definește cum vor fi puse în corespondență variabilele din setul de date cu anumite proprietăți vizuale. Definim proprietățile vizuale ale obiectelor (aesthetic) reprezentate în grafic folosind functia aes() Obligatoriu in functia aes trebuie definite variabilele pentru axa x și pentru axa y urmate eventual de alte proprietăți vizuale ce am vrea să le definim de ex. size, colour, shape (pentru forma de afișare a punctelor), aplha (pentru gradul de transparență). Deoarece primele argumente din functia aes sunt intotdeauna x și y putem să trecem direct doar numele variabilelor - următoarele două exemple de cod sunt echivalente.

De asemenea, observați că am putut continua comanda pe un alt rând în consolă dacă am adăugat semnul + la capătul liniei

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = motor, y = km_extern))

ggplot(data = mpg) +
  geom_point(mapping = aes(motor, km_extern))
```



Următoarea comandă va afișa graficul luând în considerare încă o variabilă și anume tipul mașinii care va fi reprezentată pe grafic prin:

- **variația culorii punctelor**; ggplot va atribui în mod automat câte o culoare fiecărei valori a varibilei tip și în plus va adăuga și o legendă graficului nostru; putem acum să înțelegem punctele ce par a fi în afara liniei de pe grafic – reprezintă mașini sport care deși au motor mare pot avea un număr de km mai mari parcurși deoarece sunt mai ușoare.

```
ggplot(data = mpg) +
   geom_point(mapping = aes(x = motor, y = km_extern, color=tip))
```

Obs: comparați exemplul anterior cu următorul în care proprietatea color nu va fi legată de o variabilă în cadrul funcției aes() ci va fi folosită pentru a specifica o culoare pentru elementele de pe grafic

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

- **variația dimensiunii punctelor**; obținem un mesaj de avertizare deoarece nu este indicat să legăm o variabilă neordonată precum tipul mașinii de o proprietare ordonată precum dimensiunea

```
ggplot(data = mpg) +
   geom_point(mapping = aes(x = motor, y = km_extern, size=tip))
```

- **variația gradului de transparență a punctelor**

```
ggplot(data = mpg) +
   geom_point(mapping = aes(x = motor, y = km_extern, alpha=tip))
```

- **variația formei punctelor**; obținem un mesaj de avertizare deoarece pe grafic pot fi afișate o variație de maxim 6 forme; una dintre valorile variabilei tip nu va primi o reprezentare deci nu va putea fi afișată pe grafic, ceea ce înseamnă că 62 de valori din setul de date au fost omise

```
ggplot(data = mpg) +
   geom_point(mapping = aes(x = motor, y = km_extern, alpha=tip))
```

Fațete

Până acum am văzut o modalitate de a adăuga variabile pe grafic folosindu-ne de anumite proprietăți vizuale pe care le puteam defini cu funcția aes(). O altă modalitate de a adăuga variabile (în special pentru variabilele nominale) este de a realiza subgrafice (fațete) - câte unul pentru a afișa fiecare subset de date din variabila respectivă.

Pentru a realiza graficul cu fațete după o singură variabilă vom folosi funcția *facet_wrap().* Primul argument din funcție trebuie să fie o formulă.

Reminder: exemple de formule în R:

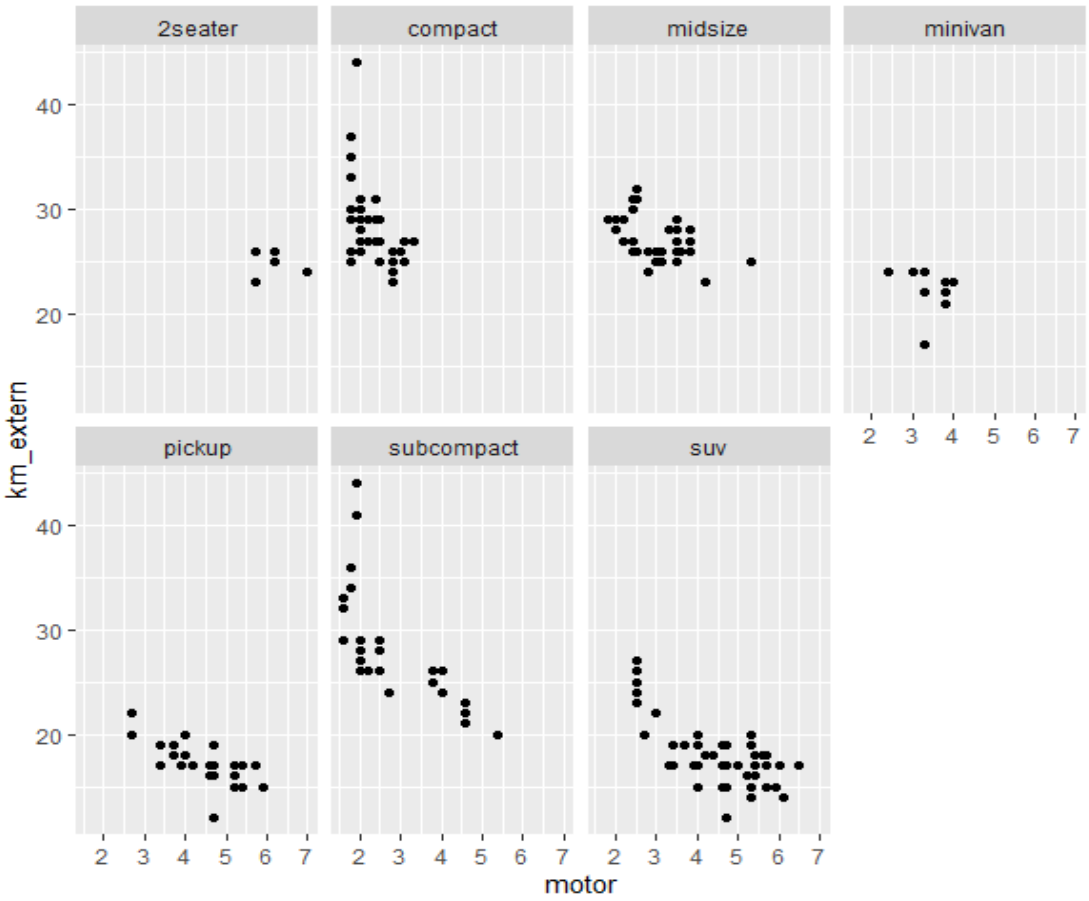
```
c<- y~x

d<-y~x+b

e<- ~a+b
```

Formulele sunt folositoare pentru a exprima o relație între variabile. Mai mult, ne permit ca valorile care ar fi atribuite simbolurilor din formulă să nu fie accesate la momentul creării formulei. Acest lucru explică de ce în anumite apeluri de funcții cum este funcția facet_wrap() folosesc formule: pentru că ne permit să preluăm valorile variabilelor fără a fi evaluate și să citim formula, interpretarea ei fiind făcută în funcție. În ultimul exemplu, putem vedea că într-o formulă nu este obligatoriu să avem definită o variabilă dependentă.

Următorul fragment de cod ne va crea un grafic cu mai multe subgrafice, fiecare pentru un subset de valori ale variabilei/coloanei tip (respectiv 2seater, compact, midsize, minivan etc).

Variabila care trebuie transmisă funcției facet_wrap() trebuie să fie discretă:

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = motor, y = km_extern)) +
  facet_wrap(~ tip, nrow = 2)
```



Cel de-al doilea argument care este transmis in funcția facet_wrap() și anume *nrow* se referă la numărul de rânduri pe care il ocupă fațetele. Puteți crea graficul fără argumentul respectiv și observați diferența. Asemănător funcționează și argumentul *ncol* pentru numărul de coloane.

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = motor, y = km_extern)) +
  facet_wrap(~ tip)
```

Pentru a crea un grafic în care să combinăm 2 variabile, folosim funcția *facet_grid().* Și la acestă funcție primul argument trebuie să fie o formulă, însă de această dată, formula trebuie să conțină două variabile: tractiune (ce are valori 4 – tracțiune 4x4, f -tracțiune față, r – tracțiune spate) afișată pe axa Oy, și cyl (nr cillindri: 4,5,6,8) separate de operatorul ~. Nu înseamnă că există o relație între cele două variabile pe setul nostru de date, ci vrem doar să afișăm valorile în raport una de alta:

```
ggplot(data = mpg) +
   geom_point(mapping = aes(x = motor, y = km_extern)) +
   facet_grid(tractiune ~ cyl)
```

Am putea să folosim funcția facet_grid() și pentru a crea subgrafice cu o singură varibilă (deci, o variantă la facet_wrap()); în acest caz vom înlocui cu . variabila ce va fi omisă; următorul exemplu este asemănător cu ceea ce am realizat prin facet_wrap():

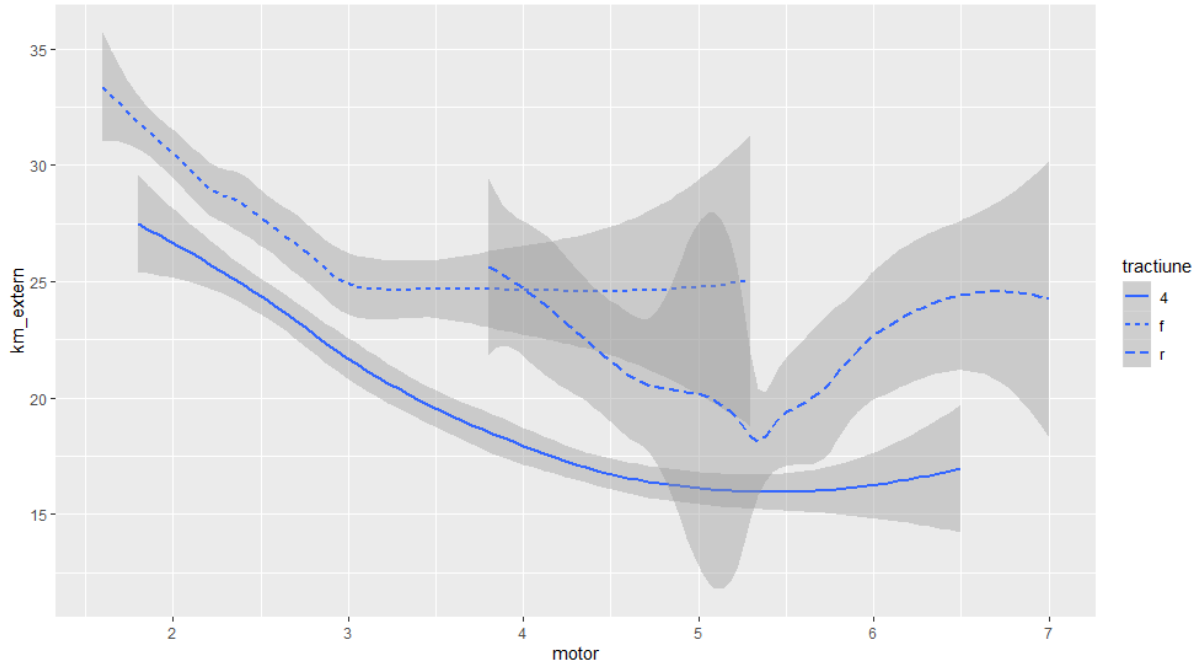
```
ggplot(data = mpg) +
   geom_point(mapping = aes(x = motor, y = km_extern)) +
   facet_grid(. ~ tip)
```

Exemplele de până acum foloseau puncte pentru reprezentarea valorilor pe grafic iar pentru aceasta foloseam funcția geom_point(). Alegând o altă funcție precum geom_smooth() graficul va fi o linie trasată după valorile din setul de date:

```
ggplot(data = mpg) +
    geom_smooth(mapping = aes(x = motor, y = km_extern))
```

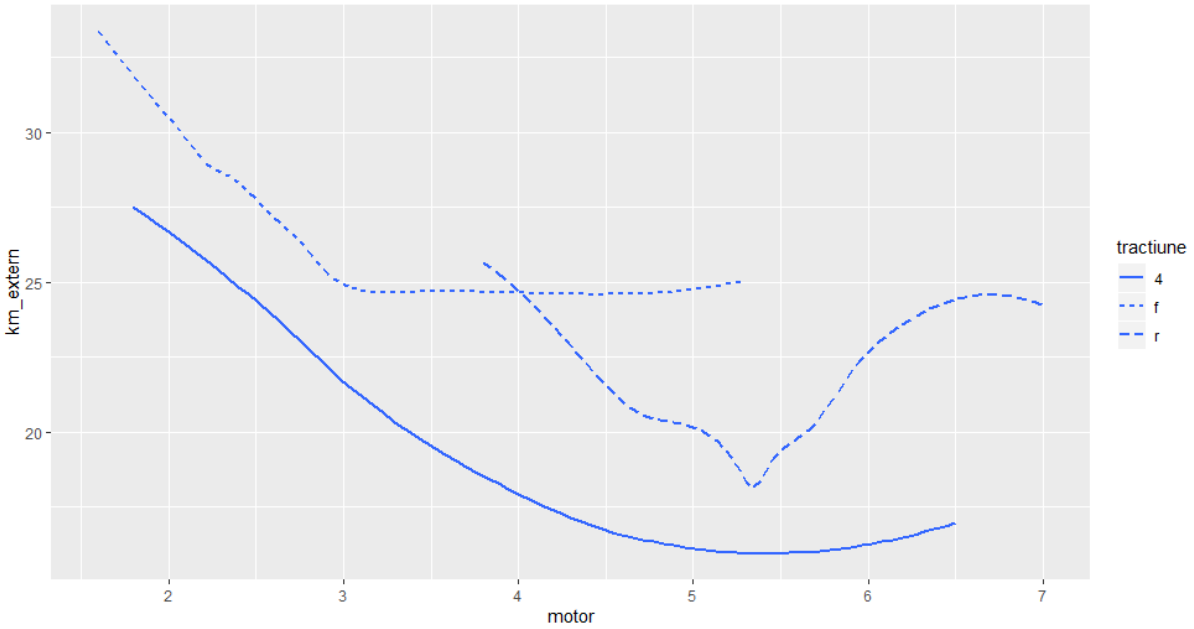
Proprietățile vizuale care le definim pentru o anume reprezentare s-ar putea să nu se potrivească pentru alta. De exemplu putem defini forma geometrică pentru punct (cerc, romb, patrat etc) dar în cazul liniei avem tipul liniei (continuă, punctată). Putem folosi această caracteristică pentru a reprezenta o a treia variabilă precum tracțiunea: una din liniile de pe grafic reprezintă valorile pentru mașini cu tracțiune integrală, altă pentru cele cu tracțiune pe față, alta pentru cele cu tracțiune pe spate.

```
ggplot(data = mpg) +
    geom_smooth(mapping = aes(x = motor, y = km_extern, linetype = tractiune))
```



Observați că linia este însoțită de reprezentarea unui interval *(intervalul de încredere*) care reprezintă limitele între care se regăsesc datele noastre (datele ce cuprind eroarea standard “standard error ”). Putem să renunțăm la reprezentarea acestui interval dacă alegem valoarea false pentru parametrul *se*.

```
ggplot(data = mpg) +
      geom_smooth(mapping = aes(x = motor, y = km_extern, linetype = tractiune), se=FALSE)
```



Putem să afișăm mai multe reprezentări grafice prin adăugarea de mai multe funcții geom – putem vizualiza atât valorile cu puncte (geom_point) cât și cu linie (geom_smooth):

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = motor, y = km_extern)) +
  geom_smooth(mapping = aes(x = motor, y = km_extern))
```

Modul anterior de a adăuga mai multe reprezentări grafice (puncte și linie) pentru aceleași variabile (motor și km_extern) ne creează cod duplicat ceea ce înseamnă că am avea nevoie să facem modificări în două locuri dacă ar trebui să înlocuim una din variabile. Pentru a evita acest lucru putem să transmitem un set de mappings în funcția ggplot care vor fi considerate globale și vor fi aplicate tuturor reprezentărilor grafice:

```
ggplot(data = mpg, mapping = aes(x = motor, y = km_extern)) +
  geom_point() +
  geom_smooth()
```

În cazul în care pe lângă setările mappings globale am folosi și în interiorul funcțiilor geom_point sau geom_smooth alte setări, acestea din urmă le vor suprascrie sau le vor extinde pe cele globale. Asta înseamnă că putem avea caracteristici vizuale diferite în fiecare strat sau pentru fiecare reprezentare grafică.

R permite nu doar caracteristici vizuale diferite în fiecare reprezentare grafică ci chiar *seturi de date diferite*: de exemplu linia va fi folosită doar pentru a reprezenta mașinile ce corespund tipului subcompact

```
ggplot(data = mpg, mapping = aes(x = motor, y = km_extern)) +
  geom_point(mapping = aes(color = tip)) +
  geom_smooth(
    data = filter(mpg, tip == "subcompact"),
  )
```

Transformări statistice

Reprezentările grafice precum graficele de tip coloană, histogramele sau poligoanele de frecvență grupează datele și apoi reprezintă numărul de elemente din fiecare grup

Smoothers

boxplots calculează un rezumat al distribuției valorilor

Algoritmul folosit pentru a calcula valori noi pentru o reprezentare grafică este numit ”stat” prescurtarea de la transformări statistice. Se poate vedea ce algoritm stat folosește fiecare reprezentare grafică geom dacă verificăm valoarea argumentului stat. De exemplu prin ?geom_bar observăm că valoarea implicită pentru stat este ”count” ceea ce înseamnă că se apelează funcția stat_count() – descrisă pe aceeași pagină cu funcția geom_bar(). Funcțiile geom_bar() și stat_count() pot fi folosite alternativ și acest lucru este posibil deoarece fiecare geom are un anume algoritm stat și fiecare stat are o anume reprezentare geom. Aceasta înseamnă că putem folosi reprezentările grafice fără să ne preocupe care sunt transformările statistice de la bază.

Există cazuri în care am dori să schimbăm valoarea predefinită a argumentului stat. De exemplu am putea folosi valoarea ”identity”, ceea ce ne permite să reprezentăm valorile variabile y (înălțimea coloanelor) și nu valorile calculate precum count.