Vizualizarea datelor cu ajutorul pachetului Tidyverse

Pachetul ggplot2

1 Vizualizarea datelor cu `ggplot2`

Tehnicile de vizualizarea a datelor joacă un rol important în orice analiză statistică, putând conduce la identificarea de noi caracteristici, perspective sau pattern-uri în acestea. Scopul acestei secțiuni este de a introduce o serie de elemente de vizualizare a datelor prin intermediul pachetului ggplot2. Pachetul a fost dezvoltat inițial de Hadley Wickham (care încă menține și îmbunătățește activ pachetul), este parte integrantă din suita tidyverse și este bazat pe, ceea ce se numește în teoria vizualizării, *gramatica elemntelor grafice* introdusă de Leland Wilkinson în cartea The Grammar of graphics (de aici vine și prefixul `gg` - grammar of graphics). Similar cu gramatica unei limbi, în care diferite elemente precum substantive, verbe, articole, prepoziții, etc. se pot combina după anumite reguli și *gramatica elementelor grafice* prezintă o serie de reguli care să permită construcția de grafice statistice prin combinarea mai multor tipuri de *straturi* (*layers*) (Wilkinson 2005). Astfel, ideea principală a pachetului `ggplot2` este de a specifica independent o serie de componente constructive ale graficului (obiecte geometrice), organizate în straturi, ce urmează a le combina pentru a genera figura dorită (Wickham 2016).

Ne dorim ca la finalul acestei secțiuni, să fim capabili să creăm grafice care să semene cu cele de mai jos:

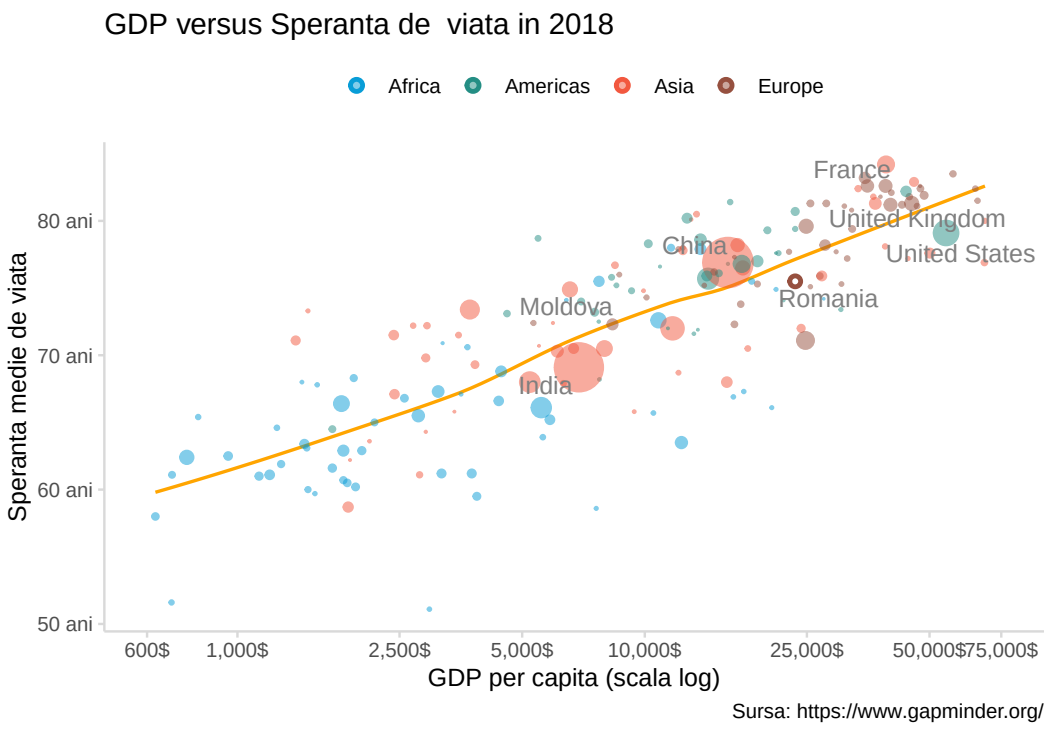


Fig. 1: Exemplu de vizualizare realizata cu ajutorul pachetului ggplot2

Fig. 2: Exemplul 2 de vizualizare realizata cu ajutorul pachetului ggplot2

Înainte de prezentarea elementelor de bază ce compun un grafic folosind pachetul `ggplot2` și de a descrie fiecare componentă în parte, vom specifica care sunt datele (setul/seturile de date) pe care le vom utiliza pe

parcursul acestui capitol.

1.1 Setul de date

Setul de date pe care îl vom folosi pentru a exemplifica elementele grafice introduse în acest capitol este creat prin manipularea unor date în format brut descărcate de pe platforma www.gapminder.org/data/. Mai precis vom descărca în format `csv` datele referitoare la durata medie de viață (sau speranța de viață) (*Health -> Life expectancy* sau http://gapm.io/ilex), produsul intern brut pe cap de locuitor ajustat la rata inflației (*Economy -> Incomes & growth -> Income* sau http://gapm.io/dgdppc) și respectiv populația totală (*Population -> Population* sau http://gapm.io/dpop) a peste 180 de state pe perioada 1800 - 2018. De asemenea, pentru a adăuga informații referitoare la locația geografică a țărilor din setul de date, respectiv continentul pe care se află fiecare țară, vom descărca fișierul excel de pe pagina https://www.gapminder.org/fw/four-regions/. Codul pentru crearea setului de date `gapminder_all` este ilustrat mai jos (acesta seamănă cu setul de date utilizat în capitolul de manipulare a datelor):

```
# citim seturile de date
life_exp_gap = read_csv("dataIn/life_expectancy_years.csv")
tot_pop_gap = read_csv("dataIn/population_total.csv")
gdp_gap = read_csv("dataIn/income_per_person_gdppercapita_ppp_inflation_adjusted.csv")
countries_gap = read_excel("dataIn/Data Geographies - v1 - by Gapminder.xlsx", sheet = 2)

# transformam seturile de date
life_exp_gap_pivot = life_exp_gap %>%
  pivot_longer(cols = -country, names_to = "year", values_to = "lifeExp")

tot_pop_gap_pivot =tot_pop_gap %>%
  pivot_longer(cols = -country, names_to = "year", values_to = "pop")

gdp_gap_pivot = gdp_gap %>%
  pivot_longer(cols = -country, names_to = "year", values_to = "gdpPerCap")

countries_gap_keep = countries_gap %>%
  select(geo, name, four_regions,
         six_regions, "World bank region",
         "World bank, 4 income groups 2017") %>%
  rename(wb_region = "World bank region",
         wb_income = "World bank, 4 income groups 2017",
         country = "name")

# unim seturile de date
gapminder_all <- life_exp_gap_pivot %>%
  left_join(tot_pop_gap_pivot) %>%
  left_join(gdp_gap_pivot) %>%
  left_join(countries_gap_keep) %>%
  # capitalizarea primei litere, i.e. transformarea: asia -> Asia
  mutate(four_regions = str_to_title(four_regions))
```

Pentru a avea o imagine de ansamblu asupra setului de date construit vom folosi funcția `glimpse`:

```
glimpse(gapminder_all)
Observations: 40,953
Variables: 10
$ country      <chr> "Afghanistan", "Afghanistan", "Afghanistan", "Afghanis...
$ year         <chr> "1800", "1801", "1802", "1803", "1804", "1805", "1806"...
```

```
$ lifeExp      <dbl> 28.2, 28.2, 28.2, 28.2, 28.2, 28.2, 28.1, 28.1, 28.1, ...
$ pop          <dbl> 3280000, 3280000, 3280000, 3280000, 3280000, 3280000, ...
$ gdpPerCap    <dbl> 603, 603, 603, 603, 603, 603, 603, 603, 603, 604,...
$ geo          <chr> "afg", "afg", "afg", "afg", "afg", "afg", "afg", "afg"...
$ four_regions <chr> "Asia", "Asia", "Asia", "Asia", "Asia", "Asia", "Asia"...
$ six_regions  <chr> "south_asia", "south_asia", "south_asia", "south_asia"...
$ wb_region    <chr> "South Asia", "South Asia", "South Asia", "South Asia"...
$ wb_income    <chr> "Low income", "Low income", "Low income", "Low income"...
```

În multe din exemplele din această secțiune vom folosi doar datele corespunzătoare anului 2018 prin urmare vom crea și acest set de date și-l vom numi `gapminder_2018`:

```
# setul de date pentru anul 2018
gapminder_2018 = gapminder_all%>%
  filter(year == 2018) %>%
  select(-geo) %>%
  rename(continent = four_regions)
```

Putem sumariza variabilele din setul de date `gapminder_2018` pentru a înțelege mai bine tipul lor și a avea mai multe informații referitoare la statisticile elementare (media, mediana, valoarea minimă și respectiv maximă, etc.) asociate acestora:

```
summary(gapminder_2018)
   country              year             lifeExp           pop
 Length:187         Length:187         Min.   :51.10   Min.   :5.320e+04
 Class :character   Class :character   1st Qu.:67.10   1st Qu.:2.460e+06
 Mode  :character   Mode  :character   Median :74.05   Median :9.420e+06
                                       Mean   :72.66   Mean   :4.062e+07
                                       3rd Qu.:78.03   3rd Qu.:3.005e+07
                                       Max.   :84.20   Max.   :1.420e+09
                                       NA's   :3
   gdpPerCap       continent         six_regions         wb_region
 Min.   :   629   Length:187         Length:187         Length:187
 1st Qu.:  3605   Class :character   Class :character   Class :character
 Median : 11700   Mode  :character   Mode  :character   Mode  :character
 Mean   : 17984
 3rd Qu.: 25200
 Max.   :121000

  wb_income
 Length:187
 Class :character
 Mode  :character
```

Observăm că pentru variabila `lifeExp` avem trei observații care nu prezintă valori, prin urmare putem exclude acele observații din setul de date:

```
gapminder_2018 = gapminder_2018 %>%
  filter(!is.na(lifeExp))
```

1.2 Elemente de bază

Graficele (figurile) realizate prin intermediul pachetului `ggplot2` au la bază o serie de elementele constructive, printe care putem enumera:

- setul de date ce urmează să fie prezentat grafic (**data**) - acesta *trebuie* să fie sub forma unui `data.frame` (adus la un format *tidy*)
- obiectele geometrice (**geom**) ce urmează să apară pe grafic (puncte, linii, etc.)
- o serie de corespondențe (**mappings**) de la variabilele din setul de date la aspectul (estetica - **aesthetics**) obiectelor geometrice
- transformări statistice (**statistical transformations**) folosite pentru a calcula valorile datelor ce sunt utilizate în grafic
- ajustări ale pozițiilor (**position adjustments**) obiectelor geometrice pe grafic
- scalări și scale (**scales**) pentru fiecare corespondență estetică folosită
- sisteme de coordonate (**coordinate systems**)
- fațete (**facets**) sau grupuri de date folosite în figuri diferite

Aceste componente constructive ale unui grafic sunt organizate în straturi (*layers*), unde fiecare strat conține un singur obiect grafic, transformare statistică sau ajustare de poziție și astfel putem vizualiza o figură/un grafic ca pe o mulțime de straturi de imagini suprapuse, fiecare prezentând un anumit aspect al datelor.

```
Error in knitr::include_graphics("images/layers2.png"): Cannot find the file(s): "images/layers2.png"
```

Primul pas în crearea unui grafic folosind pachetul `ggplot2` este de a construi un obiect *ggplot*. Acest obiect, creat prin intermediul funcției `ggplot()`, inițializează suprafața de desen (*canvas*) fără a desena nimic pe ea. Prin apelarea acestei funcții se specifică, în general, atât setul de date (în format `data.frame`) care va fi folosit în ilustrația grafică precum și proprietățile estetice (ce țin de aspect) care vor fi aplicate (vor corespunde) obiectelor geometrice folosite ulterior. Cu alte cuvinte, vom spune funcției `ggplot` care sunt datele pe care urmărim să le ilustrăm grafic și cum fiecare varaibilă din acest set de date va fi folosită (e.g. ca și coordonate x, y sau ca variabile de colorare - *colour* ori mărime - *size*, etc.). Este important de specificat că setul de date **trebuie** să fie sub forma unui `data.frame` adus la un format *tidy*. Codul generic este:

```
# il atribuim unui obiect
obiect = ggplot(data = <DATAFRAME>, aes(x = <COLOANA_1>, y = <COLOANA_2>))

# nu il atribuim unui obiect
ggplot(data = <DATAFRAME>, aes(x = <COLOANA_1>, y = <COLOANA_2>))
```

Sumarizând avem următorii pași ilustrați în figura de mai jos:

1. Apelăm funcția `ggplot()` care ne va crea o suprafață de desen goală
2. Specificăm corespondențele estetice (*aesthetic mappings*) dintre variabile setului de date și elemente ce țin de aspect. În acest caz, pentru setul de date `gapminder_2018`, între variabilele `gdpPerCap` și `lifeExp` și axele x și y.
3. Adăugăm obiecte geometrice care vor apărea pe grafic. În cazul nostru vom folosi stratul `geom_point()` pentru a adăuga puncte ca elemente grafice care să reprezinte datele.

```
# crearea suprafetei de desenare
ggplot(gapminder_2018)

# variabilele de interes sunt afisate
ggplot(gapminder_2018, aes(x = gdpPerCap, y = lifeExp))

# datele desenate
ggplot(gapminder_2018, aes(x = gdpPerCap, y = lifeExp)) +
  geom_point()
```

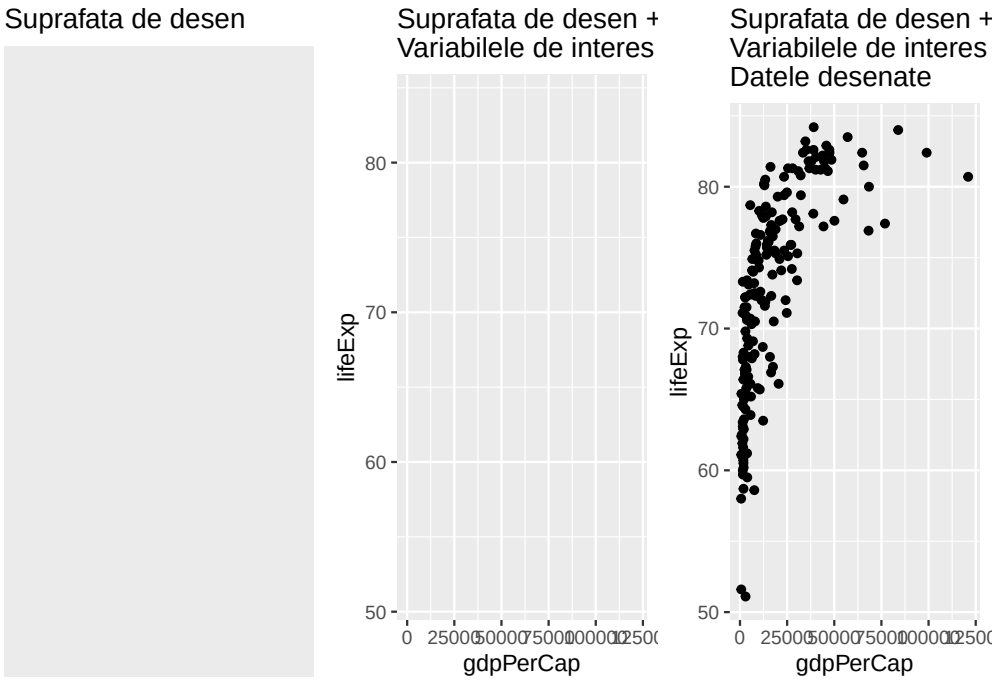


Fig. 3: Ilustrare a temelor predefinite

Este de remarcat faptul că atunci când am folosit stratul *geom* am utilizat operatorul `+`. De fiecare dată când vom adăuga un nou strat grafic la figura noastră o vom face prin intermediul operatorului `+`. Funcția `geom_point()` este doar o scriere prescurtată de trasare a graficului, în realitate se apelează funcția `layer()` care în fapt construiește un nou strat în care se specifică una din cele cinci componente principale `mapping`, `data`, `geom`, `stat` și `position`:

```
ggplot(gapminder_2018, aes(x = gdpPerCap, y = lifeExp)) +
  layer(
    mapping = NULL,
    data = NULL,
    geom = "point",
    stat = "identity",
    position = "identity"
    )
```

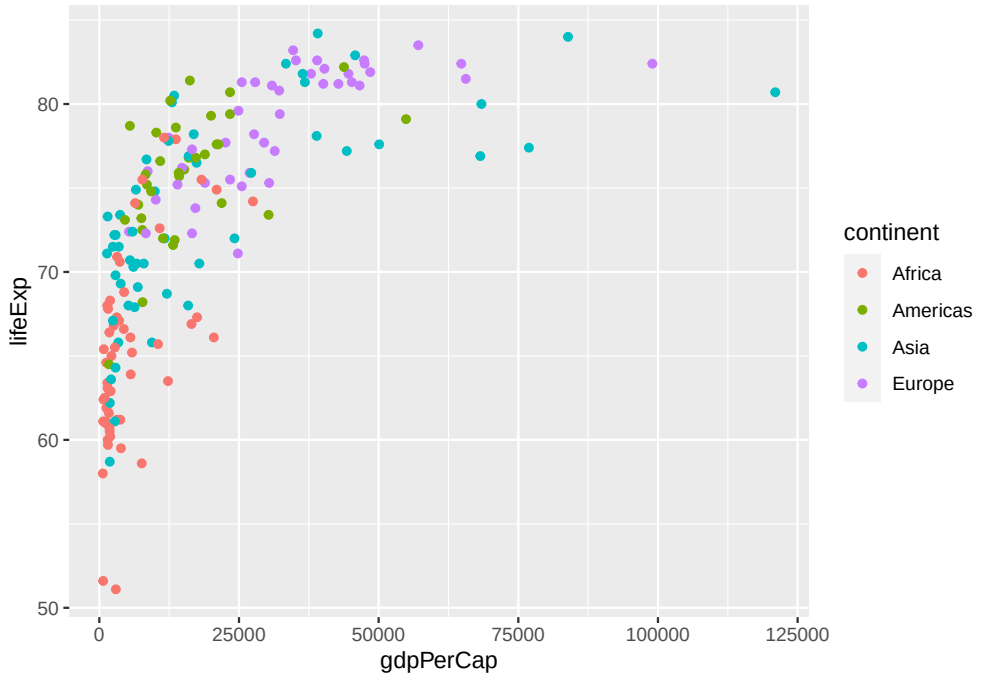
Structura generică a unui grafic realizat în `ggplot2` este:

```
ggplot(data = [DATA]) +         # element obligatoriu
 [GEOM_FUNCTION](
    mapping = aes([MAPPINGS]),  # element obligatoriu
    stat = [STAT],              # element optional
    position = [POSITION]       # element optional
 ) +
 [COORDINATE_FUNCTION] +        # element optional
 [FACET_FUNCTION] +             # element optional
 [SCALE_FUNCTION] +             # element optional
 [THEME_FUNCTION]               # element optional
```

1.3 Corespondențe estetice (*aesthetic mappings*)

Corespondențele estetice, de aspect, (aesthetic mappings) preiau proprietăți ale datelor și le folosesc pentru a influența o serie de caracteristici vizuale ale figurii, precum poziția, culoarea, mărimea, forma sau transparența. Astfel fiecare caracteristică vizuală poate codifica un aspect al datelor și, prin urmare, poate fi utilizată pentru a transmite noi informații. Toate elementele estetice se specifică folosind funcția `aes()` atât în interiorul funcției `ggplot` unde dobândesc un caracter global cât și în interiorul fiecărui strat definit de un obiect geometric (*geom*). De exemplu, în figura de mai jos, pentru setul de date `gapminder_2018`, culoarea (definită prin estetica `colour =`) descrie (corespunde la) apartenența la continent, poziția x arată produsul intern brut al statelor (`gdpPerCap`) iar poziția y arată speranța medie de viață (`lifeExp`).

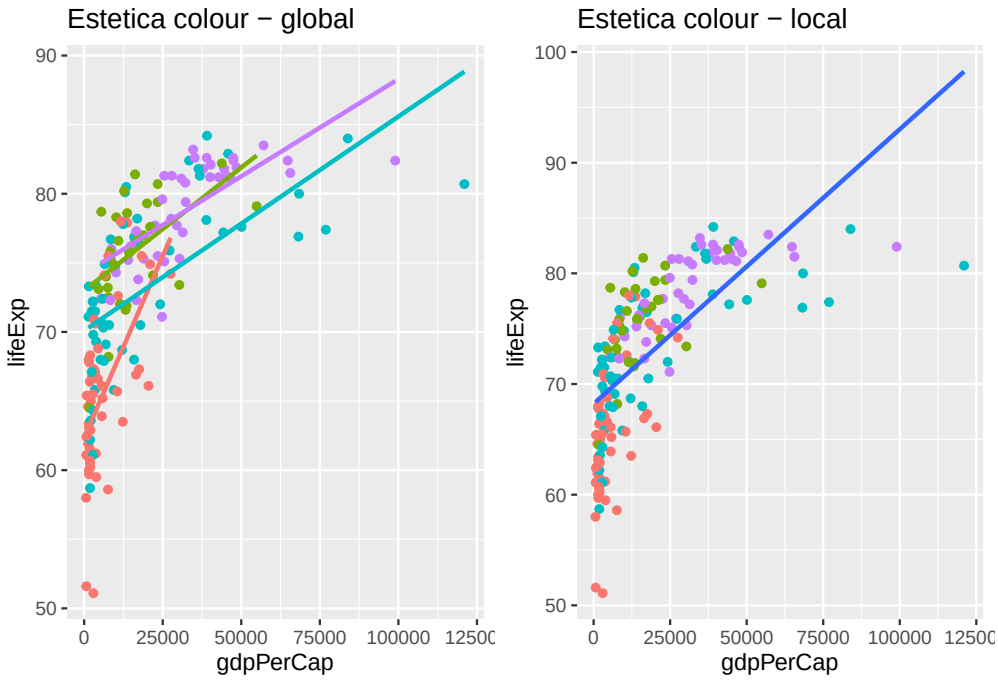
```
# caracter global pentru colour
ggplot(gapminder_2018, aes(x = gdpPerCap, y = lifeExp, colour = continent)) +
  geom_point()

# sau local
ggplot(gapminder_2018, aes(x = gdpPerCap, y = lifeExp)) +
  geom_point(aes(colour = continent))
```



Diferența dintre caracterul global și cel local al elementelor estetice (atunci când sunt aplicate funcției `ggplot` sau a straturilor individuale) se poate observa în figura de mai jos, unde adăugăm la grafic dreptele de regresie corespunzătoare țărilor de pe fiecare continent. Astfel, în cazul global, dreptele de regresie sunt asociate pentru fiecare subset de date (determinate de variabila `continent` prin intermediul esteticii de culoare) pe când atunci când elementul estetic este specificat în cadrul stratului geometric local (`geom_point()`)) se asociază dreapta de regresie asociată întregului set de date.

```
# global
ggplot(gapminder_2018, aes(x = gdpPerCap, y = lifeExp, colour = continent)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE)

# local
ggplot(gapminder_2018, aes(x = gdpPerCap, y = lifeExp)) +
```

```
  geom_point(aes(colour = continent)) +
  geom_smooth(method = "lm", se = FALSE)
```



Elementele estetice ce pot fi utilizate pentru trasarea unei figuri depind de tipul de obiect/strat geometric (*geom*) folosit. Pentru mai multe informații se poate consulta documentația funcțiilor `geom` la secțiunea *Aesthetics* (e.g. `?geom.point`) unde elementele obligatorii apar îngroșate iar cele opționale apar normal.

Printre cele mai uzuale elemente estitice ale unui grafic enumerăm:

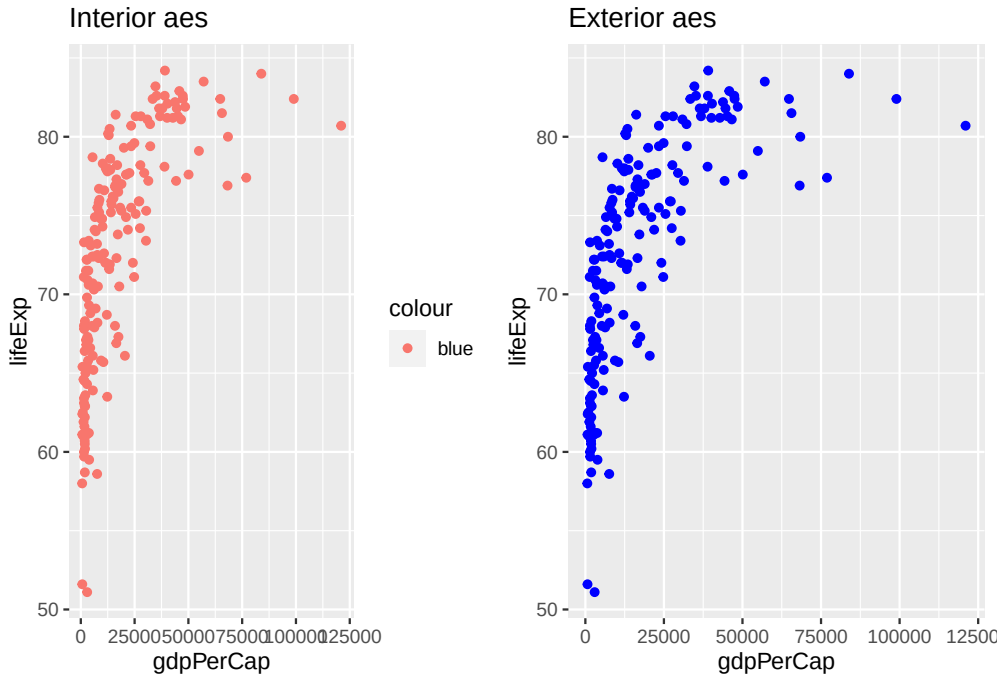
Element estetic (aesthetic)	Descriere
`x`	Poziția pe axa absciselor (x-axis)
`y`	Poziția pe axa ordonatelor (y-axis)
`shape`	Forma punctelor
`color/colour`	Culoarea marginală a elementelor
`fill`	Culoarea de umplere, interioară, a elementelor
`size`	Mărimea
`alpha`	Tranparență (1 - opac, 0 - transparent)
`linetype`	Tipul liniei (e.g. solidă, punctată, etc.)

O prezentare generală și mai amănunțită a acestora se regăsește în vigneta pachetului `ggplot2` intitulată `Aesthetic specifications` care poate fi apelată prin

```
vignette("ggplot2-specs")
```

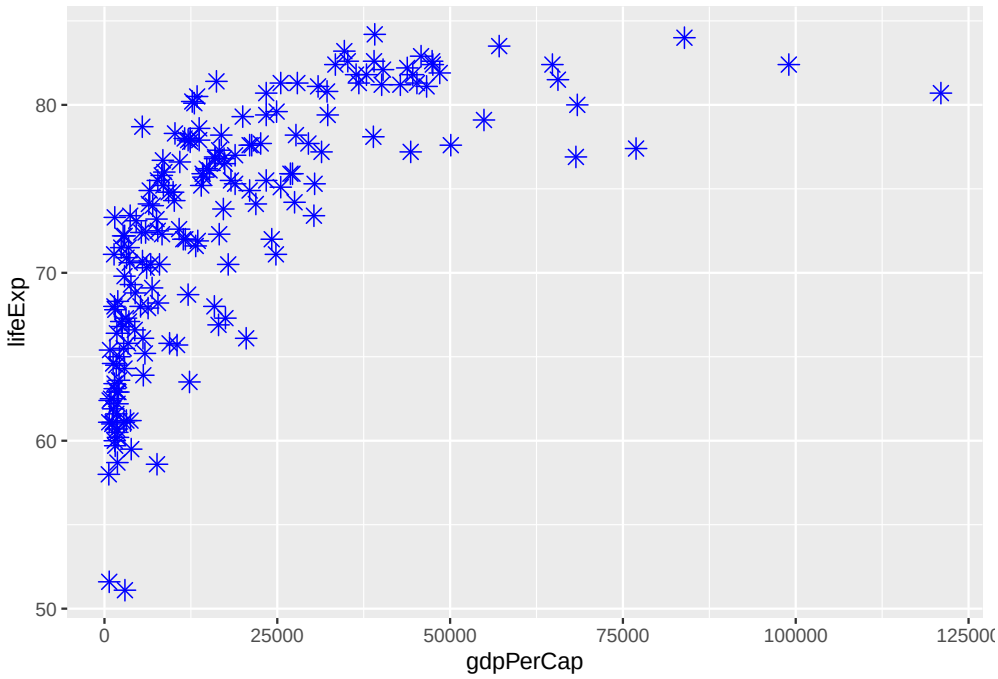
Trebuie remarcat faptul că prin utilizarea funcției `aes()`, canalul vizual va fi determinat să se bazeze pe datele specificate în argument. De exemplu, atunci când folosim comanda `aes(colour = "blue")` nu va face ca obiectul geometric (în cazul de mai sus punctele) să aibă culoarea albastră, ci va face maparea ca și cum am avea un singur tip numit "albastru". În cazul în care dorim să aplicăm valoarea elementului estetic întregului obiect geometric, de exemplu să schimbăm culoarea punctelor în albastru, atunci trebuie să specificăm elementul estetic în afara funcției `aes()`:

```
# interior
ggplot(gapminder_2018, aes(x = gdpPerCap, y = lifeExp)) +
  geom_point(aes(colour = "blue"))
# exterior
ggplot(gapminder_2018, aes(x = gdpPerCap, y = lifeExp)) +
  geom_point(colour = "blue")
```



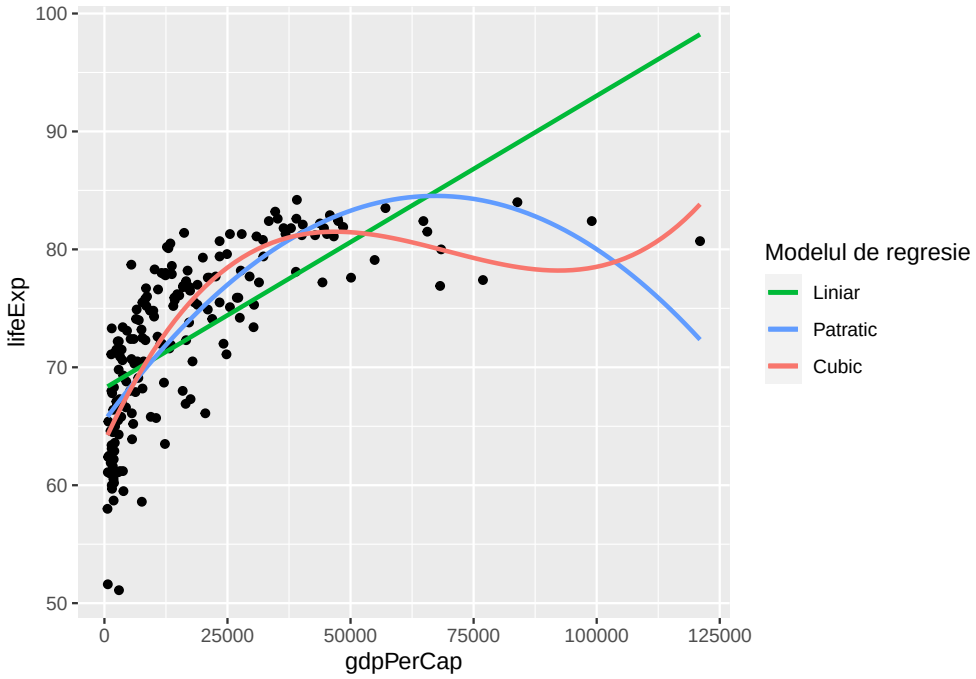
Putem face acest lucru pentru fiecare caracteristică estetică, incluzând `colour`, `fill`, `shape` sau `size`. În exemplul de mai jos schimbăm culoarea, mărimea și forma punctelor:

```
ggplot(gapminder_2018, aes(x = gdpPerCap, y = lifeExp)) +
  geom_point(colour = "blue", size = 3, shape = 8)
```



Atribuirea unei valori fixate (unei constante) unei estetici în interiorul funcției `aes()` poate fi utilă în special atunci când dorim să ilustrăm mai multe straturi cu diverși parametrii și în care dorim să numim acești parametrii. Spre exemplu să presupunem că dorim să ajustăm mai multe modele de regresie la setul de date `gapminder_2018` și să scoatem în evidență curbele generate.

```
ggplot(gapminder_2018, aes(x = gdpPerCap, y = lifeExp)) +
  geom_point() +
  geom_smooth(method = "lm",
              se = FALSE,
              aes(color = "Liniar") ) +
  geom_smooth(method = "lm",
              formula = y ~ poly(x, 2),
              se = FALSE,
              aes(color = "Patratic") ) +
  geom_smooth(method = "lm",
              formula = y ~ poly(x, 3),
              se = FALSE,
              aes(color = "Cubic") )
```

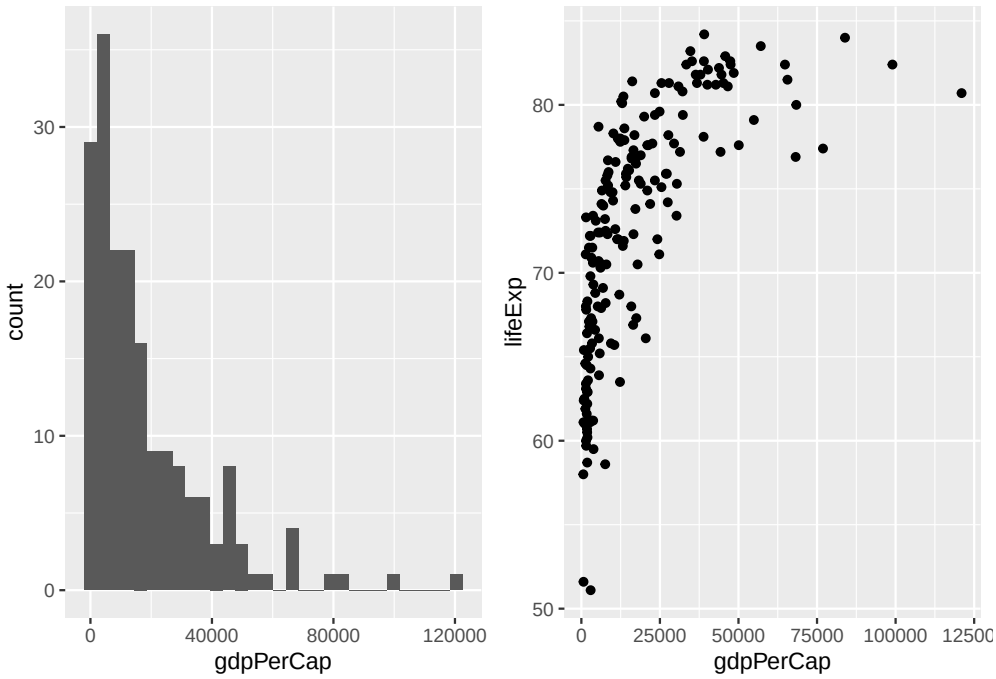


1.4 Obiecte geometrice (*geometric objects*)

Obiectele geometrice (*geometric objects*), sau `geom` pe scurt, sunt elementele fundamentale ale unui grafic creat în `ggplot2`, prin intermediul lor se efectuează reprezentarea a datelor, controlând tipul de grafic dorit. Neincluderea unui strat `geom` în desen conduce la o figură goală (a se vedea mai sus). Majoritatea obiectelor geometrice sunt asociate unui grafic a cărui denumire este prestabilită, astfel `geom_bar` corespunde unei diagrame cu bare (*barplot*), `geom_line` corespunde unei diagrame liniare (*linegraph*), `geom_histogram` corespunde unei histograme, `geom_boxplot` corespunde unei diagrame de tip boxplot (*cutie cu mustăți*) ș.a.m.d. O excepție de la această regulă este dată de diagrama de împrăștiere (*scatterplot*) pentru care se folosește `geom_point`.

Fiecare funcție `geom` are propriile elemente estetice și argumente necesare pentru a ajusta modul în care este creat graficul. De exemplu, funcția `geom_histogram` necesită doar elementul estetic `x` pe când dacă dorim să trasăm o diagramă de împrăștiere avem nevoie atât de estetica `x` cât și de `y`. În contextul setului de date `gapminder_2018`, variabila `gdpPerCap` ne arată produsul intern brut pe cap de locuitor pentru fiecare țară iar variabila `lifeExp` prezintă speranța medie de viață din țara respectivă. Pentru a vedea cum este distribuit produsul intern brut pentru anul 2018 vom trasa o histogramă a acestuia iar pentru a investiga relația dintre variabilele `gdpPerCap` și `lifeExp` vom ilustra o diagramă de împrăștiere:

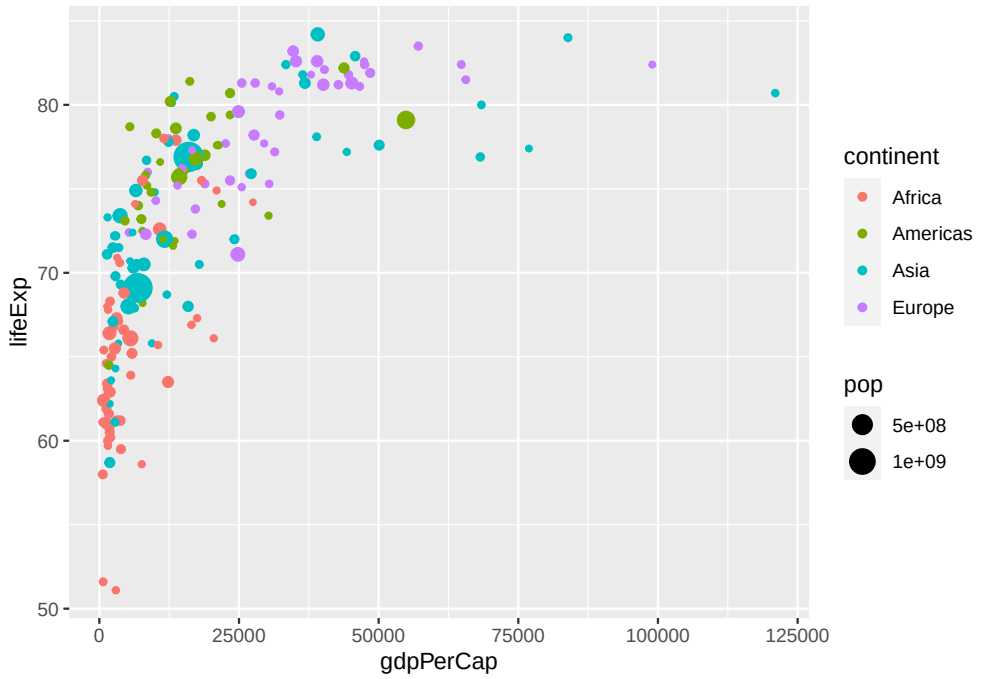
```
# histograma
ggplot(gapminder_2018, aes(x = gdpPerCap)) +
  geom_histogram()

# diagrama de imprastiere - scatterplot
ggplot(gapminder_2018, aes(x = gdpPerCap, y = lifeExp)) +
  geom_point()
```



Pentru fiecare funcție `geom` există elemente estetice obligatorii și opționale. Majoritatea obiectelor grafice necesită elementele estetice `x` și `y` dar sunt și excepții precum `geom_bar` sau `geom_histogram`. De exemplu, în cazul funcției `geom_point`, elementele estetice obligatorii sunt `x` și `y` iar cele opționale pot include: `alpha` (transparență), `colour`, `fill`, `size`, `shape` sau `stroke`. Trebuie menționat că elementele estetice pot să difere de la `geom` la `geom` în funcție de specificul vizual al fiecărui obiect grafic, de exemplu putem utiliza `shape` ca estetică pentru `geom_point` dar nu o putem utiliza și pentru `geom_line` precum putem utiliza `linetype` pentru `geom_line` dar nu și pentru `geom_point`.

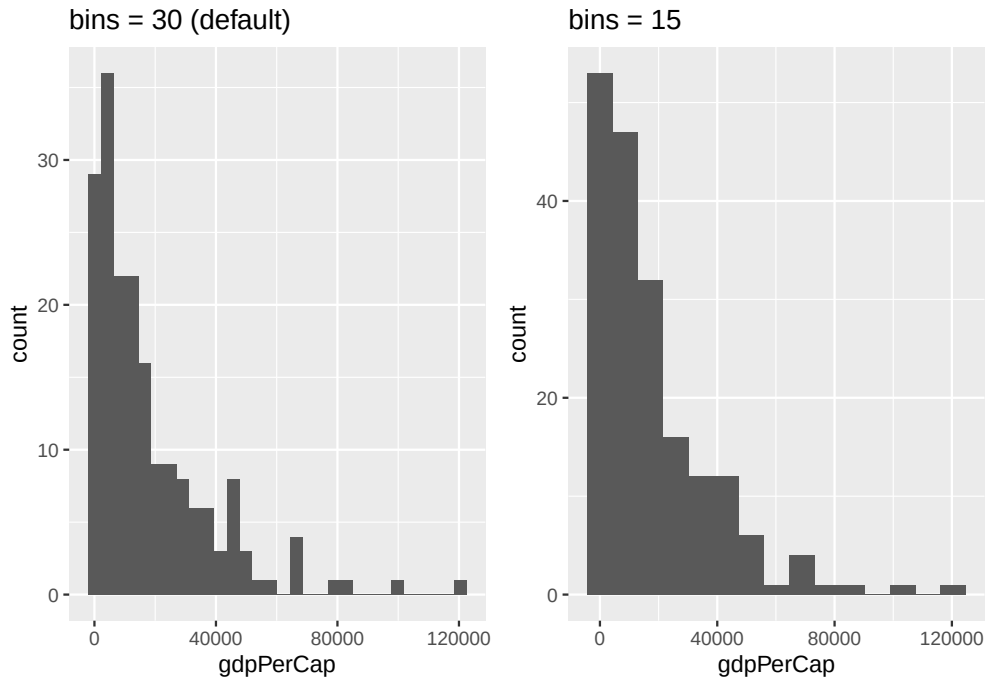
În contextul setului de date `gapminder_2018` vom utiliza atât estetica `colour` pentru a diferenția țările în funcție de `continent` cât și estetica `size` pentru a scoate în evidență țările cu o populație (`pop`) mai mare:

```
ggplot(gapminder_2018, aes(x = gdpPerCap, y = lifeExp, colour = continent, size = pop)) +
  geom_point()
```



Pe lângă elementele estetice, funcțiile `geom` admit și o serie de argumente specifice (care se regăsesc la secțiunea *arguments* în documentația acestora, e.g. `?geom_point`), de exemplu dacă dorim să modificăm numărul de subintervale (*bins*) folosit pentru trasarea unei histograme atunci putem utiliza argumentul `bins =`:

```
# histograma
ggplot(gapminder_2018, aes(x = gdpPerCap)) +
  geom_histogram(bins = 15)
```



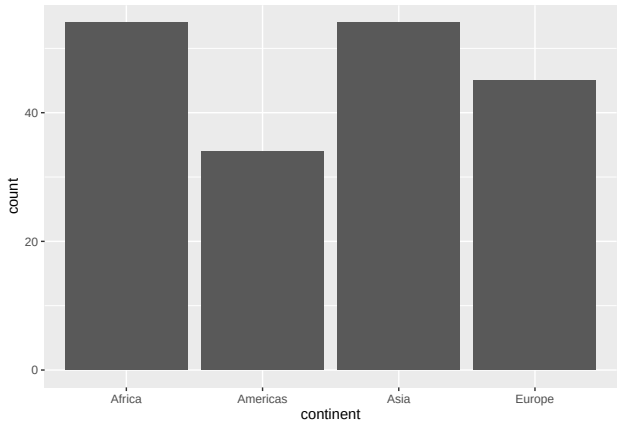
Următorul tabel prezintă o serie de funcții `geom` împreună cu elementele estetice obligatorii și o parte din argumentele specifice ale acestora.

Funcția	Elemente estetice	Argumente opționale	Descriere
`geom_point`	x, y		Trasează o diagramă de împrăștiere
`geom_line`	x, y	arrow, na.rm	Trasează o diagramă liniară
`geom_segment`	x, y, xend, yend	arrow, na.rm	Trasează un segment de dreaptă
`geom_path`	x, y	na.rm	Trasează o linie poligonală
`geom_polygon`	x, y		Trasează o linie poligonală închisă (umplută)
`geom_rect`	xmin, xmax, ymin, ymaxl		Trasează un dreptunghi
`geom_histogram`	x	bins, binwidth	Trasează o histogramă
`geom_density`	x		Trasează o diagramă de densitate estimată prin nuclee
`geom_dotplot`	x	bins, binwidth	Trasează o diagramă cu puncte similară diagramei cu bare
`geom_freqpoly`	x	binwidth	Trasează o linie poligonală de frecvență
`geom_bar`	x sau x, y	width	Trasează o diagramă cu bare
`geom_abline`	intercept, slope		Trasează drepte în care sunt specificate pantele și ordonatele la origine
`geom_hline`	yintercept		Trasează drepte de referință orizontale
`geom_vline`	xintercept		Trasează drepte de referință verticale
`geom_smooth`	x, y	method, se, span	Adaugă o curbă de regresie
`geom_text`	x, y, label		Adaugă etichete text la puncte
`geom_bin2d`	x, y	bins	Ilustrează o estimare a densității bivariate prin dreptunghiuri
`geom_hex`	x, y	bins	Ilustrează o estimare a densității bivariate prin hexagoane (faguri)

Vom prezenta mai jos o parte dintre graficele generate de aceste funcții în contextul seturilor de date `gapminder_all` și `gapminder_2018`. Vom face separarea graficelor după numărul (una sau două) și tipul (discrete sau continue) variabilelor analizate:

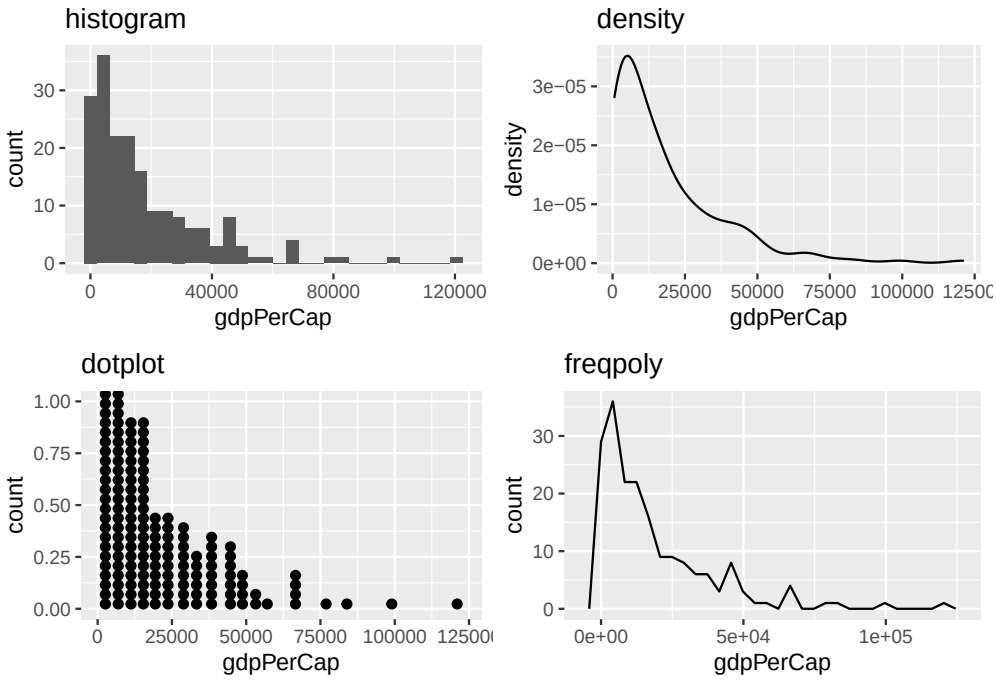
- o variabilă discretă - `geom_bar()`

```
ggplot(gapminder_2018, aes(x = continent)) +
  geom_bar()
```



- o variabilă continuă - `geom_histogram`, `geom_density`, `geom_dotplot` și `geom_freqpoly`

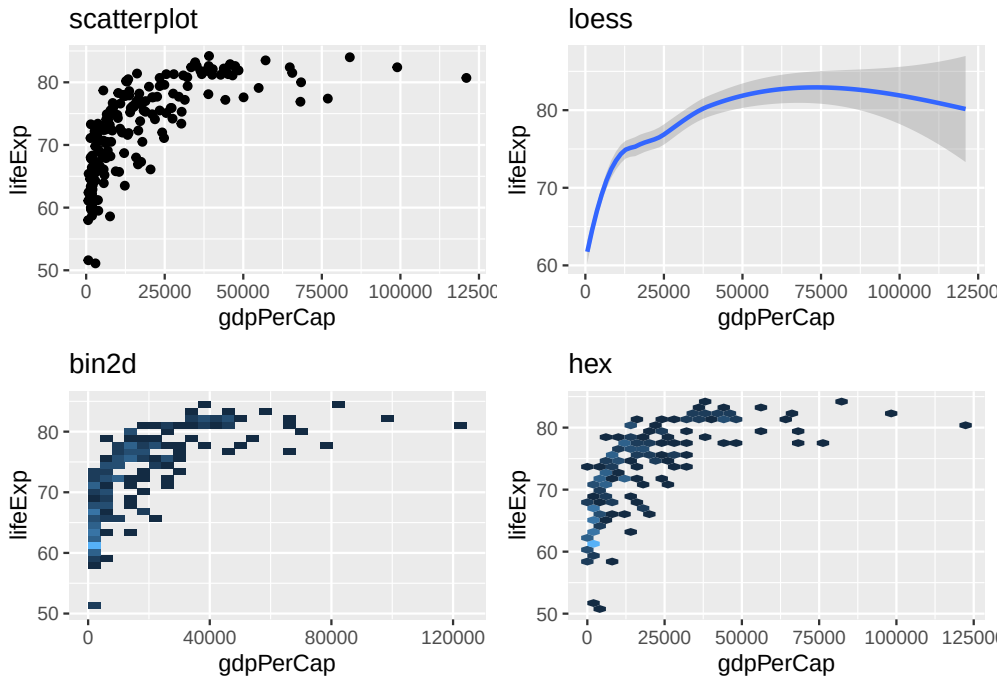
```
# histograma
ggplot(gapminder_2018, aes(x = gdpPerCap)) +
  geom_histogram()

# estimarea densitatii
ggplot(gapminder_2018, aes(x = gdpPerCap)) +
  geom_density()

# dotplot
ggplot(gapminder_2018, aes(x = gdpPerCap)) +
  geom_dotplot()

# diagrama de frecvente
ggplot(gapminder_2018, aes(x = gdpPerCap)) +
  geom_freqpoly()
```



- două variabile ambele continue - `geom_point`, `geom_line`, `geom_smooth`, `geom_bin2d`, `geom_hex`, etc.

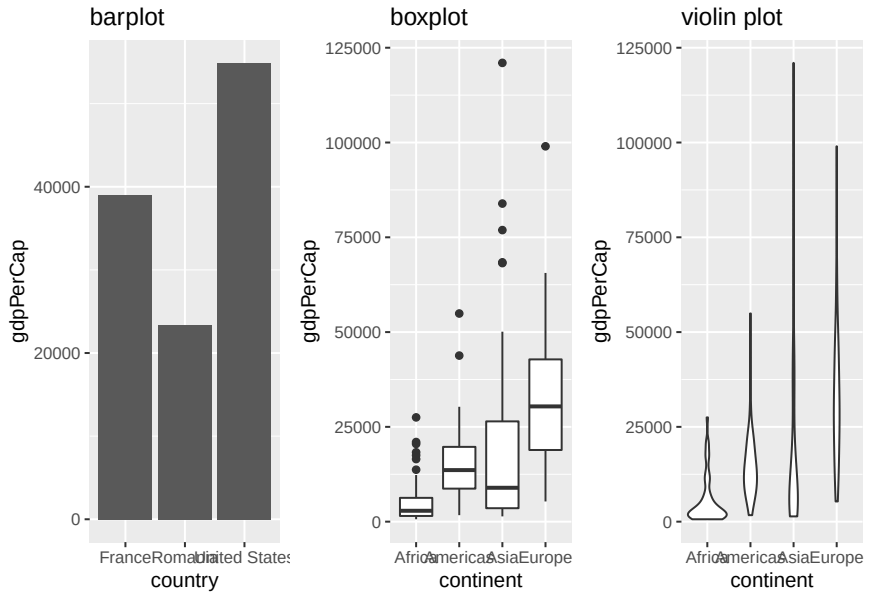
```
# Diagrama de imprastiere
ggplot(gapminder_2018, aes(x = gdpPerCap, y = lifeExp)) +
  geom_point()

# Functia de regresie - default loess
ggplot(gapminder_2018, aes(x = gdpPerCap, y = lifeExp)) +
  geom_smooth()

# Densitate bivariata
ggplot(gapminder_2018, aes(x = gdpPerCap, y = lifeExp)) +
  geom_bind2d()

# Densitate bivariata
ggplot(gapminder_2018, aes(x = gdpPerCap, y = lifeExp)) +
```

```
  geom_hex()
```



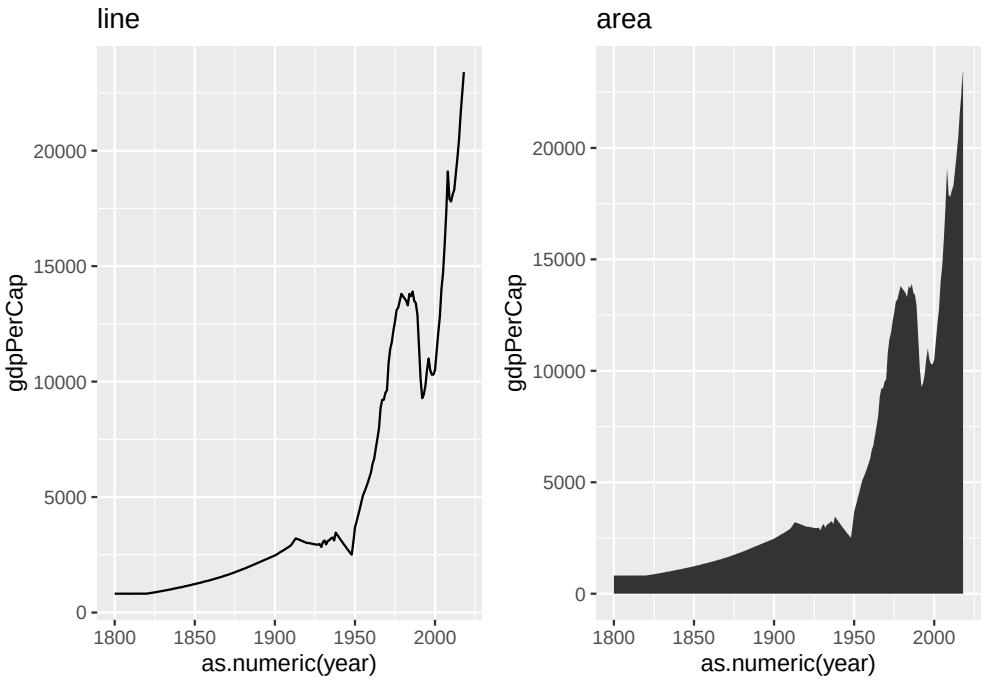
- o variabilă continuă și una discretă - `geom_bar(stat = "identity")` (sau `geom_col`), `geom_boxplot`, `geom_violin`, etc.

```
# datele
gap1 = gapminder_2018 %>%
  filter(country %in% c("Romania", "United States", "France"))

# barplot
ggplot(gap1, aes(x = country, y = gdpPerCap)) +
  geom_bar(stat = "identity")
# sau
ggplot(gap1, aes(x = country, y = gdpPerCap)) +
  geom_col()

# boxplot
ggplot(gapminder_2018, aes(x = continent, y = gdpPerCap)) +
  geom_boxplot()
```



- o variabilă continuă și alta de tip temporal: `geom_line`, `geom_area`

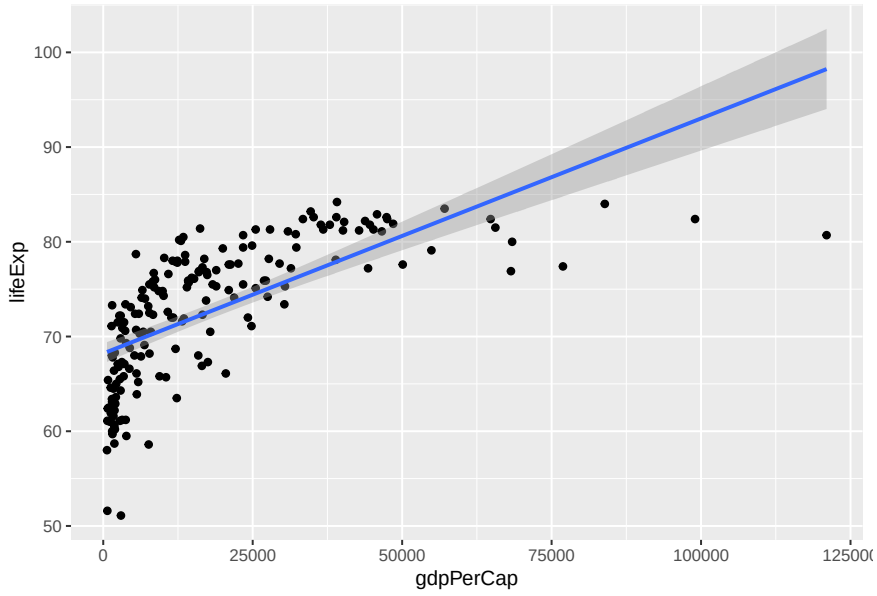
```
# diagrama liniara
ggplot(gapminder_all %>% filter(country == "Romania"),
       aes(x = as.numeric(year), y = gdpPerCap)) +
  geom_line()

# area plot
ggplot(gapminder_all %>% filter(country == "Romania"),
       aes(x = as.numeric(year), y = gdpPerCap)) +
  geom_line()
```



Ceea ce face pachetul `ggplot2` cu adevărat puternic este că acesta permite construcția de figuri complexe

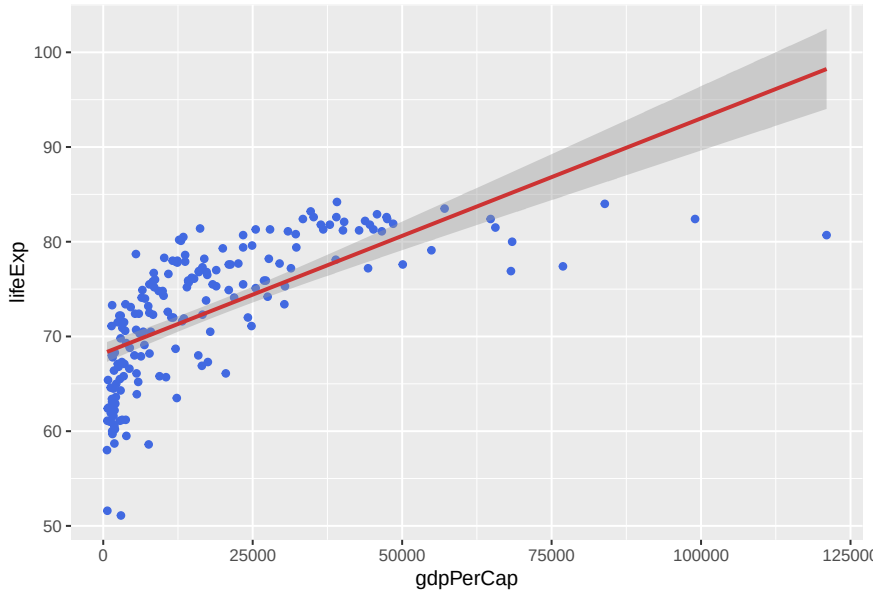
prin adăugarea mai multor obiecte grafice la aceeași figură, suprapunând straturile determinate de ele. Să considerăm setul de date `gapminder_2018` și să adăugăm la diagrama de împrăștiere a variabilelor `gdpPerCap` și `lifeExp` relația liniară determinată de dreapta de regresie:

```
ggplot(gapminder_2018, aes(x = gdpPerCap, y = lifeExp)) +
  geom_point() +
  geom_smooth(method = "lm")
```



Trebuie remarcat faptul că putem utiliza elemente estetice diferite pentru fiecare obiect geometric folosit, de exemplu în figura anterioară putem schimba culoarea punctelor și a dreptei de regresie.

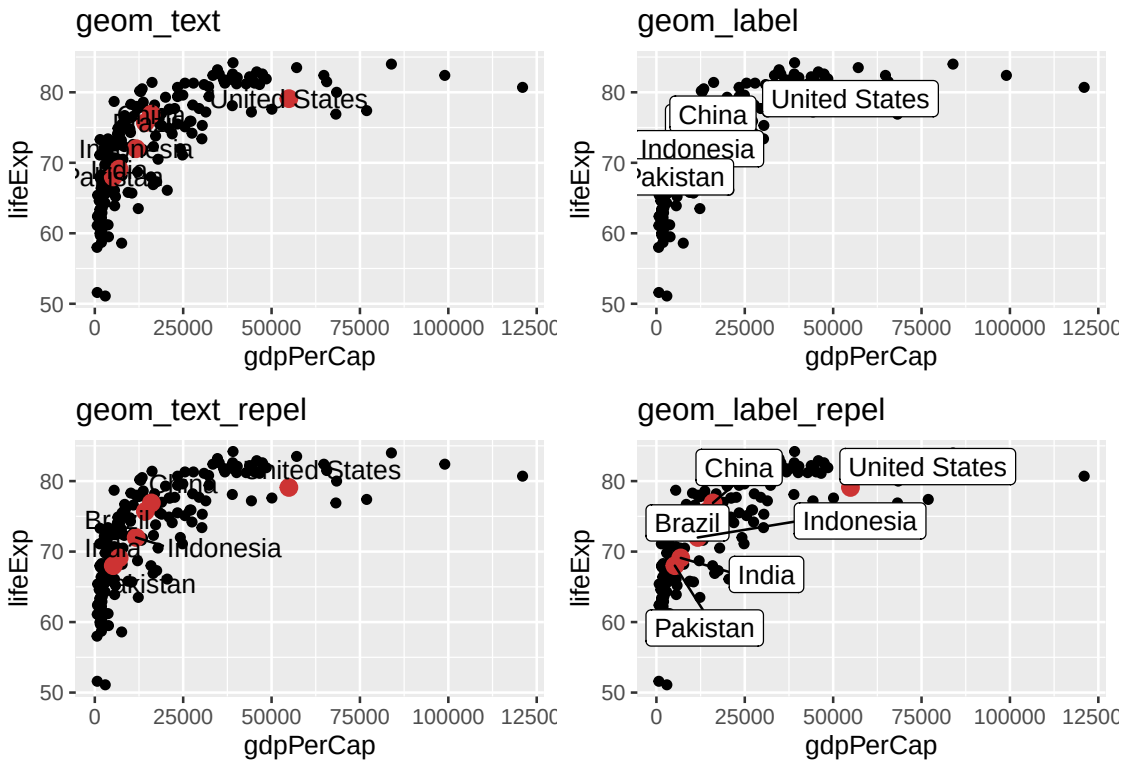
```
ggplot(gapminder_2018, aes(x = gdpPerCap, y = lifeExp)) +
  geom_point(color = "royalblue") +
  geom_smooth(method = "lm", color = "brown3")
```



De asemenea, este posibil să atribuim fiecărui strat `geom` propriul set de date astfel încât să putem utiliza

mai multe seturi de date pentru a scoate în evidență caracteristicile de interes ale graficului (variabilelor). În contextul diagramei de împrăștiere dintre produsul intern brut pe cap de locuitor și speranța medie de viață pentru setul de date `gapminder_2018` vrem să adăugăm numele (etichetele - label) țărilor care au mai mult de 200 milioane de locuitori și să colorăm punctele în roșu. Pentru început, vom construi setul de date cu țările corespunzătoare și apoi vom adăuga informația folosind `geom_text`. Etichetarea elementelor grafice se poate face cu `geom_text` sau `geom_label` dar atunci când punctele sunt foarte aglomerate se pot folosi și funcții alternative precum cele din pachetul `ggrepel` (`geom_text_repel` sau `geom_label_repel`). Pentru a colora punctele vom folosi estetica `colour` într-un nou strat `geom_point`. :

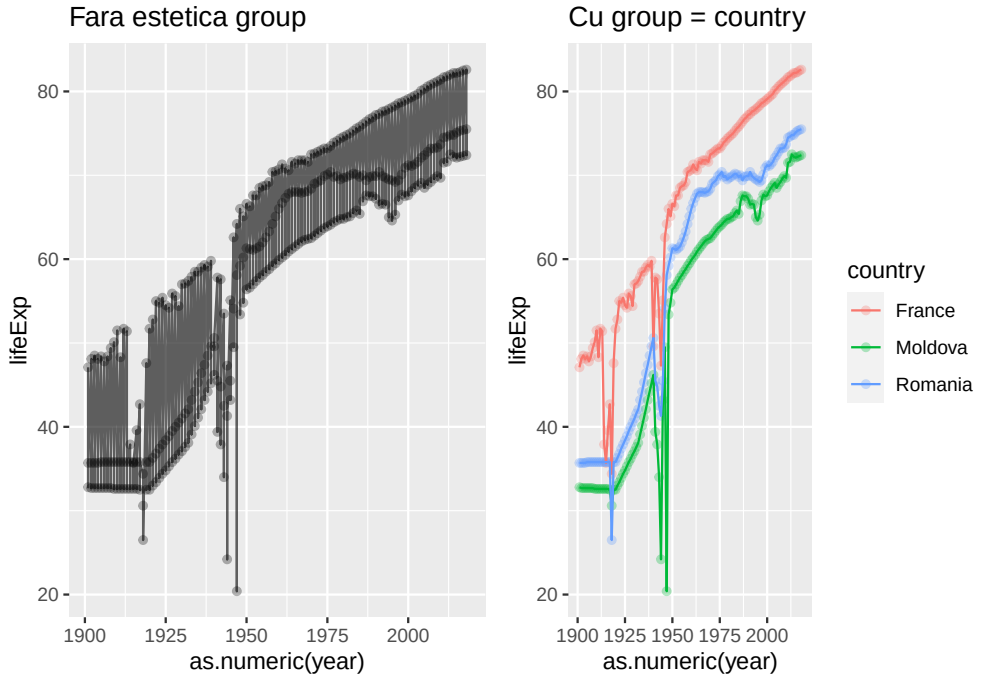
```
tari_mari = gapminder_2018 %>%
  filter(pop > 2e8)

ggplot(gapminder_2018, aes(x = gdpPerCap, y = lifeExp)) +
  geom_point() +
  geom_point(data = tari_mari,
             aes(x = gdpPerCap, y = lifeExp),
             colour = "brown3", size = 3) +
  geom_text(data = tari_mari, aes(label = country))
```



Sunt și situații în care dorim să separăm datele în grupuri dar vrem să le redăm în același mod, de exemplu atunci când avem un studiu longitudinal (studiu care presupune observații repetate asupra acelorași variabile) cu mai mulți subiecți. Acest tip de situație, întâlnit în cazul în care un `geom` afișează observații multiple printr-un singur obiect geometric (e.g. `geom_boxplot`), necesită separarea pe grupe a observațiilor, fapt realizat prin intermediul elementului estetic `group`. Pentru o mai bună înțelegere vom ilustra acest concept folosind setul de date `gapminder_all`. Să presupunem că dorim să afișăm evoluția duratei medii de viață după anul 1900 pentru o submulțime de țări (*Romania*, *Moldova* și *France*).

```
gapminder_all %>%
  filter(as.numeric(year) > 1900) %>%
  filter(country %in% c("Romania", "Moldova", "France")) %>%
```

```
ggplot(aes(x = year, y = lifeExp)) +
  geom_point()+
  geom_line()

gapminder_all %>%
  filter(as.numeric(year) > 1900) %>%
  filter(country %in% c("Romania", "Moldova", "France")) %>%
ggplot(aes(x = year, y = lifeExp, group = country, colour = country)) +
  geom_point()+
  geom_line()
```



1.5 Transformări statistice (*statistical transformations*)

Pachetul `ggplot2` folosește o serie de transformări statistice, sau `stat`, pentru a aduce datele la formatul dorit pentru trasare. Forma generală a acestora este `stat_[nume statistică]`. De obicei aceste transformări sumarizează datele într-un anume fel, de exemplu în cazul unui *boxplot* pentru valorile de pe `y` se calculează mediana (Q_2), prima și a treia cuartila (Q_1 și Q_3) și distanța dintre acestea (IQR), iar în cazul unei curbe de regresie (`geom_smooth`) se calculează valoarea medie a lui `y` condiționată la `x`. Un alt exemplu constă în trasarea unei diagrame cu bare (*barplot*) în care axa `y` este definită ca fiind numărul de elemente din fiecare clasă (`count`) a lui `x`, număr care nu aparține setului de date inițial ci este obținut prin aplicarea unei transformări statitice `stat_count`. Fiecare obiect geometric `geom` admite o transformare statistică predefinită și vice versa dar acestea pot fi schimbate. De exemplu, transformarea statistică predefinită pentru `geom_bar` este `stat_count`

```
# argumentele functiei geom_bar
args(geom_bar)
function (mapping = NULL, data = NULL, stat = "count", position = "stack",
    ..., width = NULL, binwidth = NULL, na.rm = FALSE, orientation = NA,
    show.legend = NA, inherit.aes = TRUE)
NULL
```

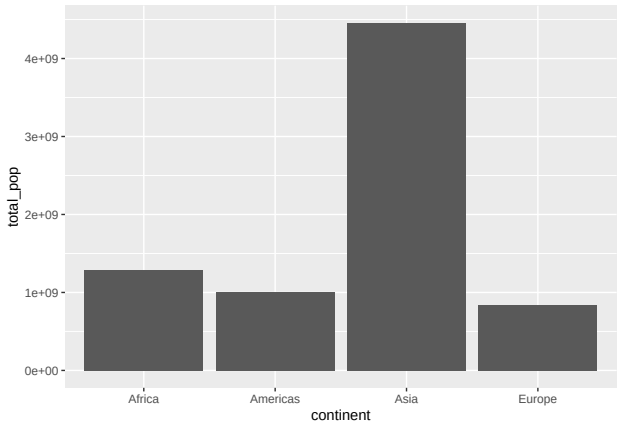
```
# argumentele functiei stat_count
args(stat_count)
function (mapping = NULL, data = NULL, geom = "bar", position = "stack",
    ..., width = NULL, na.rm = FALSE, orientation = NA, show.legend = NA,
    inherit.aes = TRUE)
NULL
```

Următorul tabel prezintă o serie de transformări statistice și obiectele geometrice pentru care acestea reprezintă un comportament implicit:

Transformare statistică	Obiect geometric
`stat_identity`	`geom_point`
`stat_count()`	`geom_bar()`
`stat_bin()`	`geom_freqpoly()`, `geom_histogram()`
`stat_boxplot()`	`geom_boxplot()`
`stat_smooth()`	`geom_smooth()`
`stat_bin2d()`	`geom_bin2d()`
`stat_binhex()`	`geom_hex()`
`stat_bindot()`	`geom_dotplot()`
`stat_contour()`	`geom_contour()`

De exemplu transformarea *identitate* (*identity*) va lăsa datele așa cum sunt și folosită în `geom_bar` conduce la construcția unei diagrame cu bare în care pe axa `y` se află valorile variabilei `y` fără a mai efectua transformarea (în acest caz este nevoie atât de estetica `x` cât și de `y`). În figura următoare afișăm populația totală a țărilor de pe fiecare dintre cele patru continente considerate în setul de date `gapminder_2018`:

```
gapminder_2018 %>%
  group_by(continent) %>%
  summarise(n = n(),
            max_pop = max(pop),
            total_pop = sum(pop)) %>%
  ggplot(aes(x = continent, y = total_pop)) +
  geom_bar(stat = "identity")
```

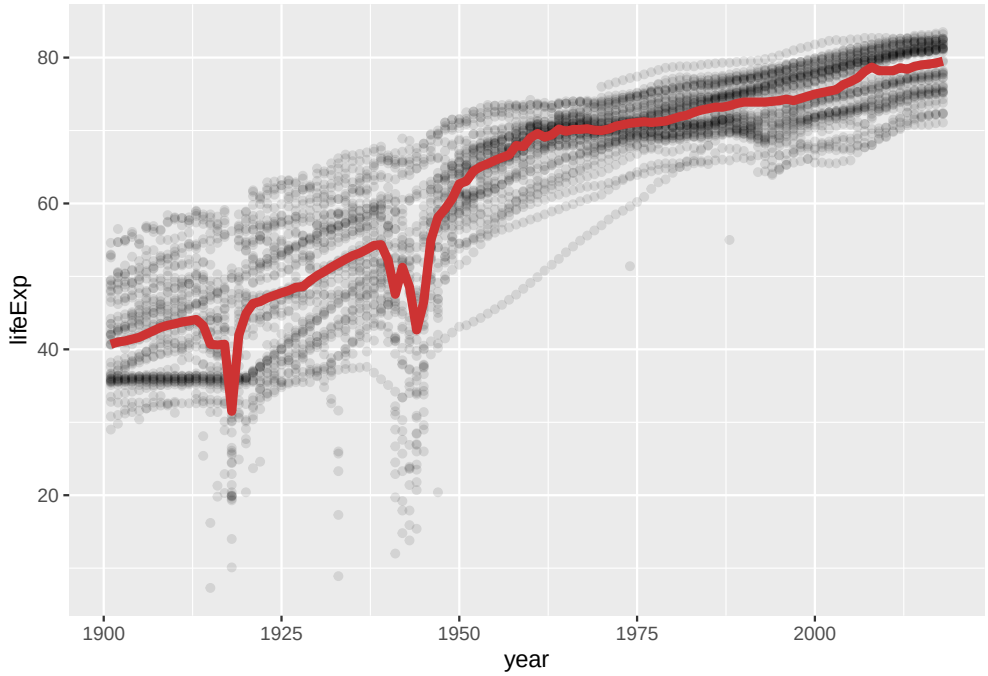


Pe lângă transformările implicite există și o serie de transformări care nu pot fi create prin aplicarea funcțiilor `geom_`. Tabelul de mai jos prezintă o parte dintre acestea (pentru mai multe informații se poate consulta documentația pachetului `ggplot2` la adresa https://ggplot2.tidyverse.org/reference/index.html#section-layer-stats):

Transformare statistică	Descriere
`stat_ecdf()`	calculează și ilustrează funcția de repartiție empirică
`stat_function()`	calculează valorile lui y aplicând o funcție valorilor lui x
`stat_summary()`	sumarizează valorile lui y la clase diferite în x
`stat_summary2d()`, `stat_summary_hex()`	este o versiune bidimensională a lui `stat_summary`
`stat_unique()`	elimină valorile duplicate
`stat_qq()`	efectuează calculele pentru un grafic cuantilă-cuantilă

Pentru a folosi aceste funcții putem sau să le apelăm în mod direct utilizând `stat_[nume]` și astfel suprascriind obiectul geometric sau să le apelăm în interiorul unui strat geometric. Pentru ilustrare vom folosi setul de date `gapminder_all` în care vom trasa o diagramă de împrăștiere unde pe axa x avem anii de după 1900 iar pe y speranța medie de viață a țărilor europene (pentru a evita supraaglomerarea punctelor - *ovreplotting* vom folosi estetic `alpha = 0.1`). Vom adăuga cu roșu la acestă diagramă valoarea mediană a duratei medii de viață a tuturor țărilor pentru fiecare an și vom uni aceste valori printr-o linie:

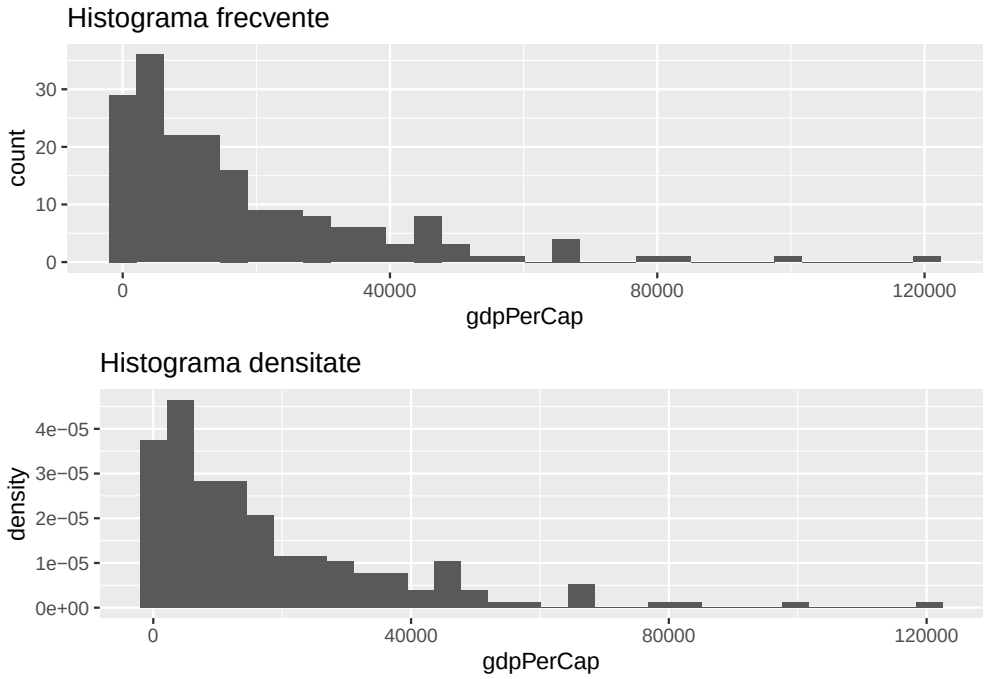
```
# transformam datele
gapminder_all %>%
  mutate(year = as.numeric(year),
         continent = four_regions) %>%
  filter(year > 1900,
         continent == "Europe") %>%
  ggplot(aes(x = year, y = lifeExp)) +
  # trasam diagrama de imprastiere
    geom_point(alpha = 0.1) +
  # unim punctele mediane
    geom_line(stat = "summary", fun = "median",
              colour = "brown3", linetype = 1, size = 2)
```



Trebuie avut în vedere că atunci când funcțiile `stat_` transformă datele, în realitate se construiește un alt `data.frame` la care pot să apară *noi* variabile (noi coloane) și prin urmare este posibil să utilizăm aceste

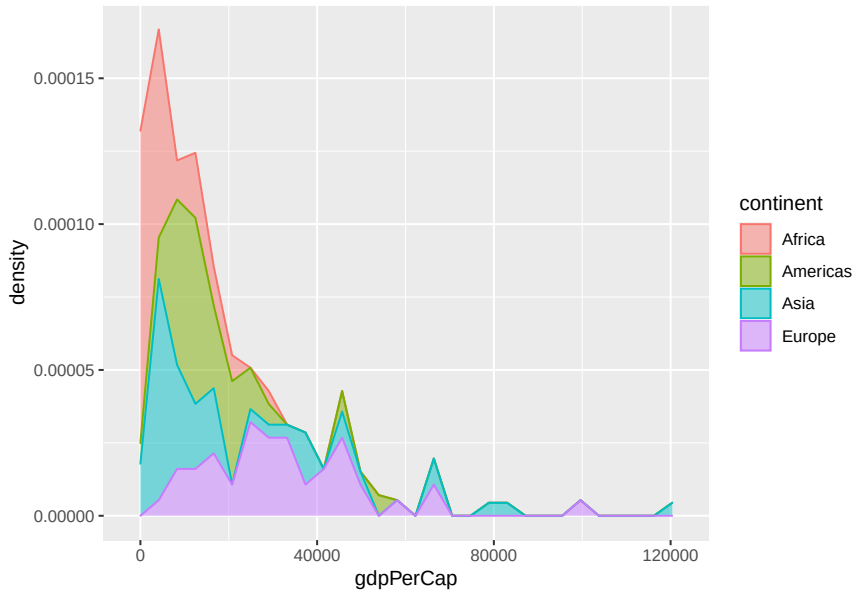
noi variabile în trasarea graficului. De exemplu, în cazul transformării `stat_bin` (statistica folosită pentru trasarea unei histograme) sunt construite următoarele variabile: `count` - numărul de observații din fiecare subinterval (*bin*), `density` - densitatea observațiilor din fiecare subinterval (procentul din total pe lungimea subintervalului), `x` - centrul subintervalului. Aceste variabile pot fi folosite în trasarea graficului folosind `..` în jurul numelui acestora (i.e. `..[nume variabilă]..`) pentru a evita o eventuală confuzie cu o variabilă din setul de date inițial care are aceeași denumire. Spre exemplu atunci când trasăm o histogramă observăm că în mod automat (implicit) înălțimea barelor este egală cu numărul de observații din subintervalul corespunzător (`count`) prin urmare vorbim de o histogramă de frecvențe și nu una de densitate. Pentru a obține-o pe cea din urmă trebuie să specificăm că înălțimea barelor este dată de variabila `density`. În contextul setului de date `gapminder_2018` aceasta se traduce prin

```
# histograma frecvente
ggplot(gapminder_2018, aes(x = gdpPerCap)) +
  geom_histogram()

# histograma densitate
ggplot(gapminder_2018, aes(x = gdpPerCap)) +
  geom_histogram(aes(y = ..density..))
```



Mai mult dacă dorim să comparăm modul de repartiție a produsului intern brut pe cap de locuitor pentru fiecare continent atunci este recomandată folosirea datelor standardizate:

```
ggplot(gapminder_2018, aes(x = gdpPerCap, colour = continent, fill = continent)) +
  geom_area(stat = "bin", aes(y = ..density..), alpha = 0.5)
```



1.6 Gestionarea scalelor (*scales*)

De fiecare dată când specificăm o corespondență de estetică/aspect (*aesthetic mapping*), funcția `ggplot` folosește o anumită *scală* (predefinită în funcție de tipul esteticii) pentru a determina intervalul de valori pe care datele trebuie să le verifice. O corespondență estetică (i.e. specificată cu `aes()`) ne spune doar că o variabilă ar trebui asociată unei caracteristici ce ține de aspect și nu cum trebuie să aibă loc această relație. De exemplu, atunci când folosim estetica `colour` (`aes(colour = x)`) nu specificăm nicăieri ce culori trebuie folosite pentru trasarea elementelor specifice nivelelor variabilei `x`. Aceeași observație este valabilă și pentru alte atribute estetice, i.e. `size`, `fill`, `shape`, etc. Pentru a specifica ce culoare, mărime, formă, etc. să fie utilizată trebuie modificată scala corespunzătoare. În pachetul `ggplot2`, scalele (*scales*) pot fi modificate prin intermediul funcțiilor de tipul

```
scale_[nume estetică]_[tip]
```

unde `nume estetică` reprezintă numele elemntului estetic (i.e. `x`, `y`, `colour`, `fill`, etc.) iar `tip` reprezintă tipul scalei folosite pentru elementul ales (i.e. `continuous`, `discrete`, `manual`, `gradient`, etc.). În funcție de estetică, tipul scalei poate să difere

Trebuie menționat că `ggplot` asociază în mod automat fiecărui element estetic folosit în trasarea graficului o scală. De exemplu dacă scriem

```
ggplot(gapminder_2018, aes(x = gdpPerCap, y = lifeExp)) +
  geom_point(aes(colour = continent))
```

în realitate are loc

```
ggplot(gapminder_2018, aes(x = gdpPerCap, y = lifeExp)) +
  geom_point(aes(colour = continent)) +
  scale_x_continuous() +
  scale_y_continuous() +
  scale_color_discrete()
```

dar pentru a evita scrierea manuală a fiecărei scale, `ggplot2` face lucrurile automat. Este foarte important de menționat că scalele pot modifica unul din cele două elemente de ghidaj (*guides*) ale unui grafic: axele și legendele. În `ggplot2` cele două elemente, chiar dacă vizual sunt diferite, ele sunt tratate în mod similar, legătura fiind evidențiată în tabelul următor:

Axă	Legendă	Numele parametrului
Nume (*Label*)	Titlu (*Title*)	`name`
Markeri (*Ticks & grid line*)	Intrările legendei (*Key*)	`breaks`
Etichete Markeri (*Tick label*)	Etichetele intrărilor legendei (*Key label*)	`labels`

În general, scalele disponibile în `ggplot2` pot fi împărțite în patru categorii, primele trei fiind cele mai utilizate:

- *scale continue de poziție* folosite pentru a face corespondența dintre datele de tip numeric sau de tip temporal (date/time) și poziția pe x sau y (i.e. `scale_x_continuous`, `scale_y_continuous`)
- *scale de culori* care permit corespondența dintre valorile continue sau discrete ale datelor și culori (i.e. `scale_color_discrete`, `scale_fill_continuous`, `scale_fill_gradient`)
- *scale manuale*, utilizate pentru a defini relația dintre valorile discrete ale datelor și mărimea (*size*), tipul liniei (*linetype*), forma (*shape*) sau culoarea (*colour*) dorită
- *scale identice*, folosite pentru a trasa variabilele fără a le scala (de exemplu atunci când avem deja un vector de culori)

O scală continuă va fi folosită cu precădere atunci când vorbim de date numerice (unde există un set continuu de numere), în timp ce o scală discretă va gestiona elemente precum culorile, forma punctelor sau tipul liniilor (deoarece există o listă mică de culori distincte).

În tabelul de mai jos sunt prezentate o parte dintre scalele cele mai utilizate (pentru mai multe detalii se poate consulta documentația pachetului ggplot2)

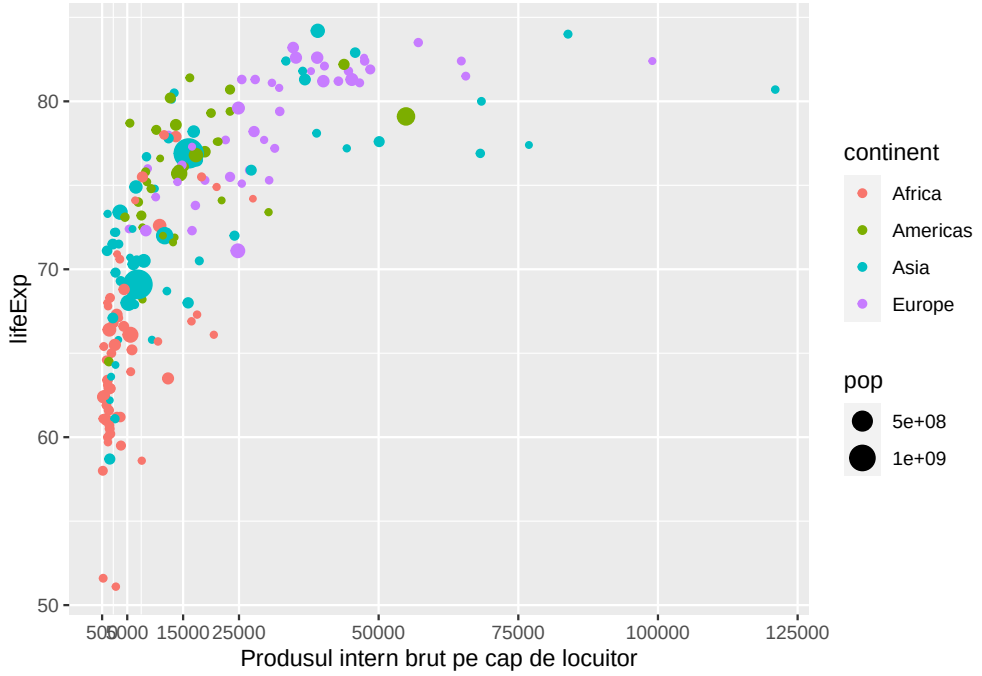
Scala	Tipul	Exemple
`scale_color_`	`identity`	`scale_fill_continuous`
`scale_fill_`	`manual`	`scale_color_discrete`
`scale_size_`	`continuous`	`scale_size_manual`
	`discrete`	`scale_size_discrete`
`scale_shape_`	`discrete`	`scale_shape_discrete`
`scale_linetype_`	`identity`	`scale_shape_manual`
	`manual`	`scale_linetype_discrete`
`scale_x_`	`continuous`	`scale_x_continuous`
`scale_y_`	`discrete`	`scale_y_discrete`
	`reverse`	`scale_x_log`
	`log`	`scale_y_reverse`
	`date`	`scale_x_date`
	`datetime`	`scale_y_datetime`

1.6.1 Scale de poziție

Pentru a ajusta scala axei x pentru o variabilă continuă vom folosi `scale_x_continuous`. Utilizarea funcțiilor de scală permite modificarea mai multor proprietăți ale elementului estetic, de exemplu în cazul scalei pe axa x putem modifica aspecte ale axei precum eticheta/titlul axei (*name*), poziția (*position*) sau distanța dintre markeri (*breaks* sau *minor_breaks*) și numele acestora (*labels*). Pentru alte elemente estetice în afara lui x și y, *axa* va fi de obicei legenda graficului și astfel modificarea scalei unui asemenea element permite modificarea elementelor componente ale legendei (nume, etichete, culori, etc.).

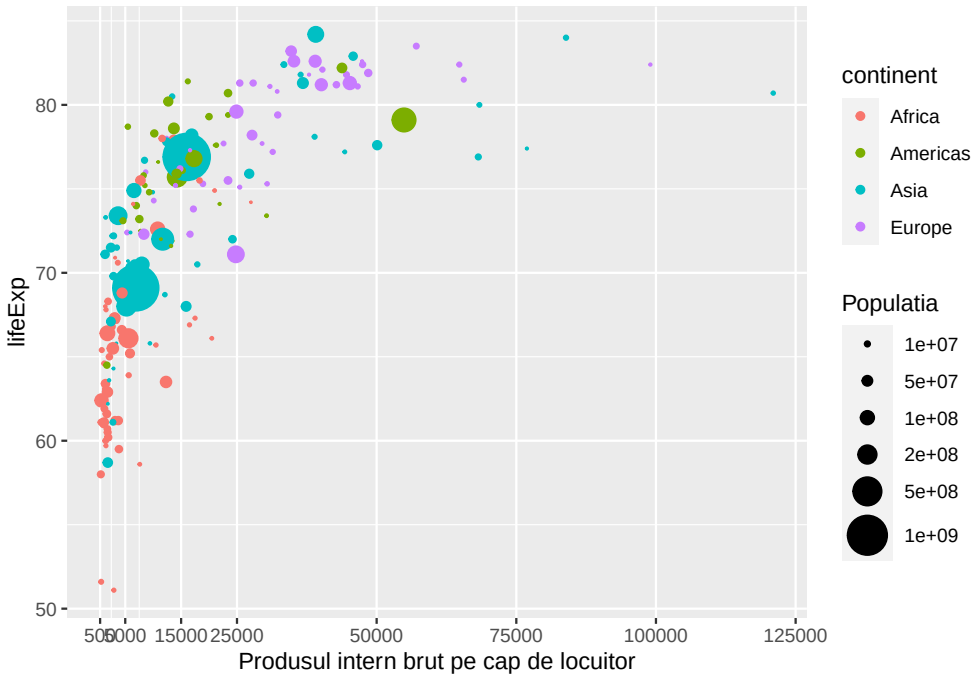
Ca exemplu, vom folosi setul de date `gapminder_2018` și folosind `scale_x_continuous` vom modifica numele axei x, poziția markerilor de separare (*breaks*) principali și secundari (*minor_breaks*):

```
ggplot(gapminder_2018, aes(x = gdpPerCap, y = lifeExp, colour = continent, size = pop)) +
  geom_point() +
  scale_x_continuous(name = "Produsul intern brut pe cap de locuitor",
                     breaks = c(500, 5000, 15000, 25000,
                                50000, 75000, 100000, 125000),
                     minor_breaks = c(500, 2500, 7500))
```



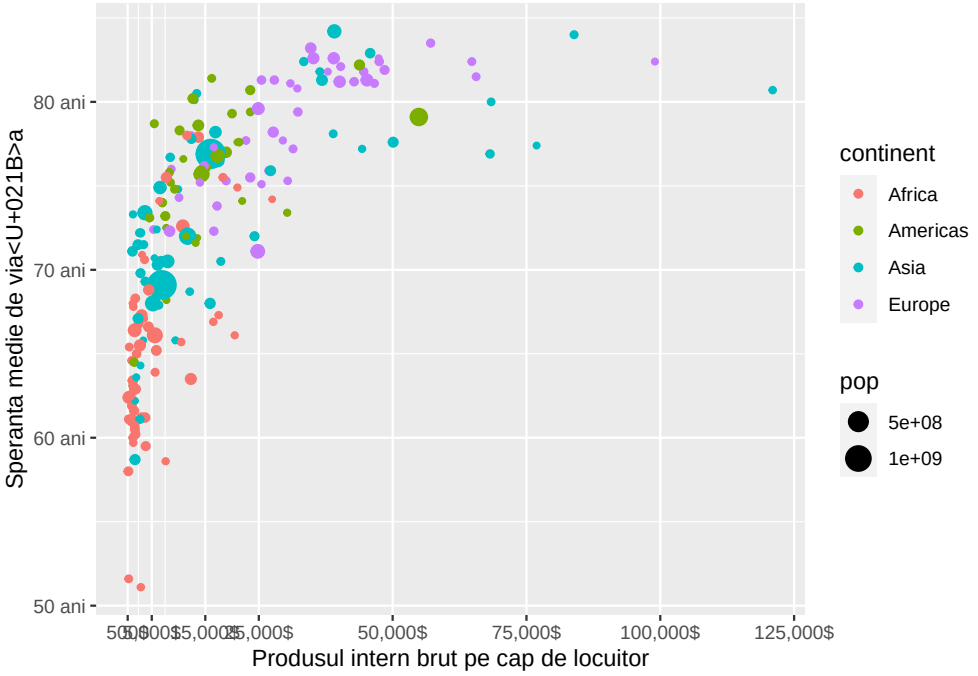
Este posibil să dorim de asemenea schimbarea numelui legendei ce corespunde variabilei `pop` (populației) din *pop* în *Populația*, să afișăm mărimile 1e7, 5e7, 1e8, 2e8, 5e8, 1e9 (*breaks*) și să schimbăm intervalul de valori (minim-maxim) pentru mărimea simbolului (*range*):

```
ggplot(gapminder_2018, aes(x = gdpPerCap, y = lifeExp, colour = continent, size = pop)) +
  geom_point() +
  scale_x_continuous(name = "Produsul intern brut pe cap de locuitor",
                     breaks = c(500, 5000, 15000, 25000,
                                50000, 75000, 100000, 125000),
                     minor_breaks = c(500, 2500, 7500)) +
  scale_size(name = "Populatia",
             breaks = c(1e7, 5e7, 1e8, 2e8, 5e8, 1e9),
             range = c(0.1, 10))
```



În plus, putem să adăugăm funcții ca argument al parametrilor `breaks` și `labels`, în primul caz funcția returnând un vector de valori numerice care să corespundă elementelor de marcaj iar în cazul etichetelor va primi ca date de intrare un vector numeric de elemente de marcaj și va returna un vector de etichete. De exemplu, pentru a adăuga simbolul `$` în dreptul valorilor produsului intern brut vom folosi funcții din pachetul `scales` precum `dollar_format()` iar pentru a modifica axa y vom crea propria funcție:

```
ggplot(gapminder_2018, aes(x = gdpPerCap, y = lifeExp, colour = continent, size = pop)) +
  geom_point() +
  scale_x_continuous(name = "Produsul intern brut pe cap de locuitor",
                     breaks = c(500, 5000, 15000, 25000,
                                50000, 75000, 100000, 125000),
                     minor_breaks = c(500, 2500, 7500),
                     labels = scales::dollar_format(prefix = "", suffix = "$")) +
  scale_y_continuous(name = "Speranta medie de viață",
                     labels = function(x){paste(x, "ani")})
```

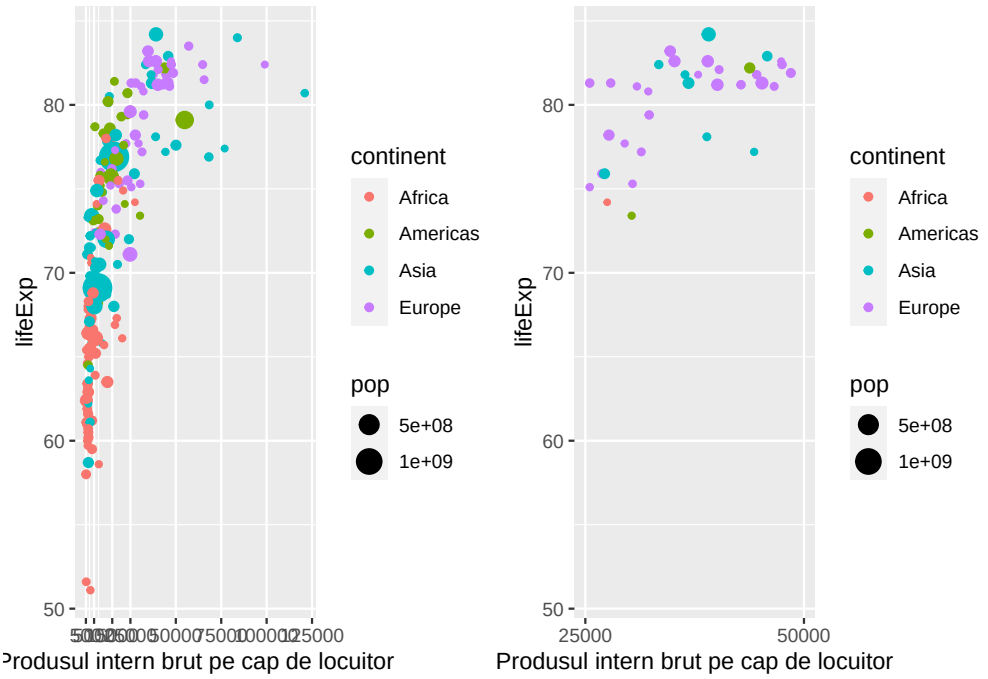


Fiecare funcție de scalare permite utilizarea unui număr diferit de parametrii, o parte dintre aceștia regăsindu-se în tabelul de mai jos (pentru a vizualiza toți parametrii disponibili se poate consulta documentația funcției, i.e. `?scale_x_continous`):

Parametru	Descriere
`name`	Eticheta (label) axei sau numele legendei
`breaks`	Vector de puncte de marcaj
`minor_breaks`	Vector de puncte de marcaj intermediare
`labels`	Etichete folosite pentru fiecare punct de marcaj
`limits`	Limitele intervalului de valori ale axei

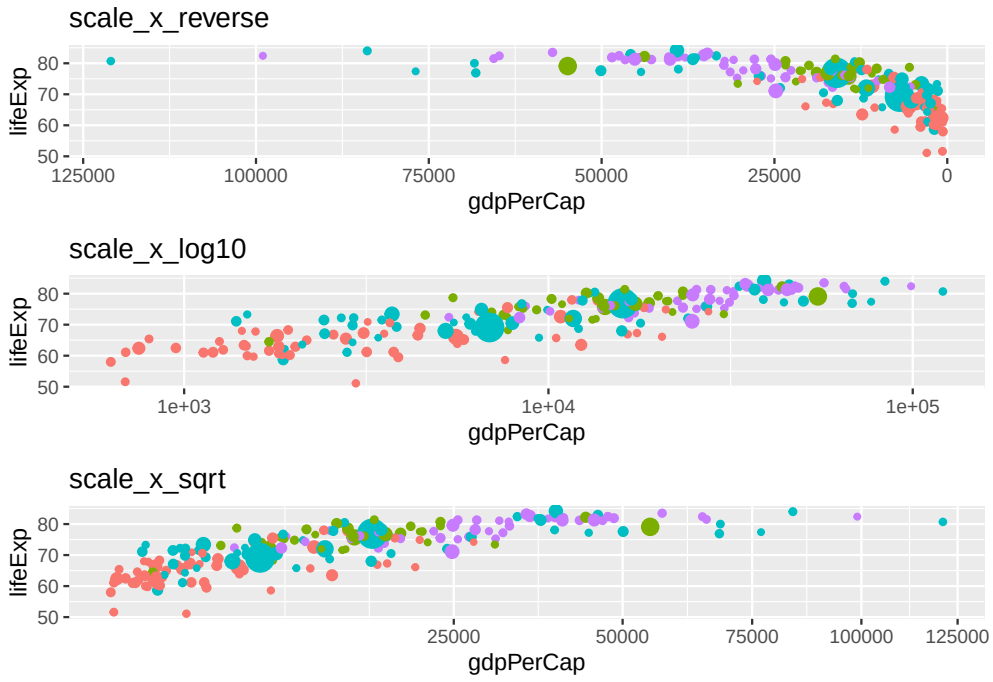
Modul de folosire a parametrilor `name`, `breaks`, `minor.breaks` și `labels` a fost ilustrat în exemplele anterioare. Parametrul `limits` este derivat din domeniul de valori al setului de date și este folosit cu precădere pentru a scoate în evidență o subregiune a graficului. Atunci când facem referire la scale continue parametrul `limits` primește ca argument de intrare un vector cu două valori (minim și maxim).

```
ggplot(gapminder_2018, aes(x = gdpPerCap, y = lifeExp, colour = continent, size = pop)) +
  geom_point() +
  scale_x_continuous(name = "Produsul intern brut pe cap de locuitor",
                     breaks = c(500, 5000, 15000, 25000,
                                50000, 75000, 100000, 125000),
                     minor_breaks = c(500, 2500, 7500),
                     limits = c(25000, 50000))
```



Deoarece modificarea limitelor axelor este o operație des întânlnită atunci când efectuăm analize exploratorii a datelor cu care lucrăm, pachetul `ggplot2` pune la dispoziție o serie de funcții predefinite precum `xlim()`, `ylim()` sau `lims()`.

Pachetul `ggplot2` pune la dispoziție și o serie de scale care permit modificarea axelor, de exemplu, prin inversarea valorilor (i.e. `scale_x_reverse`) sau prin transformarea acestora într-o scală logaritmică (folosind `scale_x_log10` sau `scale_x_continuous(trans = "log10")`) ori printr-o altă funcție (i.e. `scale_x_sqrt` sau `scale_x_continuous(trans = "sqrt")`). Pentru mai multe transformări se poate consulta documentația funcției `scale_x_continuous` la argumentul `trans`.

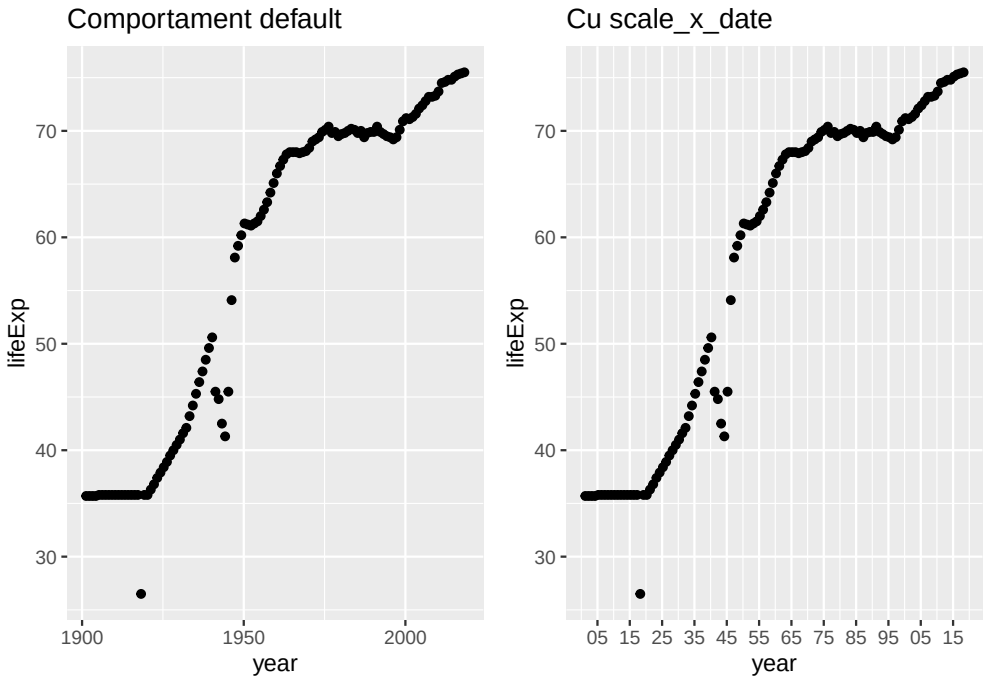


Atunci când avem de-a face cu variabile/date care sunt în format de date de timp (format `Date` sau `POSIXct`) este recomandată utilizarea funcțiilor predefinite `scale_x_date` și respectiv `scale_x_datetime` care dispun în plus de argumentele `date_breaks` și `date_labels`. Argumentul `date_breaks` permite poziționarea elementelor de marcaj în funcție de unitățile de timp (ani, luni, săptpmâni, etc.), astfel `date_breaks = 2 years` va pasa marcaje din doi în doi ani. Argumentul `date_labels` asigură ilustrarea etichetelor pentru marcaje în format `strptime()`.

Sir de caractere	Descriere
`%S`	secunde (00-59)
`%M`	minute (00-59)
`%l`	ore, format ceas 12 ore (1-12)
`%I`	ore, format ceas 12 ore (01-12)
`%p`	am/pm
`%H`	ore, format ceas 24 ore (00-23)
`%a`	zilele săptămânii, sub forma (Mon-Sun)
`%A`	zilele săptămânii, sub forma (Monday-Sunday)
`%e`	ziua din lună (1-31)
`%d`	ziua din lună (01-31)
`%m`	luna, format numeric (01-12)
`%b`	luna, format abreviat (Jan-Dec)
`%B`	luna, nume întreg (January-December)
`%y`	an, fără specificarea secolului (00-99)
`%Y`	an, cu specificarea secolului (0000-9999)

De exemplu dacă vrem să afișăm date precum 23/08/1944 vom folosi șirul de caractere `"%d/%m/%Y"`. Următorul cod ilustrează aceste opțiuni în contextul setului de date `gapminder_all` unde ne interesăm la evoluția duratei medii de viață în țara noastră după anul 1900.

```
gapminder_all %>%
  filter(country == "Romania",
         as.numeric(year) > 1900) %>%
  mutate(year = as.Date(year, format = "%Y")) %>%
ggplot(aes(x = year, y = lifeExp)) +
  geom_point() +
  scale_x_date(date_labels = "%y", date_breaks = "10 years")
```



1.6.2 Scale de culori

O altă clasă de scale des folosite sunt cele care corespund esteticii de culoare (`colour` - culoarea marginală/exterioară și respectiv `fill` - culoarea de umplere). Pachetul `ggplot2` pune la dispoziție mai multe tipuri de scale diferențiate după cum variabilele de scalare sunt continue sau discrete. Spațiul culorilor din `ggplot2` este de tip HCL (*hue*, *chroma* și *luminance*), pentru mai multe detalii se poate consulta http://www.handprint.com/HP/WCL/color7.html, dar se pot folosi si palete de culoare de tip Brewer (pentru explorarea acestora se poate vizita https://colorbrewer2.org/). Acestea din urmă pot fi clasificate în trei categorii: secvențiale, divergente și calitative unde primele două categorii corespund variabilelor continue iar cea de-a treia corespunde variabilelor cu valori discrete.

Set1 (qualitative)

PuBuGn (sequential)

Accent (qualitative)

Spectral (divergent)

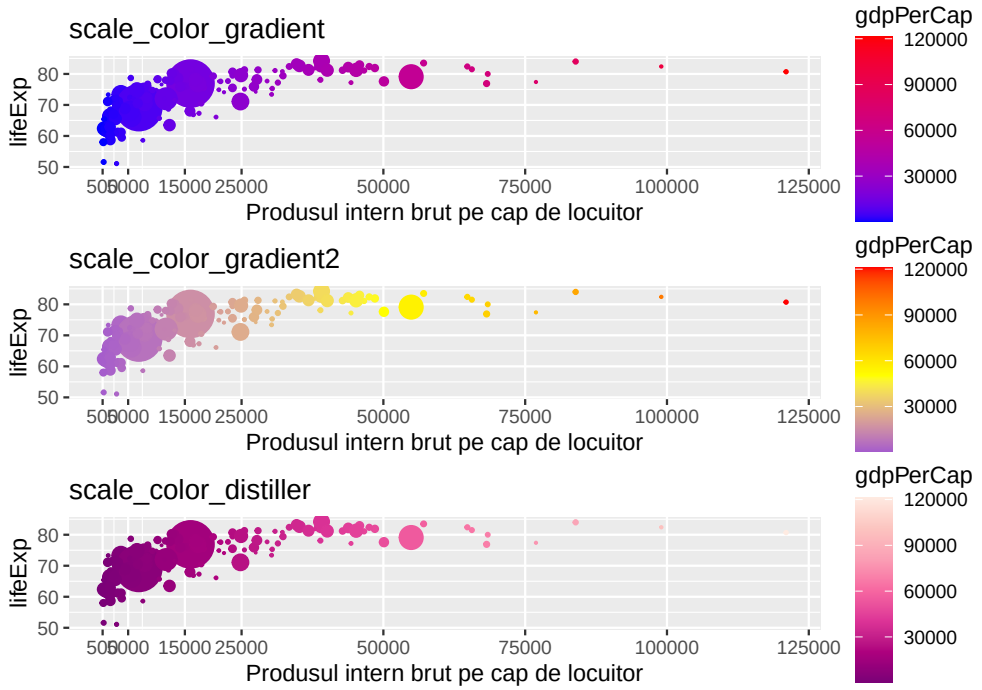
Atunci când utilizăm variabile continue se folosesc gradiente de culoare prin intermediul funcțiilor generice de forma `scale_[estetică culoare]_[tip]`, după cum urmează

Funcție generică	Scală	Tip	Descriere
`scale_[estetică culoare]_[tip]`	`colour`	`gradient`	Scală predefinită (similară cu `scale_colour_continuous()`) care folosește un gradient de culoare bazat pe două culori (*low* și *high*)
	`fill`	`gradient2`	Scală care folosește un gradient de culoare cu trei valori (*low*, *high* și *mid* - pentru acesta din urmă se poate selecta valoarea de *midpoint*)
		`gradientn`	Scală care folosește un gradient de culoare bazat pe n valori
		`distiller`	Scală care este bazată pe palete de tip Brewer

Pentru ilustrare vom folosi setul de date `gapminder_2018`:

```
# grafic de baza
baza = ggplot(gapminder_2018, aes(x = gdpPerCap, y = lifeExp, colour = gdpPerCap,
                                  size = pop)) +
  geom_point()

# gradient 2 culori
baza +
  scale_color_gradient(low = "blue", high = "red")

# gradient 3 culori
baza +
  scale_color_gradient(low = "blue", high = "red")

# gradient Brewer - RdPu
baza +
```

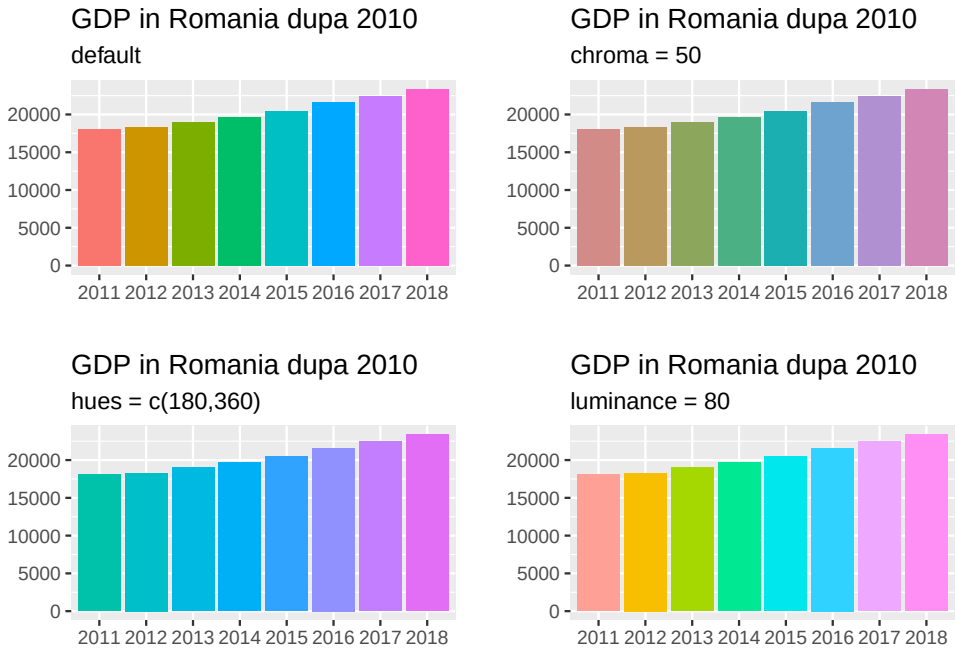
```
  scale_color_distiller(palette = "RdPu")
```



În cazul variabilelor discrete, putem folosi patru scale pentru fiecare dintre esteticile `colour` și `fill`: `scale_[estetică culoare]_hue()`, `scale_[estetică culoare]_brewer()`, `scale_[estetică culoare]_gray()` și `scale_[estetică culoare]_manual()`. Scala predefinită în cazul variabilelor discrete este scala `scale_[estetică culoare]_hue()` care alege culori egal depărtate pe roata de culori de tip HCL. Scala `scale_[estetică culoare]_brewer()` folosește paletele de culori de tip Brewer iar scala `scale_[estetică culoare]_gray()` folosește o paletă de griuri, de la gri deschis la gri închis.

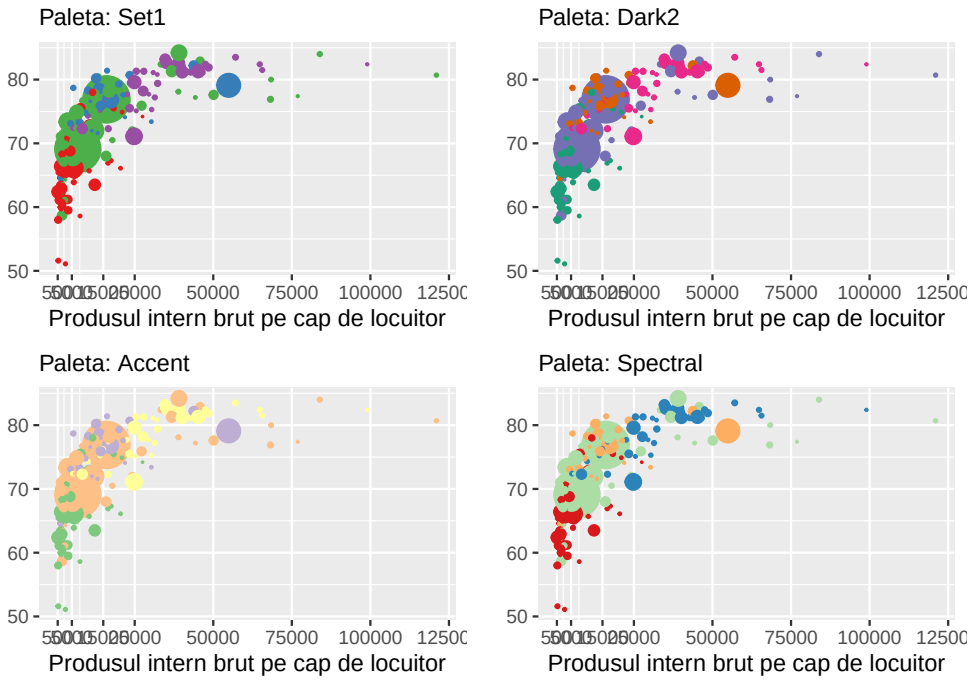
Pentru a ilustra scala `scale_[estetică culoare]_hue` vom folosi setul de date `gapminder_all` și vom afișa variația produsului intern brut pe cap de locuitor în România după anul 2000.

```
baza = gapminder_all %>%
  filter(country == "Romania",
         as.numeric(year) > 2010) %>%
  ggplot(aes(x = year, y = gdpPerCap, fill = year)) +
  geom_bar(stat = "identity")

baza +
  scale_fill_hue(c = 50)

baza +
  scale_fill_hue(h = c(180, 300))

baza +
  scale_fill_hue(l = 80)
```

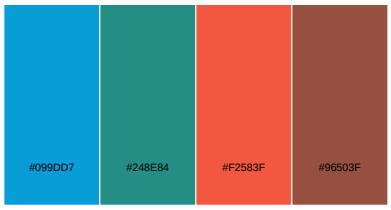


Următoarele grafice prezintă aceeași figură (situarea țărilor din setul de date `gapminder_2018` în funcție de produsul intern brut pe cap de locuitor și durată medie de viață) ilustrată cu ajutorul funcției `scale_[estetică culoare]_brewer()` pentru patru palete de culori diferite:

```
baza = ggplot(gapminder_2018, aes(x = gdpPerCap, y = lifeExp, colour = continent,
                                  size = pop), alpha = 0.7) +
  geom_point() +
  scale_x_continuous(name = "Produsul intern brut pe cap de locuitor",
                     breaks = c(500, 5000, 15000, 25000,
                                50000, 75000, 100000, 125000),
                     minor_breaks = c(500, 2500, 7500)) +
  scale_size(name = "Populatia",
             breaks = c(1e7, 5e7, 1e8, 2e8, 5e8, 1e9),
             range = c(0.1, 10)) +
  guides(size = "none", colour = "none") +
  labs(x = "", y = "")

p1 = baza +
  scale_color_brewer(palette = "Set1") +
  labs(subtitle = "Paleta: Set1")

p2 = baza +
  scale_color_brewer(palette = "Dark2") +
  labs(subtitle = "Paleta: Dark2")

p3 = baza +
  scale_color_brewer(palette = "Accent") +
  labs(subtitle = "Paleta: Accent")

p4 = baza +
  scale_color_brewer(palette = "Spectral") +
  labs(subtitle = "Paleta: Spectral")
```

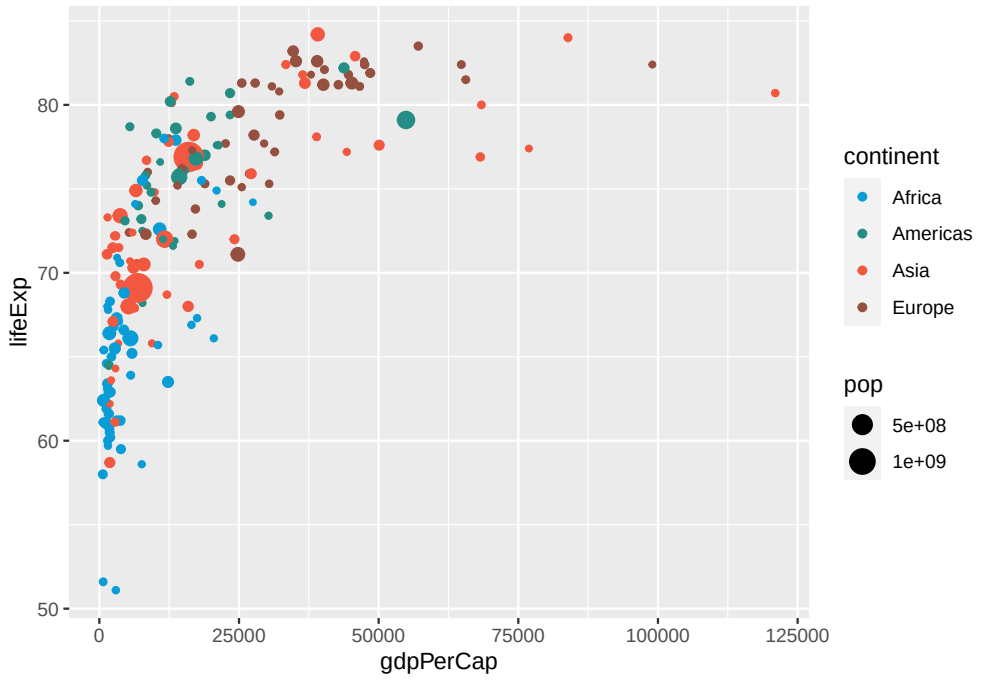


Funcția `scale_[estetică culoare]_manual()` este folosită în special atunci când dorim să folosim propria paletă de culori. De exemplu, să presupunem că dorim să colorăm statele de pe fiecare continent cu o culoare corespunzătoare



atunci putem scrie

```
ggplot(gapminder_2018, aes(x = gdpPerCap, y = lifeExp, colour = continent, size = pop)) +
  geom_point() +
  scale_colour_manual(values = c("#099DD7","#248E84","#F2583F", "#96503F"))
```

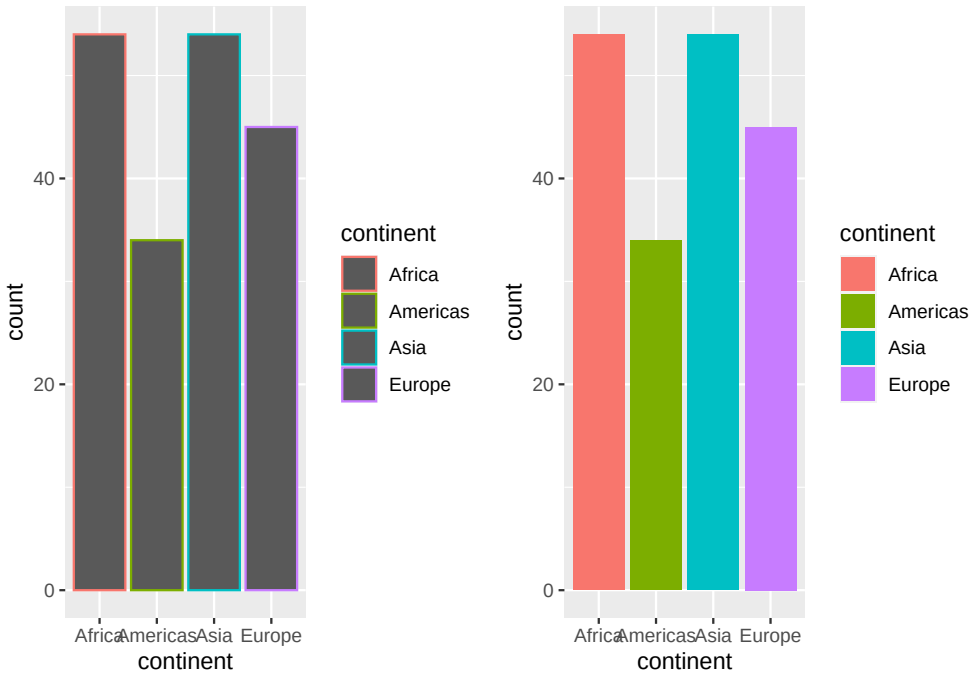


1.7 Ajustarea pozițiilor (*position adjustments*)

Fiecare strat geomentric *geom* prezintă, pe lângă o transformare statistică implicită, și o ajustare implicită a poziției care specifică un set de *reguli* privind modul în care componentele diferite ale graficului trebuie poziționate unele față de altele. Această poziționare implicită este vizibilă în special în cadrul funcției `geom_bar` atunci când aplicăm o variabilă diferită caracteristicii vizuale de colorare (e.g. `colour` sau `fill`).

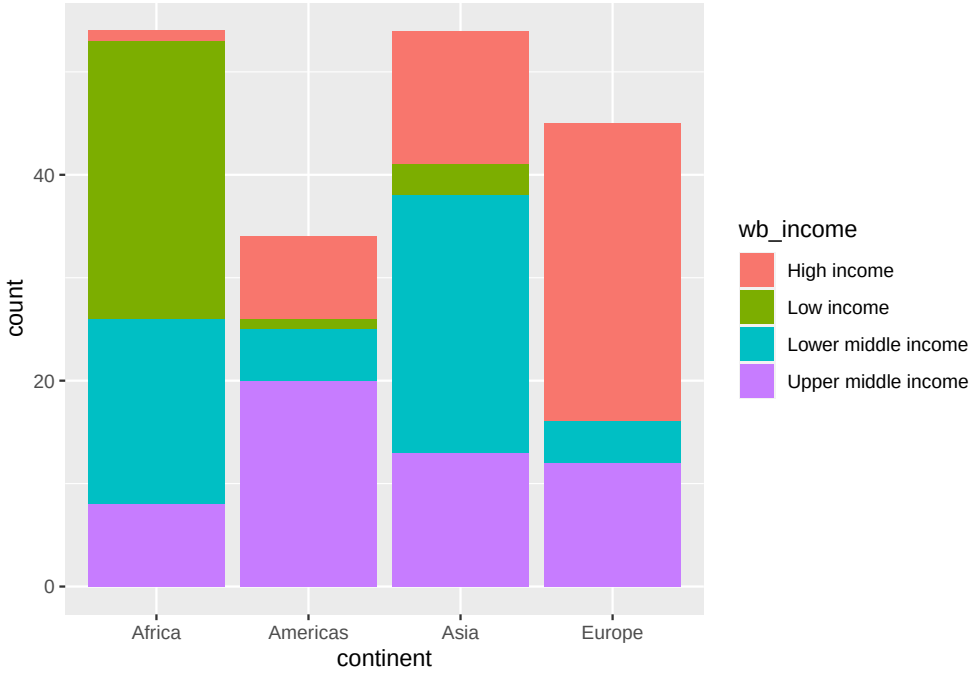
De exemplu, să considerăm diagrama cu bare care ilustrează numărul de tări de pe fiecare continent folosind setul de date `gapminder_2018`. Putem colora barele folosind sau estetica `colour` sau estetica `fill` (de preferat).

```
gapminder_2018 %>%
  ggplot() +
  geom_bar(aes(x = continent, colour = continent))

gapminder_2018 %>%
  ggplot() +
  geom_bar(aes(x = continent, fill = continent))
```



Acum dacă colorăm diagrama cu bare de mai sus în raport cu nivelul de venit (conform Băncii Mondiale) obținem o diagramă cu bare suprapuse colorate în funcție de această varaibilă.

```
gapminder_2018 %>%
  ggplot() +
  geom_bar(aes(x = continent, fill = wb_income))
```

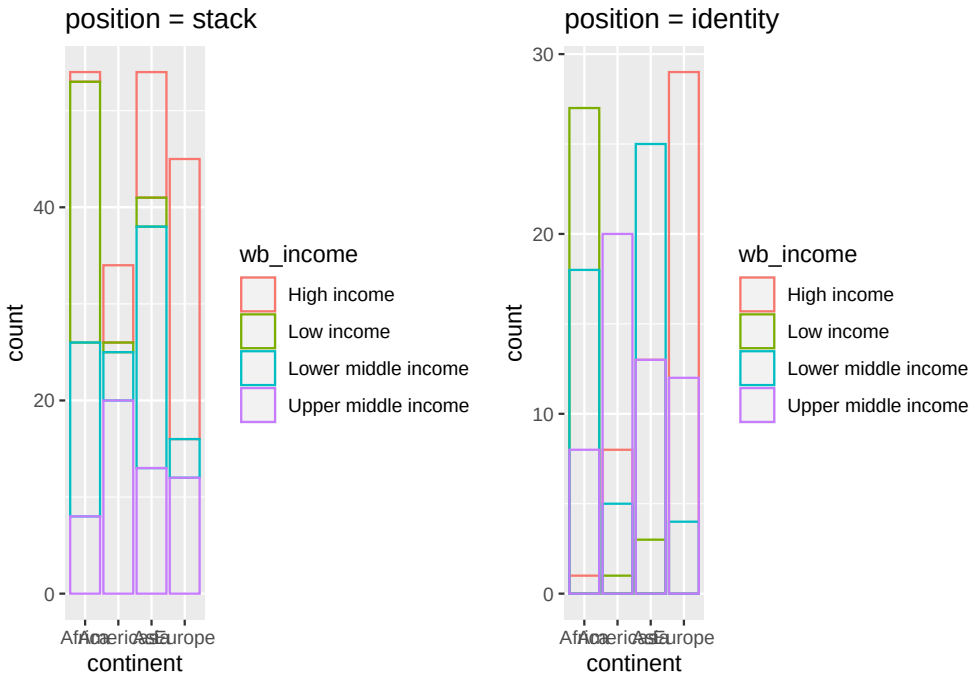


Suprapunerea barelor (`stack`) în `geom_bar` (înălțimea dreptunghiurilor este proporțională cu valoarea lor) se face în mod automat prin ajustarea poziției specificate de argumentul `position` (default `position = "stack"`). Dacă nu se dorește suprapunerea barelor atunci se poate folosi una din următoarele ajustări:

`identity`, `dodge` sau `fill`.

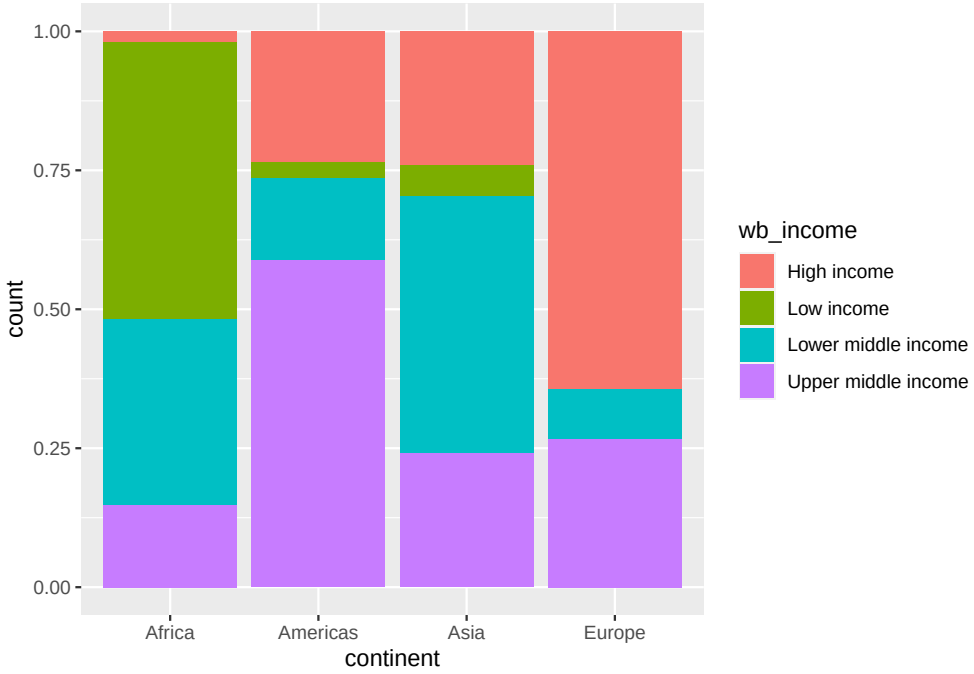
- `position = "identity"` - opțiunea impune plasarea fiecărui obiect exact unde se află în contextul figurii (nu este foarte folositoare atunci când trasăm diagrame cu bare dar este opțiunea de default pentru diagramele de împrăștiere)

```
gapminder_2018 %>%
  ggplot(aes(x = continent, colour = wb_income)) +
  geom_bar(position = "identity", fill = NA)
```



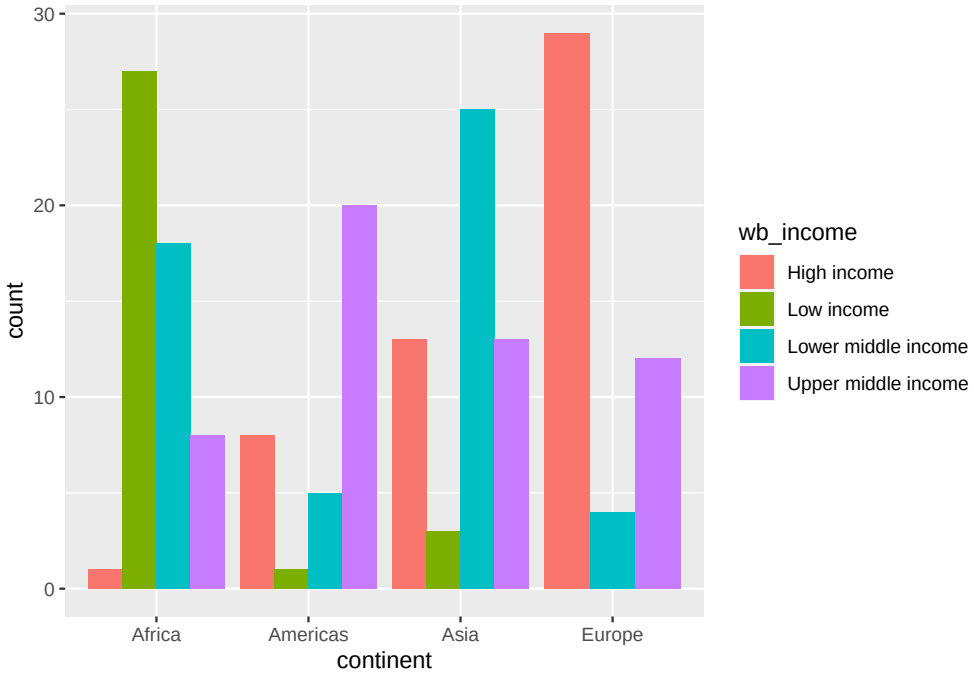
- `position = "fill"` - opțiunea este similară cu `stack`, suprapunând barele în așa fel încât înălțimea acestora să fie constantă (este de preferat atunci când dorim să comparăm proporții între grupuri)

```
gapminder_2018 %>%
  ggplot(aes(x = continent, fill = wb_income)) +
  geom_bar(position = "fill")
```



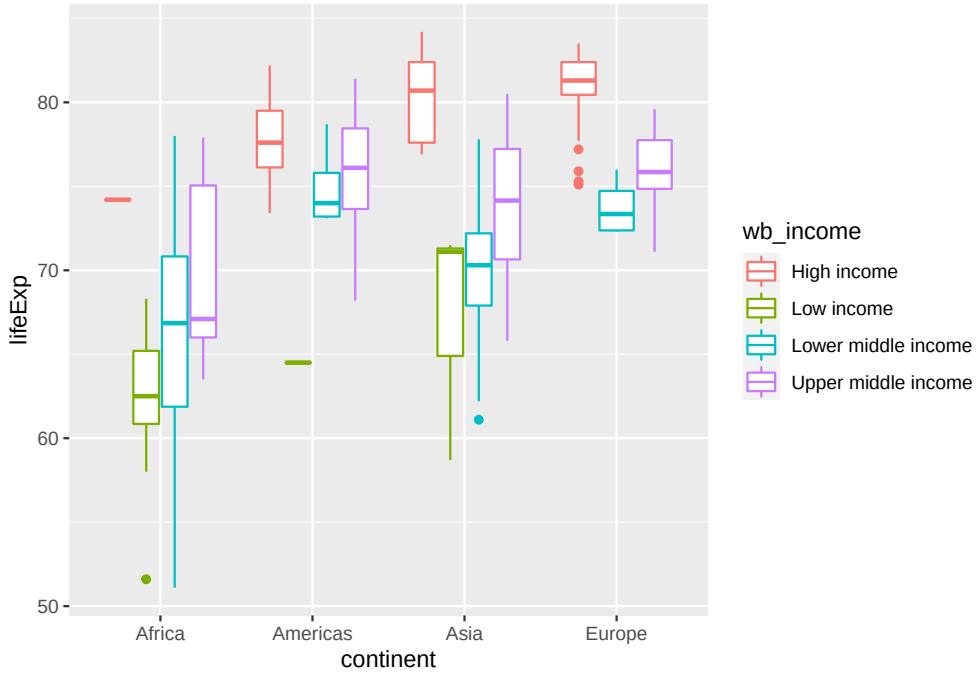
- `position = "dodge"` - opțiunea plasează obiectele unul lângă celălalt și permite astfel o mai bună comparare între valorile individuale

```
gapminder_2018 %>%
  ggplot(aes(x = continent, fill = wb_income)) +
  geom_bar(position = "dodge")
```



Trebuie menționat că `geom_boxplot` are ca și poziționare implicită poziționarea `dodge` după cum se poate observa și din exemplul de mai jos.

```
# position dodge - comportament de default pentru boxplot
gapminder_2018 %>%
  ggplot(aes(x = continent, y = lifeExp)) +
  geom_boxplot(aes(colour = wb_income))
```

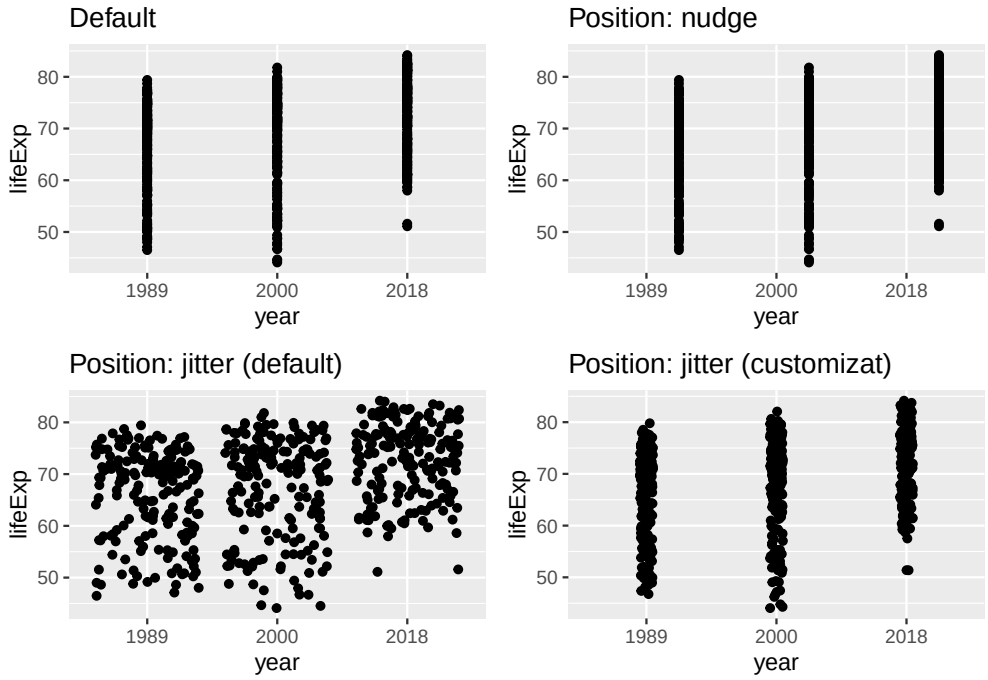


Tipurile de poziții prezentate mai sus se aplică în special diagramelor cu bare (sau boxplot-urilor) dar, trebuie menționat că pachetul `ggplot2` pune la dispoziție și trei metode de ajustare a pozițiilor în cazul punctelor: `position_nudge()` - asigură mutarea punctelor printr-o valoare fixată (default în cazul `geom_text()`); `position_jitter()` - adaugă poziției fiecărui punct un zgomot aleator pentru a dispersa un pic datele; `position_jitterdodge()` - permite separarea/evitarea (*dodge*) puntelor în interiorul grupurilor și apoi adaugă zgomot aleator pozițiilor.

Vom ilustra primele două ajustări de poziții în contextul setului de date `gapminder_all` în care afișăm pentru anii 1989, 2000 și 2018 speranța medie de viață pentru fiecare țară.

```
# initial
gapminder_all %>%
  filter(year %in% c("1989", "2000", "2018")) %>%
  ggplot(aes(x = year, y = lifeExp)) +
  geom_point()

# position nundge
gapminder_all %>%
  filter(year %in% c("1989", "2000", "2018")) %>%
  ggplot(aes(x = year, y = lifeExp)) +
  geom_point(position = position_nudge(x = 0.25, y = 0))

# position jitter - default
gapminder_all %>%
  filter(year %in% c("1989", "2000", "2018")) %>%
  ggplot(aes(x = year, y = lifeExp)) +
  geom_point(position = "jitter")
```

```
#position jitter - cu specificatii
gapminder_all %>%
  filter(year %in% c("1989", "2000", "2018")) %>%
  ggplot(aes(x = year, y = lifeExp)) +
  geom_point(position = position_jitter(width = 0.05, height = 0.5))
```



Pentru mai multe informații referitoare la tipurile de ajustare a pozițiilor în obiectele grafice din `ggplot2` se poate consulta documentația fiecărei funcții `geom` sau se poate apela comanda

```
help.search("position_", package = "ggplot2")
```

1.8 Sisteme de coordonate (*coordinate systems*)

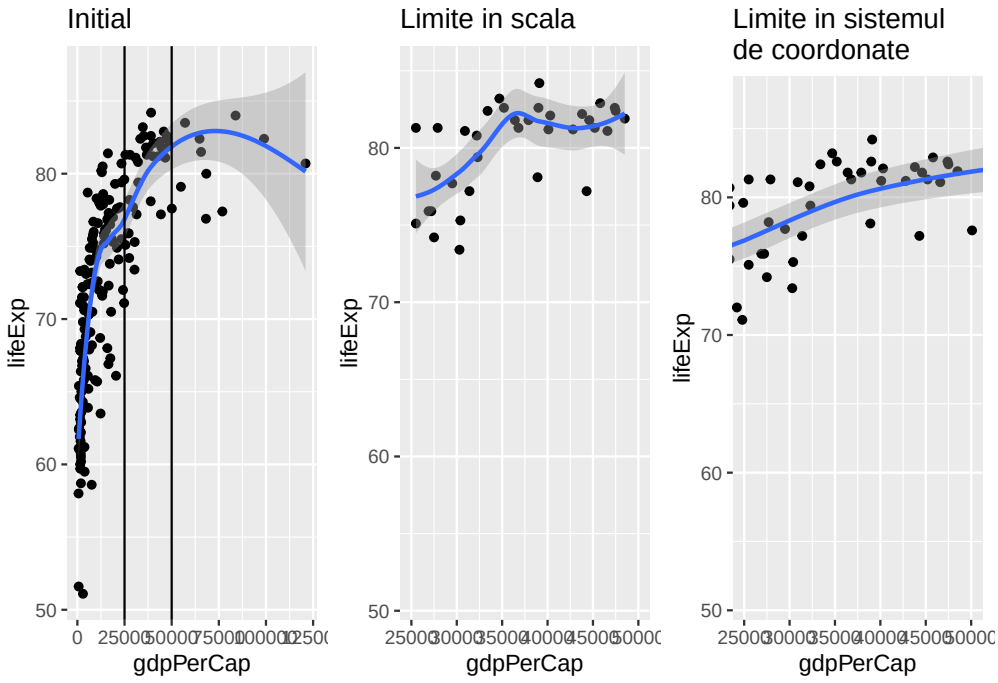
Pachetul `ggplot2` pune la dispiziție și o serie de funcții care fac referire la sistemul de coordonate în care sunt trasate elementele grafice. Sistemele de coordonate sunt specificate prin funcții care încep prin `coord_` și sunt adăugate ca straturi la graficul existent. Pachetul dispune de mai multe sisteme de coordonate printre care enumerăm:

- `coord_cartesian` - sistemul cartezian care este și cel de default și în care trebuie specificare valorile pe x și y
- `coord_flip` - sistemul cartezian în care axele sunt schimbate între ele
- `coord_fixed` - un sistem cartezian care păstrează fixat aspectul (*aspect ratio* - numărul de unități pe axa y care corespund unei unități pe axa x) cu raportul de bază de 1
- `coord_polar` - sistemul de coordonate polar
- `coord_quickmap` - un sistem de coordonate care aproximează sistemul sferic al planeti în vederea trasării hărților

Folosind sistemul cartezian cu opțiunea `xlim` (respectiv `ylim`) putem vizualiza doar acea regiune a graficului care ne interesează (putem face zoom). Diferența majoră dintre utilizarea limitelor în interiorul funcției `coord_cartesian` și utilizarea limitelor în funcțiile de scală (e.g. `scale_x_continuous`) este că în primul caz utilizăm tot setul de date pe când în cel de al doilea caz sunt folosite doar observațiile care se află între

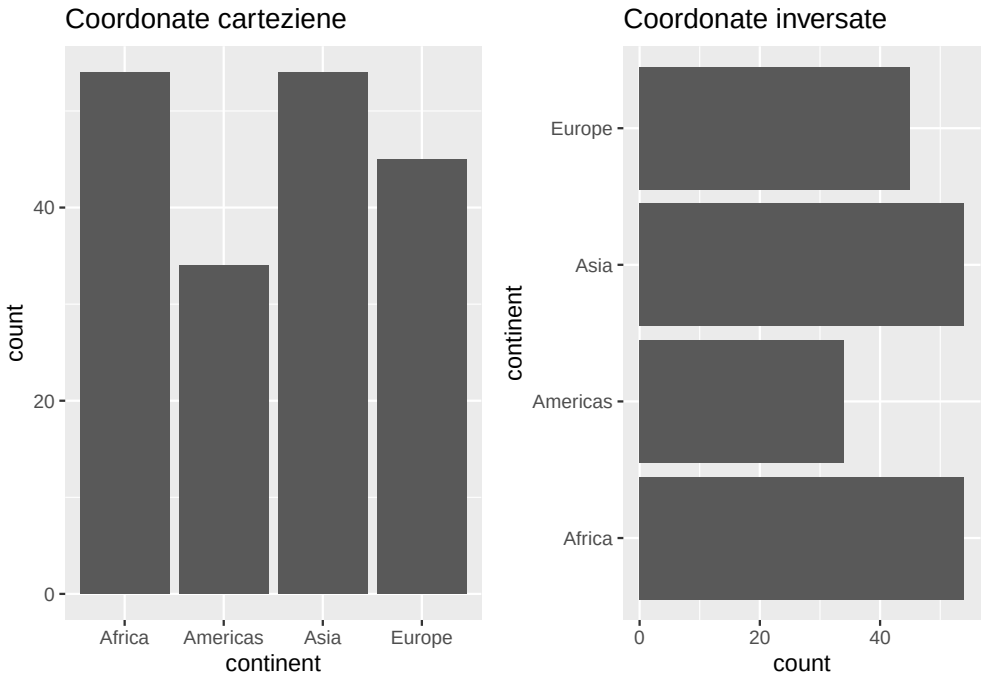
limitele setate, toate celelalte puncte fiind excluse.

```
# folosind scala - fitam curba de regresie pe datele ramase
ggplot(gapminder_2018, aes(x = gdpPerCap, y = lifeExp)) +
  geom_point() +
  geom_smooth() +
  scale_x_continuous(limits = c(25000, 50000))

# folosind sistemul de coordonate - fitam curba de regresie
# pe toate datele si evidentiem doar zona de interes
ggplot(gapminder_2018, aes(x = gdpPerCap, y = lifeExp)) +
  geom_point() +
  geom_smooth()+
  coord_cartesian(xlim = c(25000, 50000))
```

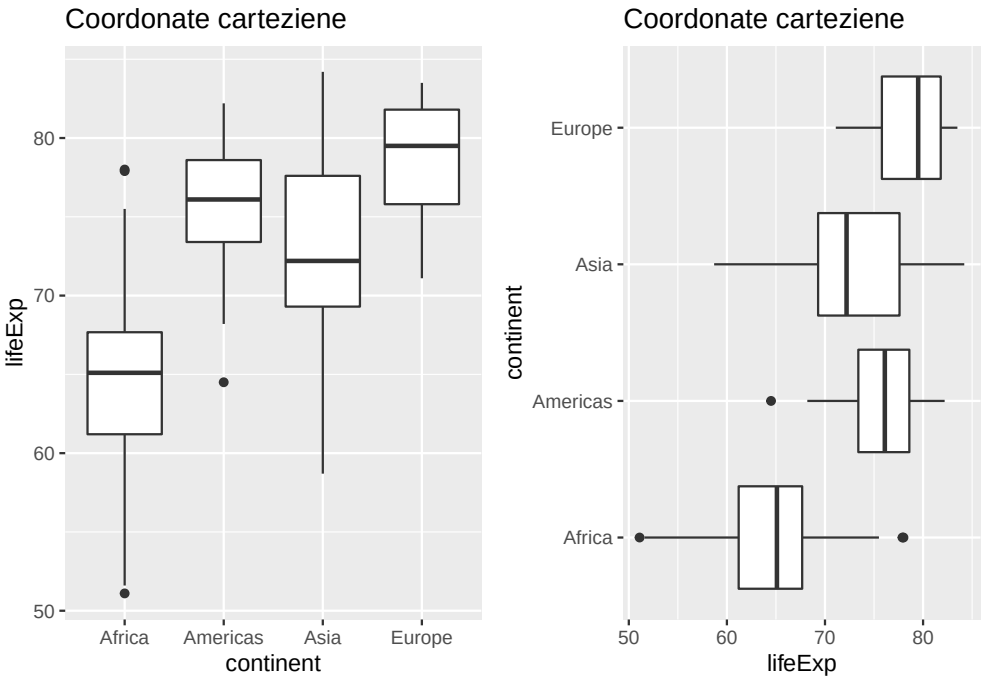


Sunt multe situațiile în care suntem interesați de valorile variabilei de pe axa x condiționate la variabila de pe axa y și în acest caz este recomandată schimbarea (întoarcerea) sistemului de coordonate prin apelarea funcției `coord_flip()`. De asemenea, o altă situație în care se recomandă rotirea coordonatelor cu 90 de grade este în cazul trasării unei diagrame cu bare sau a unui boxplot în funcție de o variabilă calitativă cu un număr mare de categorii sau cu categorii a căror denumire este lungă. Vom ilustra această transformare a sistemului de coordonate folosind diagrama cu bare pentru a evidenția numărul țărilor din setul de date `gapminder_2018` de pe fiecare continent.

```
# coordonate carteziene
gapminder_2018 %>%
  ggplot(aes(x = continent)) +
  geom_bar()

# coordonate flip
gapminder_2018 %>%
  ggplot(aes(x = continent)) +
  geom_bar() +
  coord_flip()
```



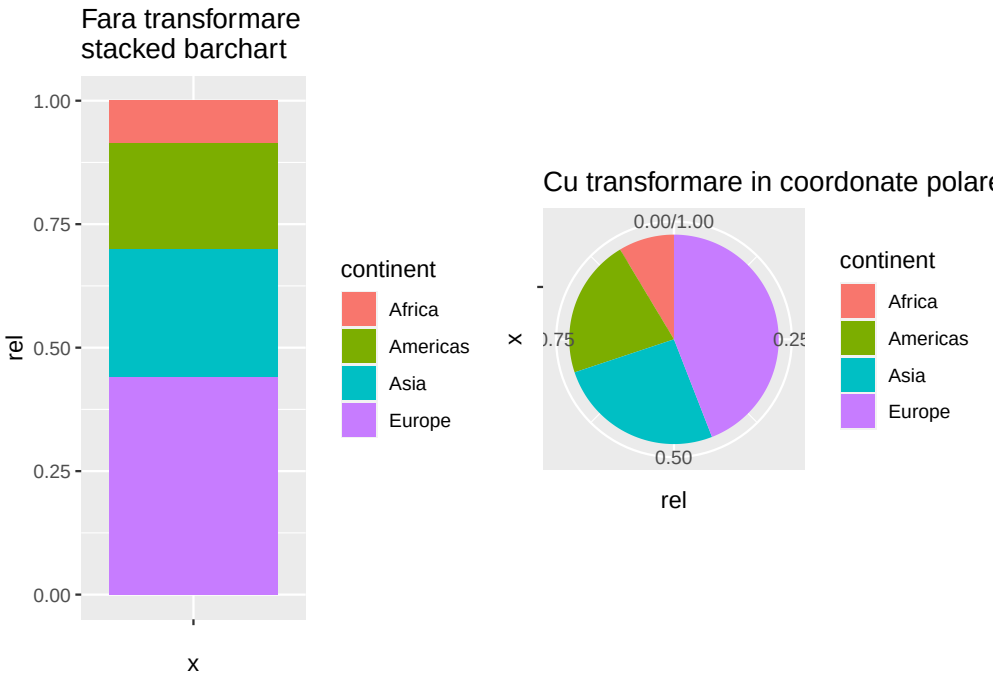
În mod similar putem vizualiza prin intermediul unui boxplor diferențele dintre duratele medii de viață de pe fiecare continent:



Un alt mod de vizualizare a datelor calitative (categoriale), frecvent utilizat în practică, este reprezentat de diagrama circulară (pie chart). Trebuie menționat că acest tip de grafic nu este recomandat având în vedere că oamenii tind să supraestimeze unghiurile mai mari de 90 de grade și să subestimeze unghiurile mai mici de 90 de grade astfel fiind dificilă compararea mărimii relative a două sectoare din diagrama circulară (Robbins 2013). Pentru a crea o diagramă circulară în `ggplot2` vom folosi transformarea în coordonate polare `coord_polar`. Ideea este de a pleca de la o diagramă cu bare suprapuse (*stacked barchart*) și apoi să o transpunem în coordonate polare.

Figura de mai jos prezintă o diagramă circulară pentru proporția țărilor de pe fiecare continent care depășesc produsul intern brut median global.

```
gapminder_2018 %>%
  mutate(gdp_avg = median(gdpPerCap)) %>%
  filter(gdpPerCap > gdp_avg) %>%
  mutate(n_all = n()) %>%
  group_by(continent) %>%
  summarize(rel = n() / unique(n_all)) %>%
  ggplot(aes(x = "", y = rel)) +
  geom_col(aes(fill = continent)) +
  coord_polar("y")
```



Mai multe detalii despre sistemele de coordonate se pot găsi în lucrarea de referință (Wickham 2016).

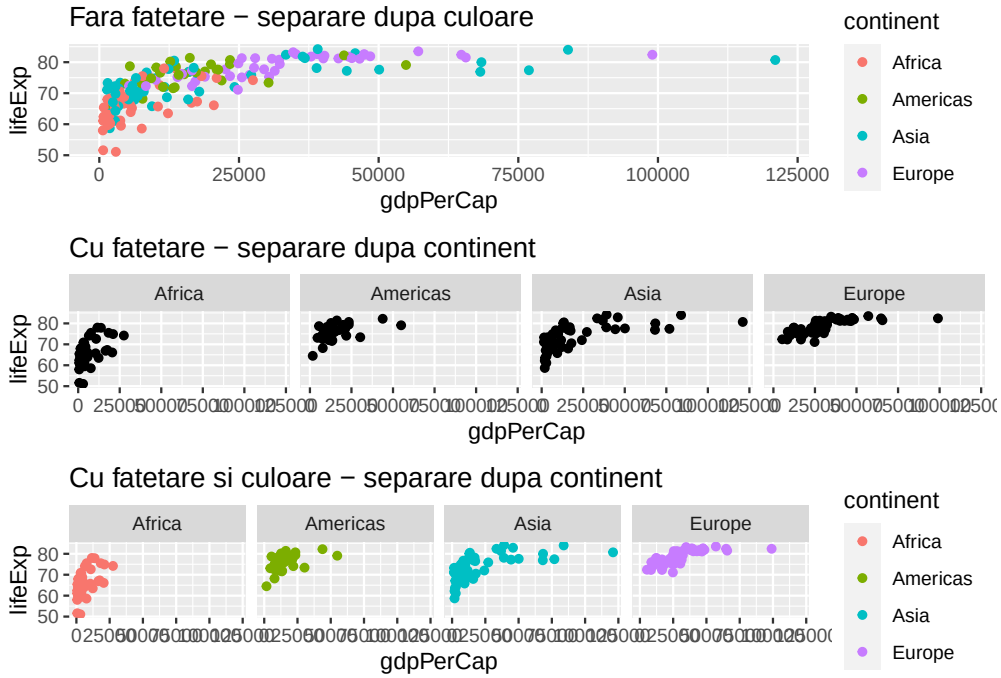
1.9 Fațetare (*facets*)

Procesul de *fațetare* (*facets*) presupune divizarea unei figuri în subfiguri determinate de valorile unei variabile categoriale sau a mai multor variabile categoriale. Astfel, în cazul în care vorbim de o singură varaibilă, pentru fiecare categorie a variabilei este trasat un subgrafic. Pachetul `ggplot2` pune la dispoziție două funcții utile pentru a efectua această operație:

- `facet_wrap()` - folosită atunci când avem de-a face cu o singură varaibilă de grupare a datelor
- `facet_grid()` - folosită atunci când dorim secționarea datelor după două sau mai multe variabile

Spre exemplu, setul de date `gapminder_2018` conține informații referitoare la produsul intern brut ajustat (`gdpPerCap`) și speranța medie de viață (`lifeExp`) pentru mai mult de 180 de țări de pe patru continente (`continents`). Dacă dorim să vedem relația dintre cele două variabile în funcție de continent atunci una dintre variante ar fi să folosi estetica `colour` pentru a diferenția. O variantă alternativă, care uneori ilustrează mai clar diferențele, este de a subsecționa graficul în subgrafice ce corespund fiecărui continent folosind funcția `facet_grid()`.

```
ggplot(gapminder_2018, aes(x = gdpPerCap, y = lifeExp, colour = continent)) +
  geom_point()

ggplot(gapminder_2018, aes(x = gdpPerCap, y = lifeExp)) +
  geom_point() +
  facet_grid(.~continent)
```

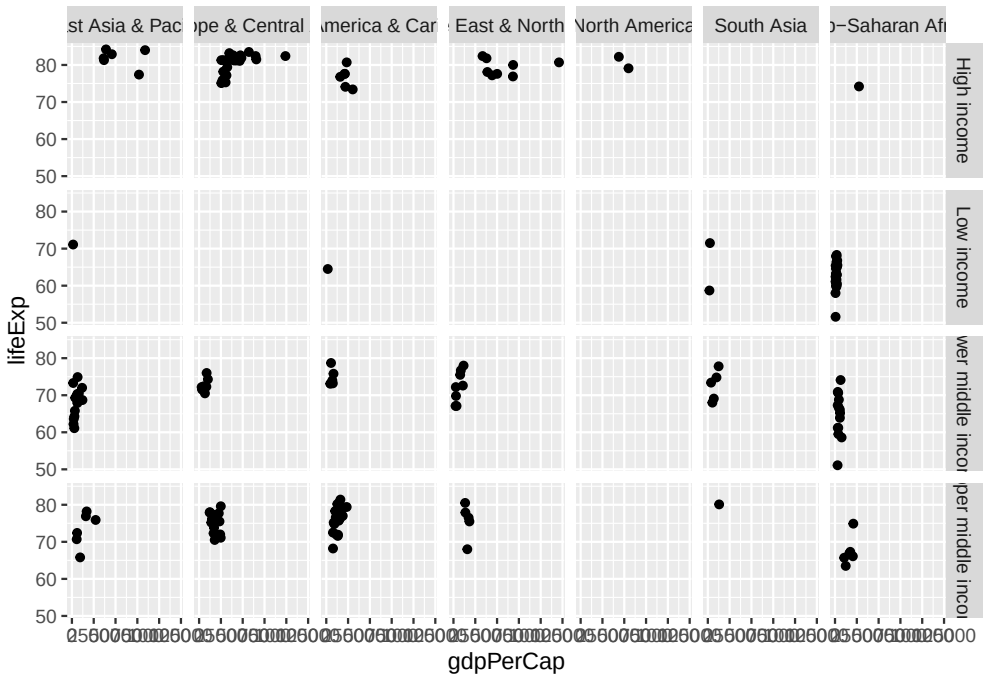


În figura de mai sus am folosit `facet_grid` pentru a separa în subgrafice chiar dacă separarea s-a făcut în funcție de o singură variabilă (*continent*). Observăm că în apelarea funcției `facet_grid` (`?facet_grid`) am folosit simbolul `~` care separă variabila/variabilele din stânga (trasată/trasate de-a lungul liniilor) de cea/cele din dreapta (trasată/trasate pe coloane). Codul generic pentru funcția `facet_grid` este

```
facet_grid([factor pentru linii] ~ [factor pentru coloane])
```

Pentru ilustrare să considerăm că ne dorim să investigăm relația dintre `gdpPerCap` și `lifeExp` în funcție de regiunea și nivelul de venit stabilite de Banca Mondială (variabilele `wb_region` și respectiv `wb_income`).

```
ggplot(gapminder_2018, aes(x = gdpPerCap, y = lifeExp)) +
  geom_point() +
  facet_grid(wb_income ~ wb_region)
```

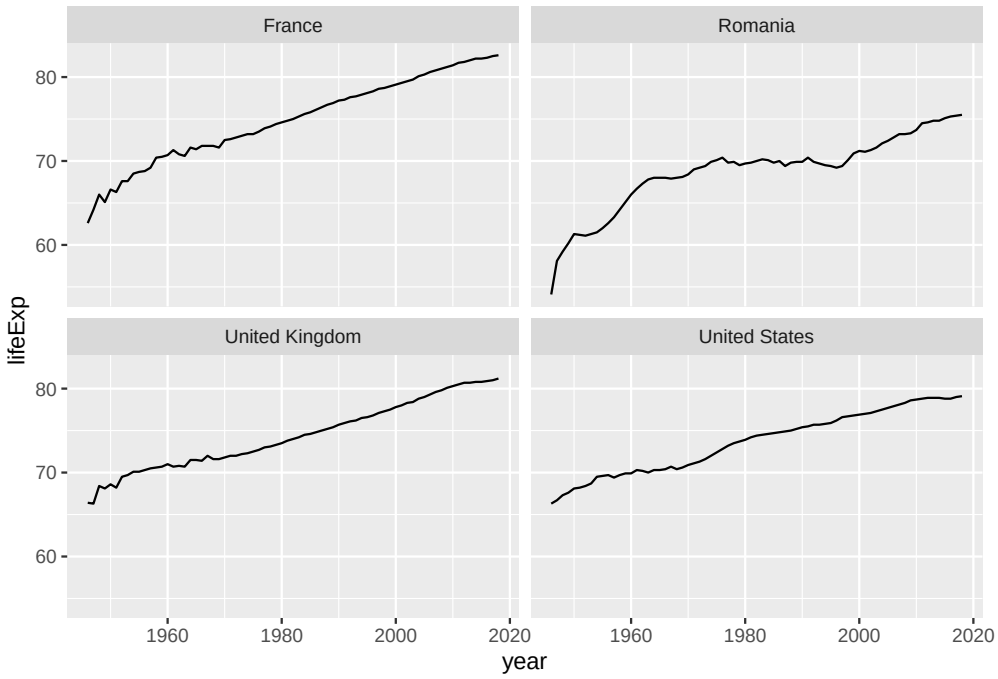


Funcția `facet_wrap()` este folosită cu precădere atunci când dorim secționarea datelor după o singură variabilă calitativă (dar poate fi folosită și cu mai multe - `?facet_wrap`). Prin aplicarea acestei funcții, subgraficele corespunzătoare fiecărui nivel al varaibilei de separare nu trebuie să fie repartizate pe o singură linie sau coloană în schimb se poate specifica numărul de coloane (sau de linii) dorite. Codul generic pentru funcția `facet_wrap` este

```
facet_wrap( ~ [variabila de fatetare],
            ncol = [numar de coloane])
```

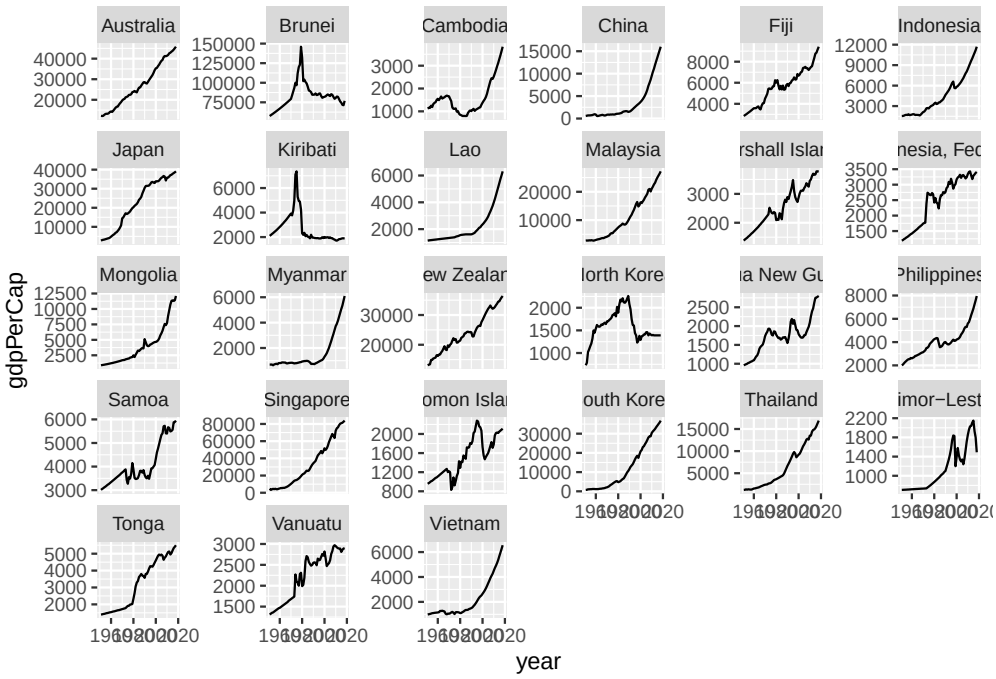
Vom ilustra aplicarea acestei funcții în contextul setului de date `gapminder_all`. Vom afișa cum a evoluat evoluția duratei medii de viață pentru patru țări (*Statele Unite*, *UK*, *Franța* și *România*) după Cel de-al Doilea Război Mondial:

```
gapminder_all %>%
  mutate(year = as.numeric(year)) %>%
  filter(country %in% c("United States", "France", "United Kingdom", "Romania"),
         year > 1945) %>%
  ggplot(aes(x = year, y = lifeExp)) +
  geom_line() +
  facet_wrap(~country,
             ncol = 2)
```



În figura de mai jos ilustrăm evoluția produsului intern brut pentru statele din Asia de Est după anul 1950. Opțiunea `ncol` descrie numărul de coloane în care vrem să fie împărțit grid-ul de subgrafice iar opțiunea `scales = "free_y"` permite ajustarea scalei pentru axa y pentru fiecare categorie (țară) în parte.

```
gapminder_all %>%
  mutate(year = as.numeric(year)) %>%
  filter(six_regions == "east_asia_pacific",
         year > 1950) %>%
  ggplot(aes(x = year, y = gdpPerCap)) +
  geom_line()+
  facet_wrap(~country,
             ncol = 6, scales = "free_y")
```



1.10 Customizarea graficelor

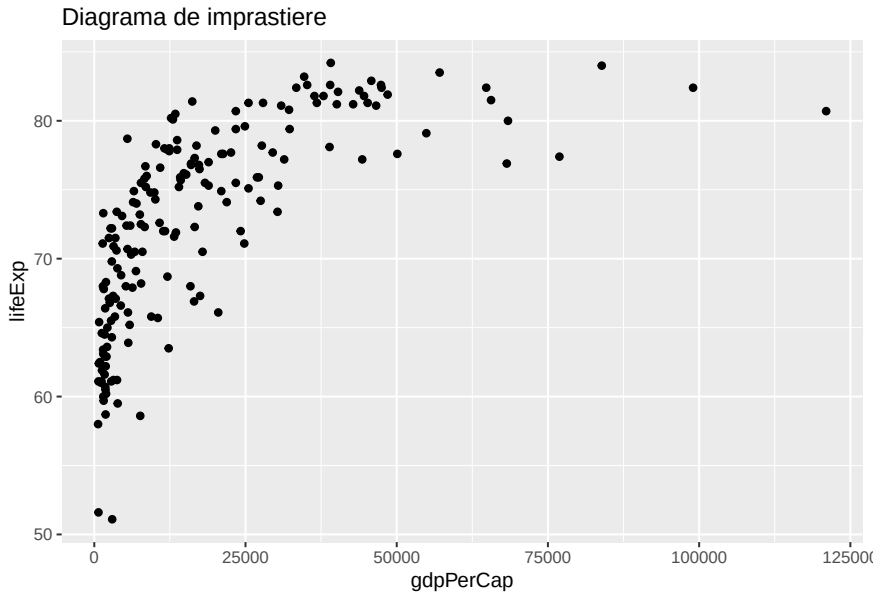
În această secțiune vom prezenta o serie de metode și tehnici de personalizare/customizare a graficelor atât prin adăugarea de elemente clasice precum titlu, etichetă pentru axe, etc. cât și prin modificarea temelor grafice.

1.10.1 Personalizarea elementelor grafice (axe, titlu, etc.)

Titlurile, etichetele axelor și adnotările textuale (pe grafic, axe, obiecte geometrice și legendă) reprezintă un aspect important în crearea unei figuri prin care vrem să transmitem o serie de informații. Pachetul `ggplot2` pune la dispoziție mai multe funcții care permit efectuarea adnotării elementelor grafice fără mare dificultate.

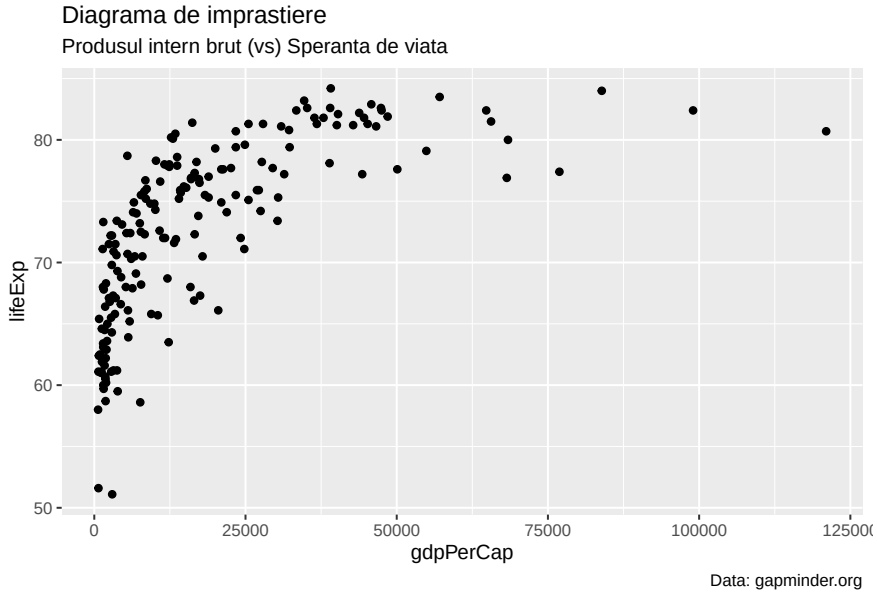
Astfel putem adauga/modifica titlul unui grafic folosind comanda `ggtitle([titlu])`:

```
p = ggplot(gapminder_2018, aes(x = gdpPerCap, y = lifeExp))+
  geom_point()

p + ggtitle("Diagrama de imprastiere")
```



sau alternativ putem utiliza funcția `labs(title = [titlu])` care în plus permite adăugarea unui subtitlu (`subtitle`) si a unei surse/credit (`caption`):

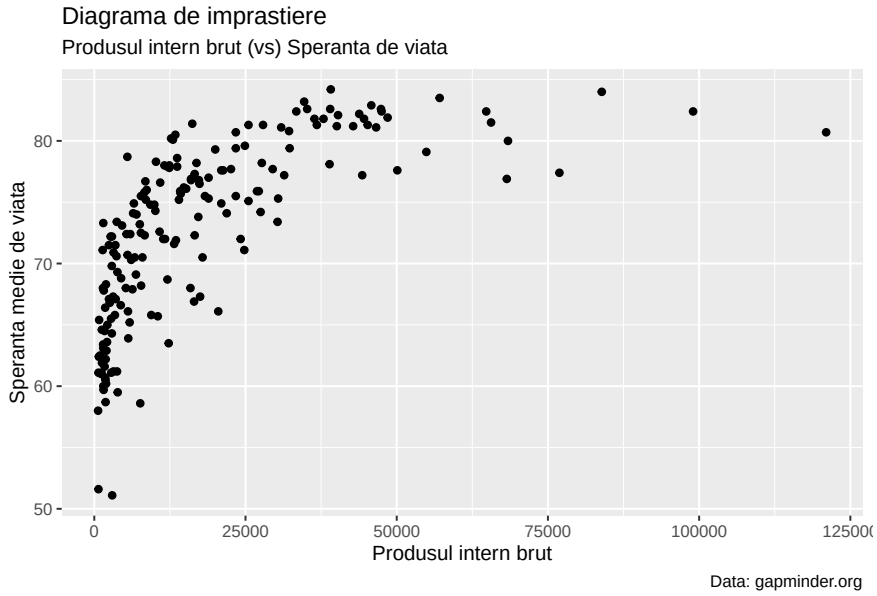
```
p +
  labs(title = "Diagrama de imprastiere",
       subtitle = "Produsul intern brut (vs) Speranta de viata",
       caption = "Data: gapminder.org")
```



Pentru modificarea etichetelor axelor de coordonate putem folosi sau funcțiile `xlab()` pentru axa x și respectiv `ylab()` pentru axa y sau comanda `labs(x = [eticheta axei x], y = [eticheta axei y])`.

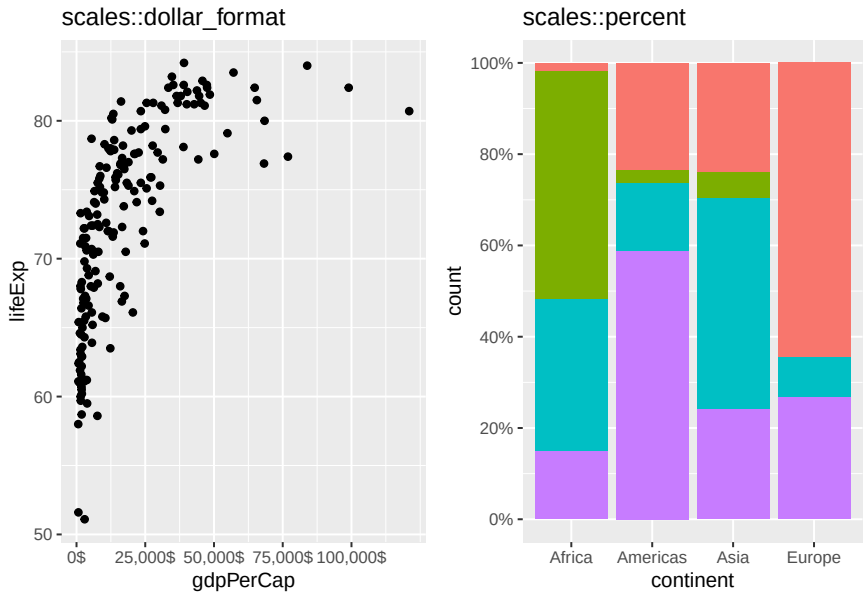
```
p +
  labs(title = "Diagrama de imprastiere",
       subtitle = "Produsul intern brut (vs) Speranta de viata",
       caption = "Data: gapminder.org",
       x = "Produsul intern brut",
```

```
       y = "Speranta medie de viata")
```



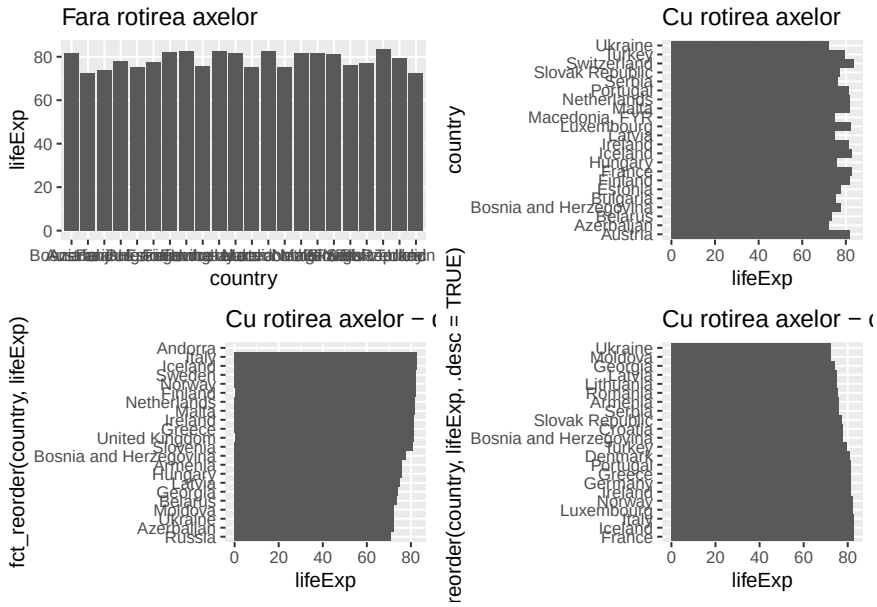
De asemenea, am văzut că putem modifica/adăuga nume axelor și prin specificarea acestuia parametrului `name` într-o scală (e.g. `scale_x_continuous`). Atunci când situația impune este recomandat să folosim unităle de măsură ale valorilor variabilelor. Pachetul `scales`, https://scales.r-lib.org/, pune la dispoziție o serie de funcții ajutătoare, de exemplu atunci când unitățile de măsură sunt `$` sau `%`:

```
ggplot(gapminder_2018, aes(x = gdpPerCap, y = lifeExp))+
  geom_point() +
  scale_x_continuous(labels = scales::dollar_format(prefix = "", suffix = "$")) +
  ggtitle("scales::dollar_format")

gapminder_2018 %>%
  ggplot(aes(x = continent, fill = wb_income)) +
  geom_bar(position = "fill") +
  scale_y_continuous(breaks = seq(0, 1, by = .2), labels = scales::percent)+
  ggtitle("scales::percent")
```



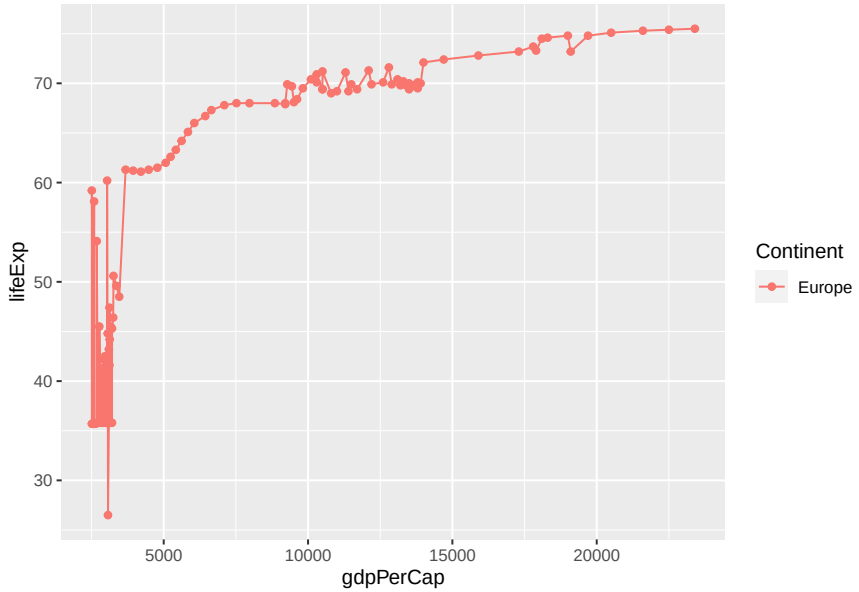
Totodată se pot întâlni situații în care axa x necesită etichete de lungime mai lungă, conducând astfel la o supraaglomerare a acestora, și în acest caz este recomandată rotirea axelor (`coord_flip()` - poate și ordonarea nivelelor `fct_reorder()`).

```
set.seed(1234)

gapminder_2018 %>%
  filter(continent == "Europe") %>%
  sample_frac(0.5) %>%
  ggplot(aes(x = country, y = lifeExp)) +
  geom_bar(stat = "identity") +
  ggtitle("Fara rotirea axelor")

set.seed(1234)

gapminder_2018 %>%
  filter(continent == "Europe") %>%
  sample_frac(0.5) %>%
  ggplot(aes(x = country, y = lifeExp)) +
  geom_bar(stat = "identity") +
  ggtitle("Cu rotirea axelor") +
  coord_flip()
```



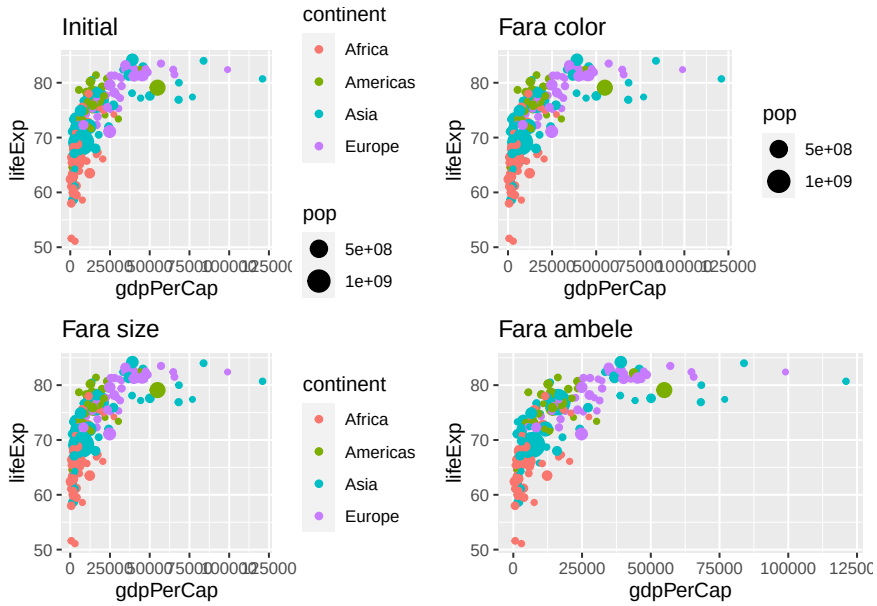
Un alt element grafic important în crearea unei figuri este legenda. Am văzut că în `ggplot2` legenda este creată automat atunci când folosim un element estetic precum `colour`, `fill`, `size`, `shape`, etc. După cum am văzut, legendele și axele sunt cele două elemente de ghidaj (*guides*) care pot fi modificate prin intermediul funcțiilor de scalare. Spre deosebire de axe, legendele sunt elemente mai complexe deoarece, pe lângă faptul că pot fi poziționate în diverse locuri, ele pot afișa mai multe estetici simultan (e.g. `colour`, `shape`) provenite de la mai multe straturi geometrice.

```
gapminder_all %>%
  filter(country == "Romania",
         as.numeric(year) > 1900) %>%
ggplot(aes(x = gdpPerCap, y = lifeExp, colour = four_regions)) +
  geom_point() +
  geom_line() +
  # schimbam titlul legendei
  labs(colour = "Continent")
```



Atunci când vrem să eliminăm legenda sau stratul care nu vrem să apară în legendă avem la dispoziție mai multe variante: folosim funcția `guides([nume estetica] = FALSE)`, funcția `scale_[nume estetică]_[tip](guide = FALSE)` sau argumentul `show.legend = FALSE` în elementul geometric:

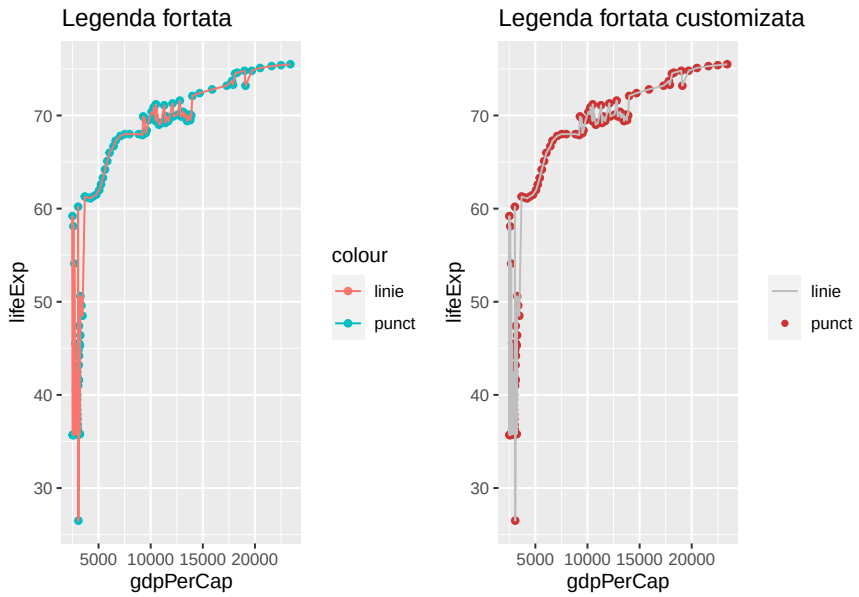
```
ggplot(gapminder_2018, aes(x = gdpPerCap, y = lifeExp, colour = continent, size = pop)) +
  geom_point() +
  ggtitle("Initial")

ggplot(gapminder_2018, aes(x = gdpPerCap, y = lifeExp, colour = continent, size = pop)) +
  geom_point() +
  scale_color_discrete(guide = FALSE) +
  ggtitle("Initial")

ggplot(gapminder_2018, aes(x = gdpPerCap, y = lifeExp, colour = continent, size = pop)) +
  geom_point() +
  guides(size = FALSE) +
  ggtitle("Initial")
```



Pachetul `ggplot2` nu va adăuga o legendă în mod automat decât dacă avem o corespondență între un element estetic (`colour`, `size`, etc.) și o variabilă dar sunt multe situațiile în care dorim să evidențiem elementele din grafic printr-o legendă. Putem *forța* apariția unei legende prin maparea în interiorul obiectului geometric (atribuirea) a unei estetici într-o valoare fixată. Mai mult, în cazul în care dorim modificarea aspectului elementelor geometrice din legendă putem folosi funcția `guide_legend()` cu opțiunea `override.aes`.

```
gapminder_all %>%
  filter(country == "Romania",
         as.numeric(year) > 1900) %>%
ggplot(aes(x = gdpPerCap, y = lifeExp)) +
  geom_point() +
  geom_line() +
  labs(title = "Initial")

gapminder_all %>%
  filter(country == "Romania",
         as.numeric(year) > 1900) %>%
```

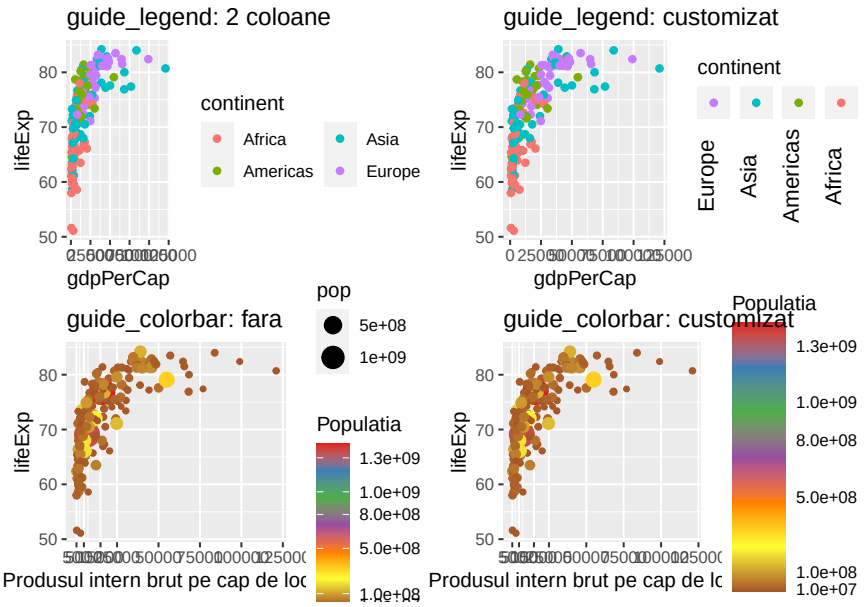
```
ggplot(aes(x = gdpPerCap, y = lifeExp)) +
  geom_point(aes(colour = "punct")) +
  geom_line(aes(colour = "linie")) +
  labs(title = "Legenda fortata")

gapminder_all %>%
  filter(country == "Romania",
         as.numeric(year) > 1900) %>%
ggplot(aes(x = gdpPerCap, y = lifeExp)) +
  geom_point(aes(colour = "puncte")) +
  geom_line(aes(colour = "linie")) +
  # modificam culorile
  scale_color_manual("", guide = "legend",
                     values = c("puncte" = "brown3",
                                "linie" = "gray")) +
  # vrem sa apara doar linie si punct
  guides(color = guide_legend(override.aes = list(linetype = c(1, 0),
                                                  shape = c(NA, 16)))) +
  labs(title = "Legenda fortata customizata")
```



Funcțiile ajutătoare `guide_legend()` și `guide_colorbar()` oferă un control suplimentar asupra detaliilor de prezentare ale unei legende, astfel prima poate fi aplicată atât elementelor estetice discrete cât și celor continue pe când cea de-a doua este aplicată doar elementelor estetice continue (este folosită pentru reprezentarea gamelor continue de culori). Pentru utilizarea acestor funcții, ele trebuie atribuite parametrului `guide` (i.e. `guide = guide_legend(...)`) din funcția de scală corespunzătoare esteticii pe care dorim să o modificăm sau mai simplu în interiorul funcției ajutătoare `guides` (i.e. `guides([estetică] = guide_legend(...)`).

```
# guide_legend
# pe doua coloane
ggplot(gapminder_2018, aes(x = gdpPerCap, y = lifeExp, colour = continent)) +
  geom_point() +
  guides(colour = guide_legend(ncol = 2))
# customizata
ggplot(gapminder_2018, aes(x = gdpPerCap, y = lifeExp, colour = continent)) +
  geom_point() +
```

```
  guides(colour = guide_legend(reverse = TRUE,
                               direction = "horizontal",
    title.position = "top",
    label.position = "bottom",
    label.hjust = 0.5,
    label.vjust = 1,
    label.theme = element_text(angle = 90)))

# guide_colorbar
ggplot(gapminder_2018, aes(x = gdpPerCap, y = lifeExp, colour = pop, size = pop)) +
  geom_point() +
  scale_x_continuous(name = "Produsul intern brut pe cap de locuitor",
                     breaks = c(500, 5000, 15000, 25000,
                                50000, 75000, 100000, 125000),
                     minor_breaks = c(500, 2500, 7500)) +
  scale_colour_distiller(palette = "Set1",
                         breaks = c(1e7, 1e8, 5e8, 8e8, 1e9, 1.3e9),
                         name = "Populatia")

ggplot(gapminder_2018, aes(x = gdpPerCap, y = lifeExp, colour = pop, size = pop)) +
  geom_point() +
  scale_x_continuous(name = "Produsul intern brut pe cap de locuitor",
                     breaks = c(500, 5000, 15000, 25000,
                                50000, 75000, 100000, 125000),
                     minor_breaks = c(500, 2500, 7500)) +
  scale_colour_distiller(palette = "Set1",
                         breaks = c(1e7, 1e8, 5e8, 8e8, 1e9, 1.3e9),
                         name = "Populatia") +
  guides(colour = guide_colorbar(barwidth = 2,
                                 barheight = 10,
                                 ticks = FALSE),
         size = FALSE)
```

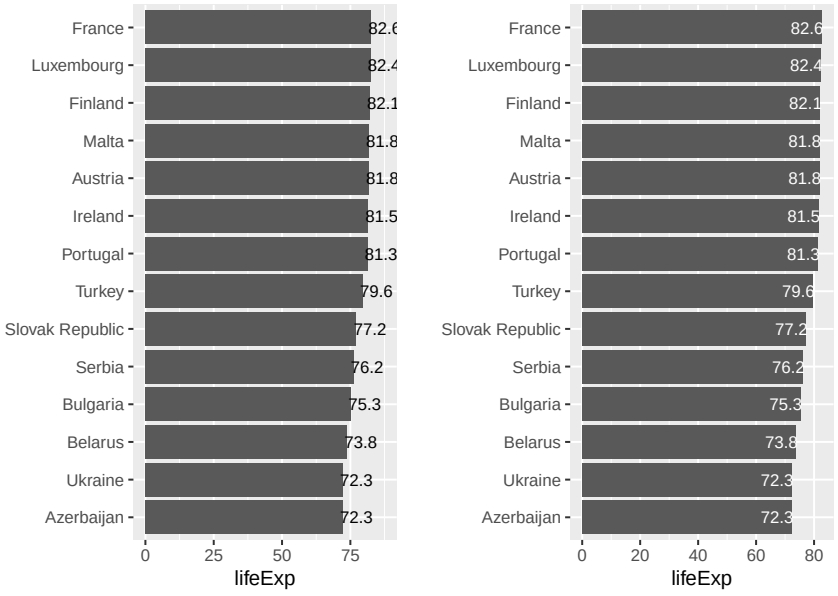


1.10.2 Adăugarea elementelor textuale la grafic

Sunt multe situații (e.g. în cazul unor prezentări, rapoarte, proiecte, etc.) în care pentru a face un grafic mai expresiv/ilustrativ este necesară adăugarea de elemente textuale explicative. Aceste elemente pot fi adăugate de cele mai multe ori folosind funcțiile `geom_text`, `geom_label`, `annotate` sau funcțiile corespunzătoare din pachetul `ggrepel`. Vom prezenta mai jos câteva exeple de grafice în care adăugarea de elemente textuale facilitează înțelegerea mesajului transmis de figură.

Pentru început vom folosi un barplot orizontal în care dorim să adăugăm marcaje în dreptul fiecărei bare:

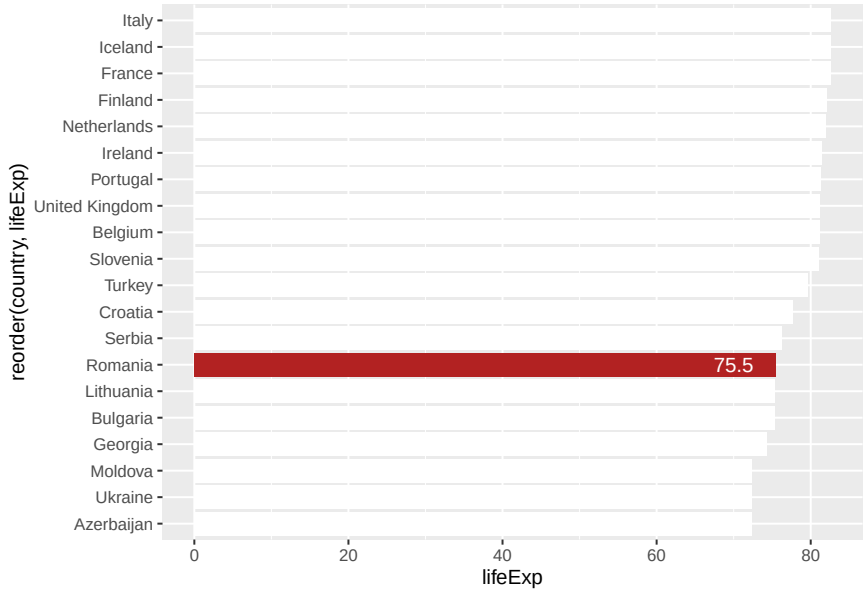
```
set.seed(1234)

baza = gapminder_2018 %>%
  filter(continent == "Europe") %>%
  sample_frac(0.3) %>%
  ggplot(aes(x = fct_reorder(country, lifeExp), y = lifeExp)) +
  geom_bar(stat = "identity") +
  coord_flip() +
  labs(x = "")

baza +
  geom_text(aes(label = lifeExp),
            size = 3, nudge_y = 5)

baza +
  geom_text(aes(label = lifeExp), size = 3,
            nudge_y = - 5, color = "white")
```



În cazul în care dorim să atragem atenția doar asupra unei bare atunci:

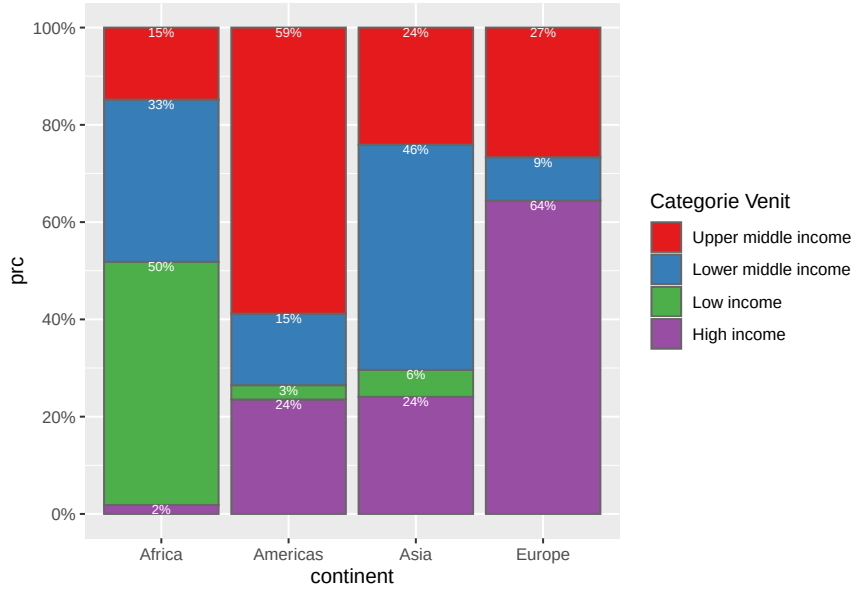
```
countries = c("Moldova", "Georgia", "Italy", "Croatia", "Slovenia",
              "Ukraine", "Azerbaijan", "Romania", "Turkey", "Macedonia",
              "Belgium", "France", "Finland", "Iceland", "Netherlands",
              "Lithuania", "Serbia", "Portugal", "Ireland", "Bulgaria",
              "United Kingdom")
```

```
baza_gap = gapminder_2018 %>%
  filter(country %in% countries) %>%
  mutate(ID = ifelse(country == "Romania", TRUE, FALSE))

ggplot(baza_gap, aes(x = reorder(country, lifeExp), y = lifeExp, fill = ID)) +
        geom_bar(stat = "identity") +
        coord_flip() +
        scale_fill_manual(values = c("white", "firebrick"),
                          guide = FALSE) +
        annotate("text", x = "Romania", y = 70, label = "75.5", color = "white")
```



Pentru a adăuga etichete la o diagramă cu bare proporțională (pentru care avem ajustată poziția la `fill`) trebuie să creăm o nouă variabilă care să stocheze locația (suma cumulativă a proporțiilor):

```
# pregatim setul de date
gapminder_2018_bf = gapminder_2018 %>%
  group_by(continent, wb_income) %>%
  # cate elemente am din fiecare wb_income pentru
  # fiecare continent in parte
  tally() %>%
  # grupez dupa continent
  group_by(continent) %>%
  # determin proportia
  mutate(prc = n / sum(n),
         label_y = cumsum(prc))


gapminder_2018_bf %>%
  ggplot(aes(x = continent, y = prc, fill = fct_rev(wb_income))) +
  geom_bar(stat = "identity", color = "grey40") +
  scale_y_continuous(breaks = seq(0, 1, by = .2), labels = scales::percent) +
  geom_text(aes(label = scales::percent(prc), y = label_y),
            vjust = 1, color = "white", size = 2.5) +
  scale_fill_manual(values = c("#E41A1C", "#377EB8", "#4DAF4A", "#984EA3")) +
  labs(fill = "Categorie Venit")
```



1.10.3 Teme grafice

Putem schimba întregul aspect al graficului prin modificarea *temei* (funcția `theme()`) acestuia. Temele reprezintă o modalitate consistentă de a customiza/personaliza elementele grafice ale unei figuri, elemente care nu fac referire la setul de date (titlu, etichetele axelor, mărimea și tipul fontului, culoarea de fundal, liniile de marcaj - gridlines, legendele, etc.).

Pentru a utiliza o temă trebuie să adăugăm unui obiect de tip `ggplot` o funcție de temă care are forma generală `theme_<nume temă>`. Pachetul `ggplot2` vine cu o serie de teme predefinite printre care menționăm: `theme_grey`, `theme_linedraw`, `theme_bw`, `theme_minimal`, `theme_void`, `theme_dark`, `theme_classic`, `theme_light`. Figura de mai jos prezintă fiecare dintre aceste teme pentru setul de date `gapminder_2018`:

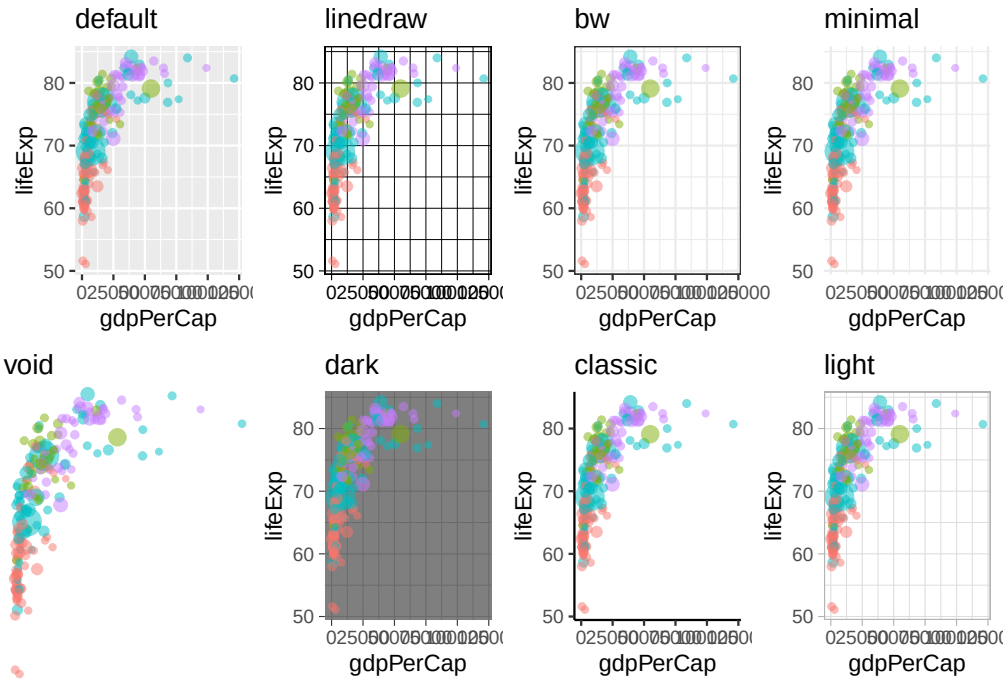


Fig. 4: Ilustrare a temelor predefinite

Pe lângă temele predefinite în `ggplot2` se pot folosi și teme din pachetul `ggthemes` dezvoltat de Jeffrey Arnold (pentru mai multe detalii vizitați site-ul ggthemes) care cuprinde mai mult de 20 de astfel de teme unele fiind reprezentări fidele ale tematicilor grafice folosite de reviste sau programe consacrate (de exemplu *Economist* sau *Excel*). Mai jos sunt prezentate doar patru dintre acestea:

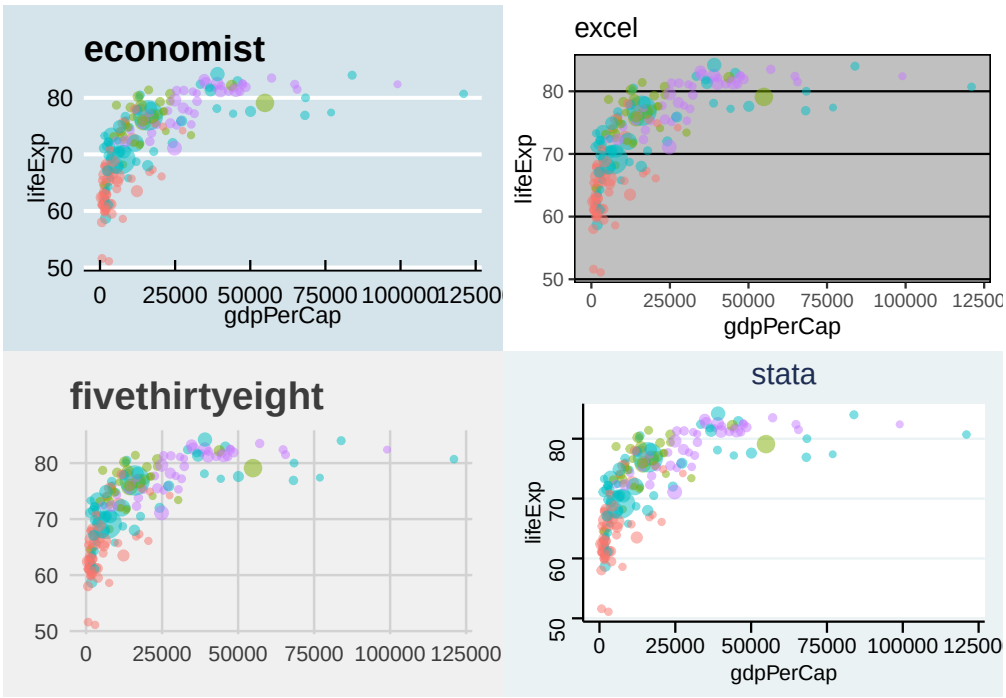


Fig. 5: Ilustrare a temelor din pachetul ggthemes

Pentru a modifica elementele unei temei vom folosi funcția `theme()`. Această funcție ne permite controlul (aproape total) asupra elementelor grafice care nu țin de setul de date dar prin intermediul cărora figura devine mai expresivă: marimea fontului, culoarea de fundal, fontul și culoarea etichetelor axelor și ale legendei, etc.

Structura generală este `theme([element al temei] = element_[nume functie asociată]([proprietati schimbate]))` unde

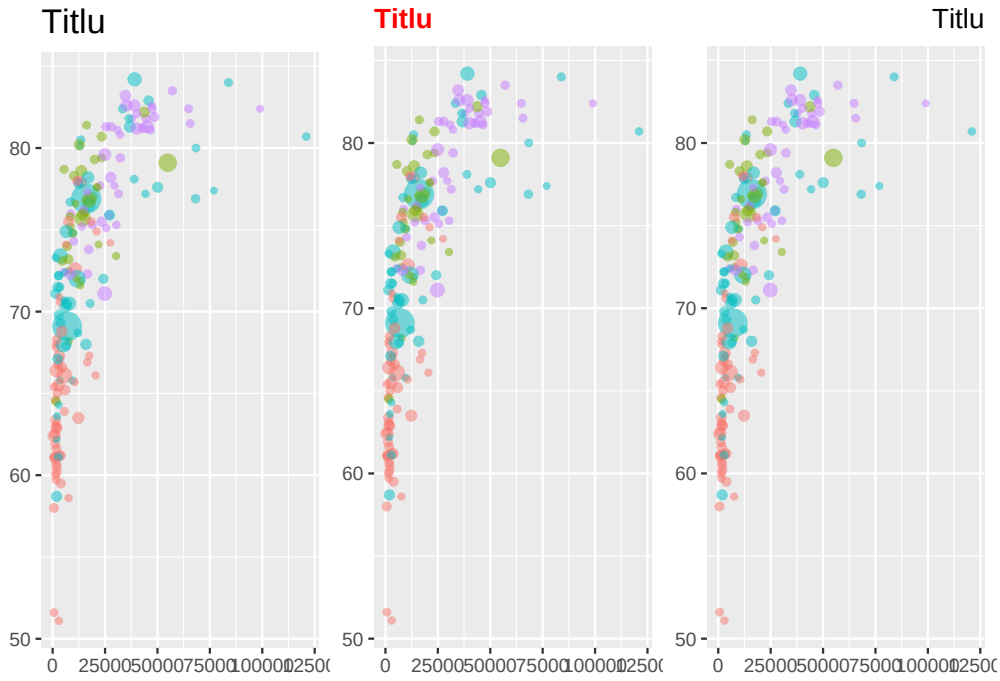
- `[element al temei]` face referire la un element grafic precum `plot.title` (controleaza modul de afișare a titlului graficului), `axis.ticks.x` (elemente de marcaj de pe axa x), `legend.key.height` (înălțimea elementelor din legendă) etc. Pentru a vizualiza toate elementele grafice ale unei teme ce pot fi modificate se poate consulta documentația https://ggplot2.tidyverse.org/reference/theme.html
- `element_[nume functie asociată]` reprezintă o funcție asociată elementelor temei care descrie caracteristicile vizuale ale elementului (toate elementele au asociate o astfel de funcție). Sunt patru astfel de funcții: `element_text`, `element_line`, `element_rect` și `element_blank`.
- `[proprietati schimbate]` reprezintă caracteristicile vizuale ce urmează să fie modificate, i.e. `font`, `family`, `linetype`, `colour`, `size`, etc.

Tabelul de mai jos prezintă o serie de caracteristici vizuale pentru fiecare funcție asociată elementelor unei teme grafice:

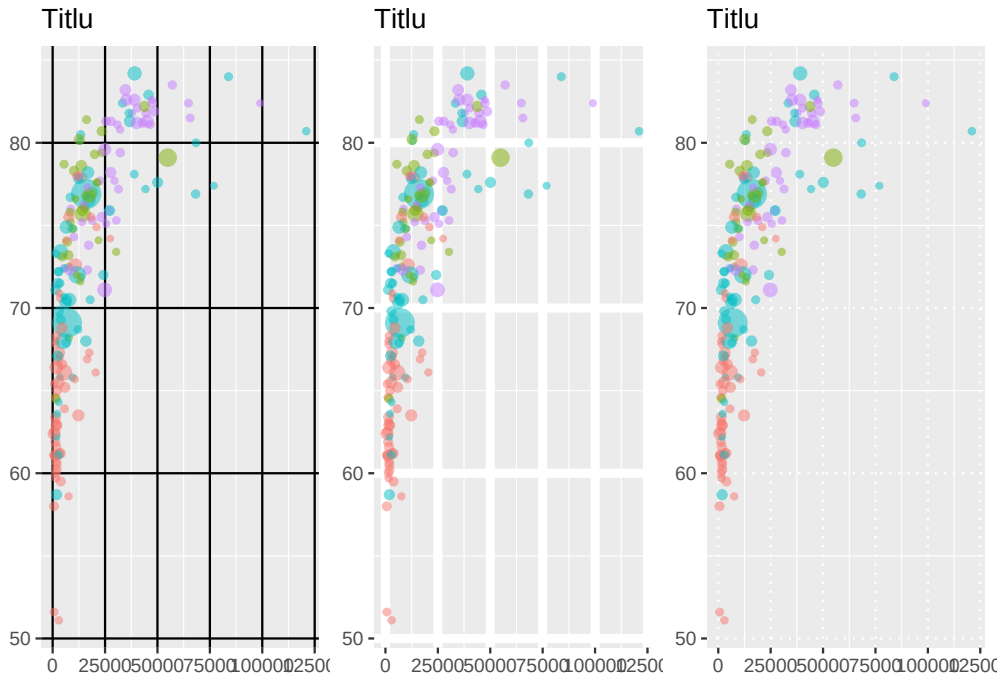
Funcție	Caracteristici
`element_text()`	`family`, `face`, `colour`, `size` (în points), `hjust`, `vjust`, `angle` (în grade), `lineheight`
`element_line()`	`colour`, `size`, `linetype`, `lineend`
`element_rect()`	`fill`, `colour`, `size`, `linetype`
`element_blank()`	

Exemple:

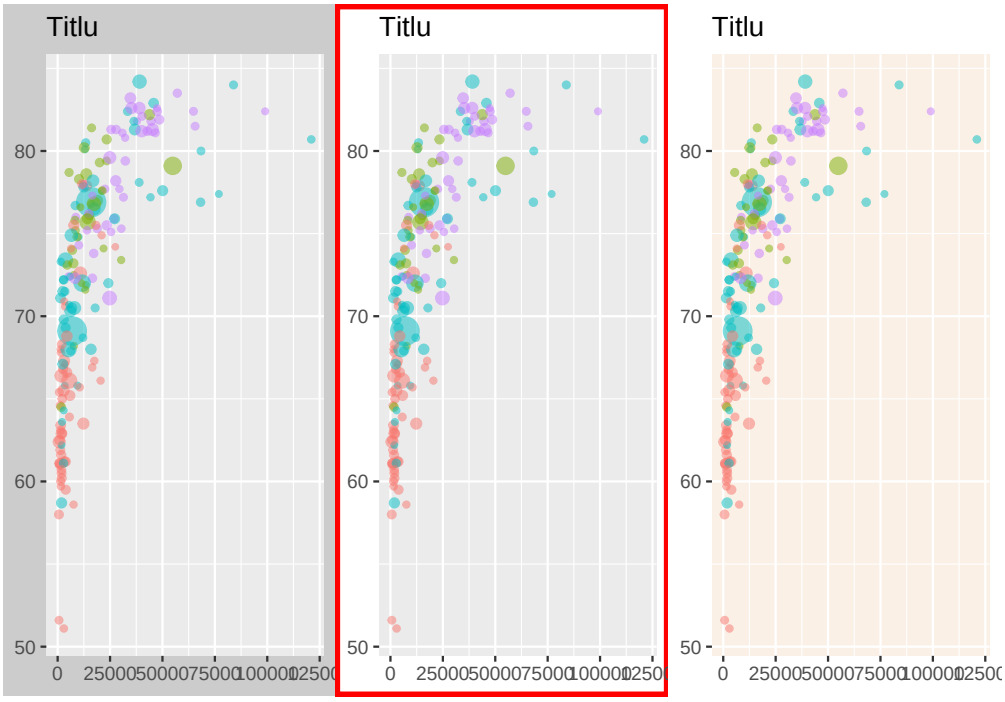
```
baza = ggplot(gapminder_2018) +
  geom_point(aes(x = gdpPerCap, y = lifeExp, color = continent, size = pop),
             alpha = 0.5, show.legend = FALSE)

baza <- baza + labs(title = "Titlu") + xlab(NULL) + ylab(NULL)
baza + theme(plot.title = element_text(size = 16))
baza + theme(plot.title = element_text(face = "bold", colour = "red"))
baza + theme(plot.title = element_text(hjust = 1))
```




```
baza + theme(panel.grid.major = element_line(colour = "black"))
baza + theme(panel.grid.major = element_line(size = 2))
baza + theme(panel.grid.major = element_line(linetype = "dotted"))
```



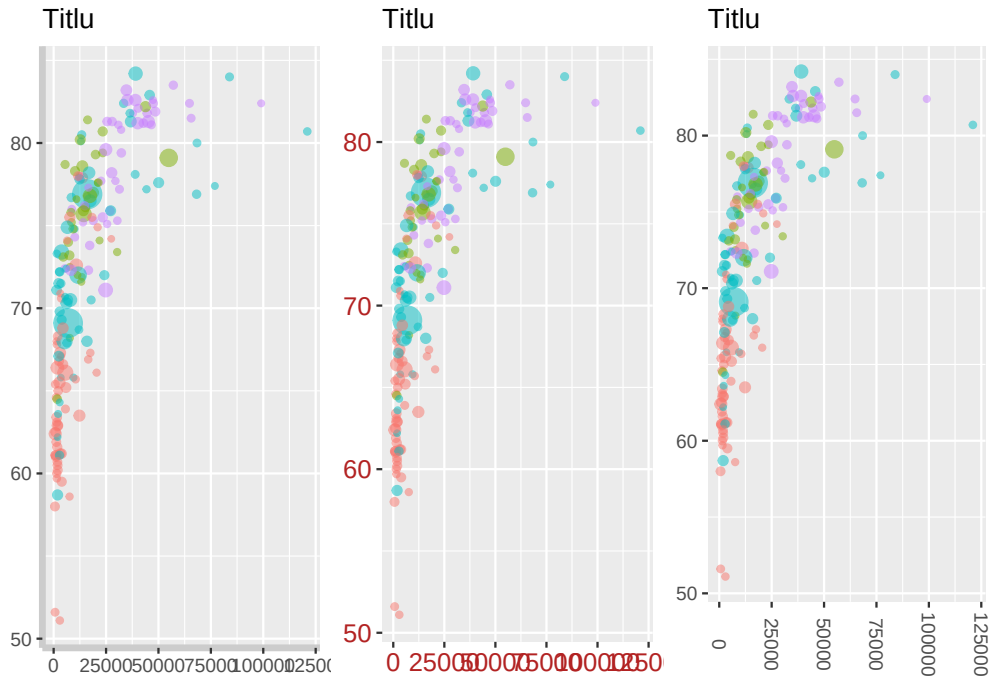

```
baza + theme(plot.background = element_rect(fill = "grey80", colour = NA))
baza + theme(plot.background = element_rect(colour = "red", size = 2))
baza + theme(panel.background = element_rect(fill = "linen"))
```



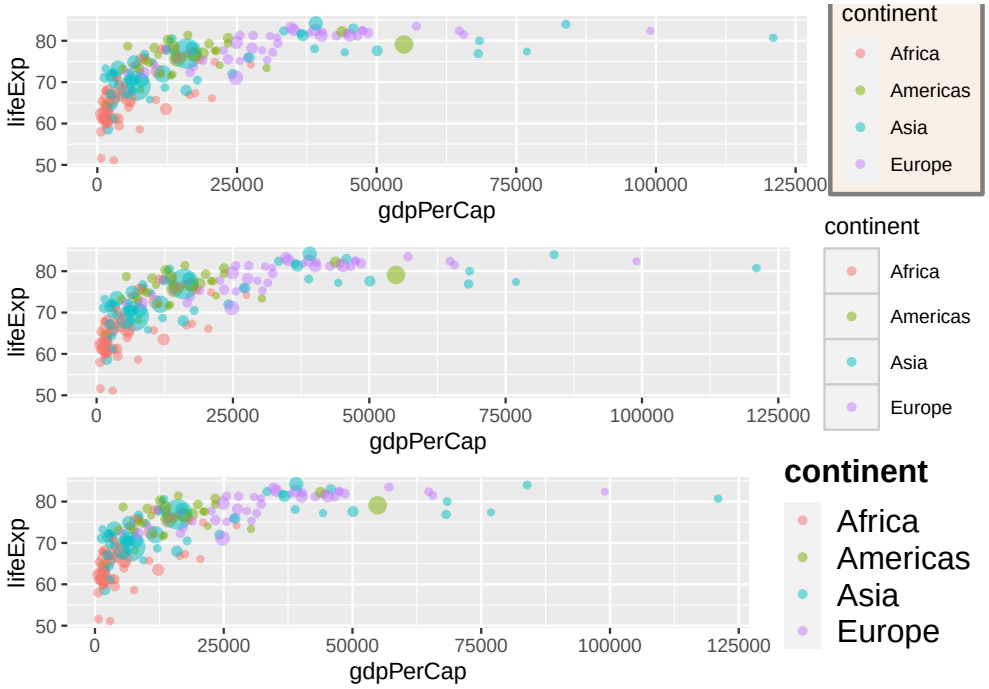
Mai jos regăsim o serie de elemente care permit controlul vizual al axelor și legendelor împreună cu funcțiile asociate acestora:

Obiect	Element	Setter	Description
axe	axis.line	`element_line()`	drepte paralele cu axele (nu se văd implicit)
	axis.text	`element_text()`	etichetele axelor (x și y)
	axis.text.x	`element_text()`	etichetele axei x
	axis.text.y	`element_text()`	etichetele axei y
	axis.title	`element_text()`	titlurile axelor
	axis.title.x	`element_text()`	tiltlul axei x
	axis.title.y	`element_text()`	tiltlul axei y
	axis.ticks	`element_line()`	elementele de marcaj ale axelor
	axis.ticks.length	`unit()`	lungimea elementelor de marcaj
legendă	legend.background	`element_rect()`	fundalul legendei
	legend.key	`element_rect()`	fundalul elementelor legendei
	legend.key.size	`unit()`	marimea elementelor legendei
	legend.key.height	`unit()`	înălțimea elementelor legendei
	legend.key.width	`unit()`	lățimea elementelor legendei
	legend.margin	`unit()`	marginile legendei
	legend.text	`element_text()`	etichetele legendei
	legend.text.align	0–1	alinierea etichetelor legendei (0 = dreapta, 1 = stânga)
	legend.title	`element_text()`	titlul legendei
	legend.title.align	0–1	alinierea titlului legendei (0 = dreapta, 1 = stânga)

```
baza + theme(axis.line = element_line(colour = "grey80", size = 1.5))
baza + theme(axis.text = element_text(color = "firebrick", size = 12))
baza + theme(axis.text.x = element_text(angle = -90, vjust = 0.5))
```




```
baza2 = ggplot(gapminder_2018) +
  geom_point(aes(x = gdpPerCap, y = lifeExp, color = continent, size = pop),
             alpha = 0.5) +
  guides(size = "none")

baza2 + theme(
  legend.background = element_rect(
    fill = "linen",
    colour = "grey50",
    size = 1
  )
)
baza2 + theme(
  legend.key = element_rect(color = "grey80"),
  legend.key.width = unit(0.9, "cm"),
  legend.key.height = unit(0.75, "cm")
)
baza2 + theme(
  legend.text = element_text(size = 15),
  legend.title = element_text(size = 15, face = "bold")
)
```



Pentru a mai multe detalii și exemple privind elementele grafice ale unei teme ce pot fi modificate se poate consulta documentația https://ggplot2.tidyverse.org/reference/theme.html.

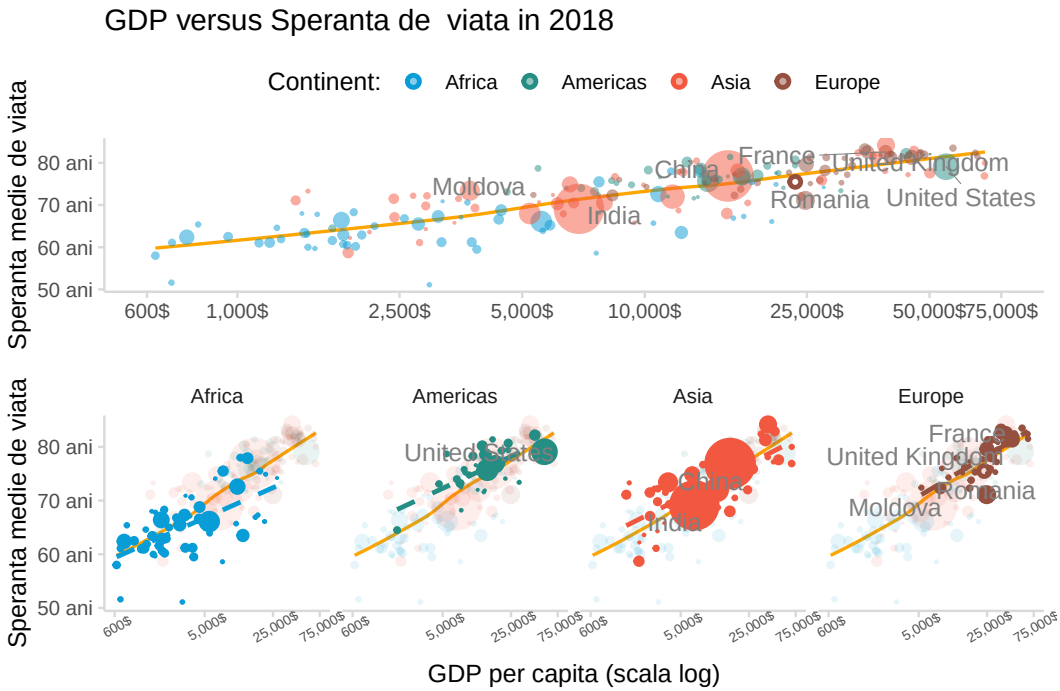
1.11 Exemple folosind setul de date `gapminder`

Noțiunile prezentate în acest capitol, puse cap la cap, permit generarea figurii de la începutul capitolului. Mai jos este prezentat codul folosit pentru generarea figurii (pentru anul 2018):

```
ggplot(gapminder_2018) +
  # adaugam curba de regresie loess
  geom_smooth(aes(x = gdpPerCap, y = lifeExp),
              se = FALSE, method = "loess",
              color = "orange", size = 0.7, alpha = 0.7) +
  # adaugam puncte fara Romania
  geom_point(data = gapminder_2018 %>% filter(country != "Romania"),
             aes(x = gdpPerCap, y = lifeExp, color = continent, size = pop),
             alpha = 0.5) +
  # adaugam Romania si o facem de forma diferita - cerc gol
  geom_point(data = gapminder_2018 %>% filter(country == "Romania"),
             aes(x = gdpPerCap, y = lifeExp, color = continent, size = pop),
             shape = 1,
             stroke = 1.5)+
  # adaugam text pentru o serie de state (ggrepel)
  geom_text_repel(aes(x = gdpPerCap, y = lifeExp, label = country),
            color = "grey50",
            segment.size = 0.2,
            data = gapminder_2018 %>%
              filter(pop > 1e9 | country %in% c("Moldova",
                                                "United States",
                                                "Romania",
                                                "United Kingdom",
```

```
                                                "France"))) +
# schimbam scala pe x in scala logaritmica
scale_x_log10(limits = c(600, 75000),
              breaks = c(600, 1000, 2500, 5000, 10000,
                         25000, 50000, 75000),
              label = scales::dollar_format(prefix = "", suffix = "$", accuracy = 1)) +
# adaugam ani la scala de pe y
scale_y_continuous(label = function(x) {return(paste(x, "ani"))}) +
# schimbam etichetele si adaugam titlu
labs(title = "GDP versus Speranta de  viata in 2018",
     caption = "Sursa: https://www.gapminder.org/",
     x = "GDP per capita (scala log)",
     y = "Speranta medie de viata",
     size = "Populatia",
     color = "Continent") +
# schimbam marimea scalei
scale_size(range = c(0.1, 10),
           # stergem legenda pentru size
           guide = "none") +
# adaugam culori customizate pentru continente
scale_color_manual(name = "",
                   values = c("#099DD7",
                              "#248E84",
                              "#F2583F",
                              "#96503F"),
                   guide = guide_legend(nrow = 1, order=1)) +
# schimbam tema
theme_minimal() +
# plasam legenda sus si scoatem gridul
theme(legend.position = "top",
      axis.line = element_line(color = "grey85"),
      axis.ticks = element_line(color = "grey85"),
      panel.grid.major = element_blank(),
      panel.grid.minor = element_blank())
```

Cu un pic mai mult efort se poate construi figura de mai jos care în plus evidențiază pentru fiecare continent în parte tendința (dreapta de regresie) corespunzătoare:



Sursa: https://www.gapminder.org/

Pentru cel de-al doilea grafic avem

```
countries = c("Romania", "France", "Germany", "United Kingdom", "Italy", "Greece",
              "United States", "Canada", "Brazil", "Mexico",
              "China", "India", "Vietnam",
              "Ethiopia", "South Africa", "Nigeria",
              "Australia", "New Zealand")

gapminder_life_exp_diff <- gapminder_all %>%
  mutate(year = as.integer(year),
         continent = four_regions) %>%
  # filtram anii de start si de final
  filter(year == 2000 | year == 2018) %>%
  # ne asiguram ca datele sunt aranjate astfel ca anul 2000 sa fie inaintea lui 2018
  arrange(country, year) %>%
  # pentru fiecare tara adaugam variabila care ne da diferenta sperantei de viata
  group_by(country) %>%
  mutate(lifeExp_diff = lifeExp[2] - lifeExp[1]) %>%
  ungroup() %>%
  # aranjam in ordinea diferentelor cele mai mari
  arrange(lifeExp_diff) %>%
  # restrangem la tarile selectate
  filter(country %in% countries) %>%
  select(country, year, continent, lifeExp, lifeExp_diff)


gapminder_life_exp_diff %>%
  mutate(country = fct_inorder(country)) %>%
  # pentru fiecare tara definim minimul si maximul sperantei de viata
  group_by(country) %>%
  mutate(max_lifeExp = max(lifeExp),
```

```
        min_lifeExp = min(lifeExp)) %>%
  ungroup() %>%
  ggplot() +
  # trasam segmentele
  geom_segment(aes(x = min_lifeExp, xend = max_lifeExp,
                   y = country, yend = country,
                   col = continent), alpha = 0.5, size = 5) +
  # adaugam punctele de capat
  geom_point(aes(x = lifeExp, y = country, col = continent), size = 8,
             shape = 21, fill = "white", stroke = 2) +
  # adaugam elementele textuale
  geom_text(aes(x = min_lifeExp + 0.47, y = country,
                label = paste(country, " ", round(min_lifeExp))),
            col = "grey50", hjust = "right") +
  geom_text(aes(x = max_lifeExp - 0.4, y = country,
                label = round(max_lifeExp)),
            col = "grey50", hjust = "left") +
  # delimitam axa x
  scale_x_continuous(limits = c(45, 85)) +
  # alegem culorile
  scale_color_manual(values = c("#099DD7",
                                "#248E84",
                                "#F2583F",
                                "#96503F")) +
  # stabilim titlul si axele
  labs(title = "Schimbarea speranței de viață",
       subtitle = "Între anii 2000 și 2018",
       x = "Speranța de viață (în 2000 și 2018)",
       y = NULL,
       col = "Continent: ") +
  # folosim tema clasica
  theme_classic() +
  # eliminam axele si pozitionam legenda
  theme(legend.position = "top",
        axis.line = element_blank(),
        axis.ticks = element_blank(),
        axis.text = element_blank())
```

Mai mult dacă dorim să ilustrăm evoluția speranței de viață în Europa evidențiind pentru fiecare țară în parte parcurul atunci putem obține un grafic de forma

```
# setul de background
gap_bg = gapminder_all %>%
  mutate(continent = four_regions,
         year = as.integer(year)) %>%
  filter(continent == "Europe",
         year > 1900) %>%
  rename(country2 = country)

# capetele
gap_bg_endpoints = gap_bg %>%
  filter(year == 2018) %>%
  rename(country = country2)
```

```
## tarile
gap_bg_countries = gap_bg %>%
  group_by(country2) %>%
  filter(year == min(year)) %>%
  ungroup() %>%
  mutate(year = 1900,
         country = country2,
         lifeExp = max(gap_bg$lifeExp, na.rm = TRUE) - 2)

# constructia graficului
gapminder_all %>%
  mutate(continent = four_regions,
         year = as.integer(year)) %>%
  filter(continent == "Europe",
         year > 1900) %>%
ggplot(aes(x = year, y = lifeExp)) +
  # trasarea curbelore de background
  geom_line(data = gap_bg,
            aes(x = year, y = lifeExp, group = country2),
            size = 0.15, color = "gray80", alpha = 0.5)+
  # trasarea liniilor evidentiate
  geom_line(aes(group = country),
            color = "royalblue",
            lineend = "round") +
  # adaugarea punctelor la final
  geom_point(data = gap_bg_endpoints,
             aes(x = year, y = lifeExp),
             size = 1.1,
             shape = 21,
             color = "royalblue",
             fill = "royalblue") +
  # adaugarea tarilor - stanga sus
  geom_text(data = gap_bg_countries,
            mapping = aes(label = country),
            vjust = "inward",
            hjust = "inward",
            fontface = "bold",
            size = 2.1) +
  # fatetarea dupa country reordonat
  facet_wrap(~ reorder(country, - lifeExp), ncol = 5)+
  # redenumirea axelor
  labs(x = "Perioada 1900 - 2018",
       y = "Speranta de viata",
       title = "Evolutia sperantei de viata in Europa",
       caption = "Sursa: https://www.gapminder.org/") +
  # customizarea temei
  theme_classic()+
  theme(plot.title = element_text(size = rel(1), face = "bold"),
        plot.caption = element_text(size = rel(1)),
        # scoatem titlurile fatetelor
        strip.text = element_blank(),
        panel.spacing.x = unit(-0.05, "lines"),
        panel.spacing.y = unit(0.3, "lines"),
```

```
        axis.text.y = element_text(size = rel(0.5)),
        axis.title.x = element_text(size = rel(1)),
        axis.title.y = element_text(size = rel(1)),
        axis.text.x = element_text(size = rel(0.5)),
        legend.text = element_text(size = rel(1)))
```

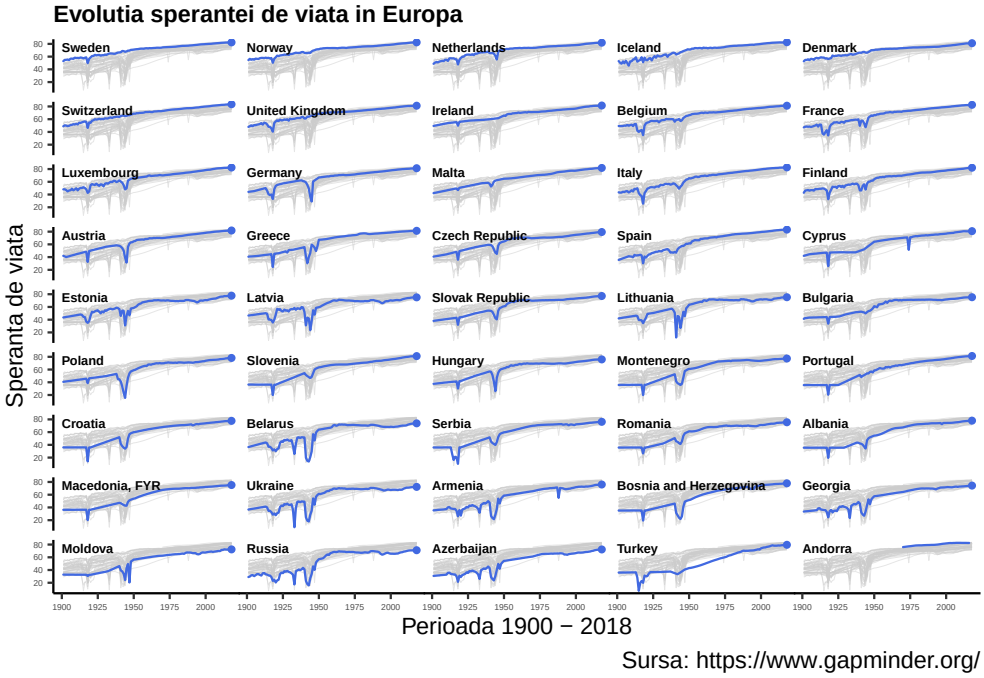


Fig. 6: Evolutia sperantei de viata in Europa dupa anul 1900

Atunci când dorim să salvăm un grafic putem folosi funcția `ggsave()` care prin comportamentul de bază salvează ultimul grafic generat. Funcția permite specificarea graficului pe care dorim să îl salvăm (`plot =`), locația și numele fișierului salvat (`filename =`), dimensiunea (`width =`, `height =`) precum și tipul acestuia (`png`, `jpeg`, `tiff`, `pdf`, `tex`, etc.).

De exemplu, pentru a salva figura generată de codul de mai sus (figura de la începutul capitolului) într-un fișier *pdf* în format *landscape* putem scrie

```
ggsave(filename = "grafic.pdf",
       width = 11, height = 8.5)
```

ținând cont că acesta este ultimul grafic generat, în caz contrar trebuie să specificăm pe care grafic dorim să îl salvăm. Astfel, dacă figura este atribuită elementului `p` (`p = ggplot(gapminder_2018) + ...`) atunci pentru a salva scriem

```
ggsave(filename = "grafic.pdf",
       plot = p,
       width = 11, height = 8.5)
```

Ex. 1.1

Încercați să recreați grafice similare cu cel de la începutul capitolului (sau cel de mai sus) pentru alte seturi de date ce pot fi descărcate de pe platforma www.gapminder.org/data/.

Referințe

Robbins, Naomi. 2013. *Creating More Effective Graphs*. First. New York, NY: Chart House.

Wickham, Hadley. 2016. *Ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. https://ggplot2.tidyverse.org.

Wilkinson, Leland. 2005. *The Grammar of Graphics (Statistics and Computing)*. First. Secaucus, NJ: Springer-Verlag.