# Sessió 10

# Introducció a R modern amb Tidyverse

## *10.1 Què és Tidyverse?*

*Tidyverse* és una col·lecció de paquets d'R dissenyats per a la ciència de dades que utilitzen la mateixa filosofia de disseny, gramàtica i estructura de dades. Estan dissenyats per treballar junts amb naturalitat.

Instal·lant el paquet amb `install.packages('tidyverse')` i carregant-lo amb `library(tidyverse)` es carreguen els paquets `ggplot2`, `tibble`, `tidyr`, `readr`, `purrr`, `dplyr`, `stringr` i `forcats`, que són considerats el cor de `Tidyverse` perquè són els que s'utilitzen amb més freqüència.


El nom *Tidyverse* ve de la paraula anglesa *tidy* (ordenar) i pretén ser una forma més fàcil i ràpida de treballar amb R.

Els paquets del Tidyverse canvien amb força freqüència. Es pot veure si existeixen actualitzacions disponibles i opcionalment instal·lar-les executant `tidyverse_update()`.

- **ggplot2** és un sistema de crear gràfics amb una gramàtica especial: s'especifica el data frame, es diu quines variables es volen representar (a `aesthetics`), quin tipus de gràfic, i altres detalls. És el més utilitzat de `tidyverse` i es pot fer servir de manera independent de la resta de paquets. Malgrat que la sintaxi resulta complicada i diferent de la dels gràfics de R que hem vist fins ara, fa representacions de les dades molt boniques, amb una sintaxi unificada.
- **dplyr** és un paquet per manipular data frames, donant instruccions per fer transformacions i subsets en un data.frame en un llenguatge més "humà":
  - `mutate()` afegeix al data.frame noves variables que són functions de variables existents.
  - `select()` selecciona unes quantes variables del data.frame, a partir dels seus noms.
  - `filter()` selecciona uns quants casos a partir de condicions
  - `summarise()` fa resums, però permet fer resums per grups
  - `arrange()` canvia l'ordre dels casos d'un data.frame.
  - Les funcions `inner_dplyr`, `left_dplyr`, `right_dplyr` i `full_dplyr` per ajuntar data.frames.
- **tibble**, ens permet visualitzar els data.frames donant més informació de les variables.

- L'objectiu de **tidyr** és ajudar a l'usuari a crear dades endreçades. Per exemple conté la funció `drop_na()` que elimina d'un data.frame els casos que tenen algun valor perdut. També `replace_na` que permet substituir els valors perduts de les diferents variables per valors personalitzats, de cop.
- L'objectiu de **readr** és donar una manera amable de llegir fitxers externs amb dades rectangulars.
- **purrr** té funcions per treballar amb llistes (transformar-les, filtrar-les, resumir-les,...).
- **stringr** serveix per treballar amb variables tipus caràcter. En molts casos després d'una importanció cal netejar, escurçar, simplificar aquest tipus de variables.
- **forcats** conté eines útils per resoldre problemes que solen apareixen al treballar amb factors.

A banda d'aquest cor de paquets, n'hi ha molts d'altres de la col·lecció *Tidyverse*


**Atenció als usuaris de Linux:** Quan instal·lem Tidyverse en linux, sovint dona problemes a causa del Java. Dos paquets que són necessaris per instal·lar Tidyverse, relacionats amb el Java de l'ordinador són `rjava` i `rJSon`. La solució, si la instal·lació de Tidyverse no funciona, és instal·lar els paquets principals per separat (paquets `ggplot2`, `tibble`, `tidyr`, `readr`, `purrr`, `dplyr`, `stringr` i `forcats`)

Nota del Desembre de 2021: A la pàgina https://blog.zenggyu.com/en/post/2018-01-29/installing-r-r-packages-e-g-tidyverse-and-rstudio-on-ubuntu-linux/ explica com solucionar en linux els problemes d'instal·lació de Tidyverse. Cal instal·lar abans paquets del sistema linux (que no són de R) des de bash (libcurl4-openssl-dev, libssl-dev, libxml2-dev).

```
# per instal.lar els paquets de cop
instal.packages(c('ggplot2','tibble','tidyr','readr',
            'purrr','dplyr','stringr'))
# per carregar-los de cop
sapply(c('ggplot2','tibble','tidyr','readr','purrr','dplyr',
          'stringr'),library,character.only = TRUE)
install.packages("ggplot2",dependencies=TRUE)  # si no us va en linux
```

## 10.2 Les pipes de Tidyverse

Tidyverse treballa amb "pipes" o canonades (s'escriuren amb el codi `%>%`) que faciliten la programació. Les pipes encadenen la sortida d'una comanda amb l'entrada de la comanda següent. Vegem-ho amb exemples

```
data(mtcars)
# volem fer un resum de les variables mpg i disp del fitxer mtcars
# però per a casos amb am==0
# amb R clàssic
sub.mtcars<-subset(mtcars,am==0,select=c('mpg','disp'))
summary(sub.mtcars)
# amb la filosofia de les pipes
```

```
subset(mtcars,am==0,select=c('mpg','disp')) %>% summary() # també:
mtcars %>% subset(.,am==0,select=c('mpg','disp')) %>% summary()
# el punt substitueix el que hi ha a l'equerra de %>%
```

Altres exemples de pipes:

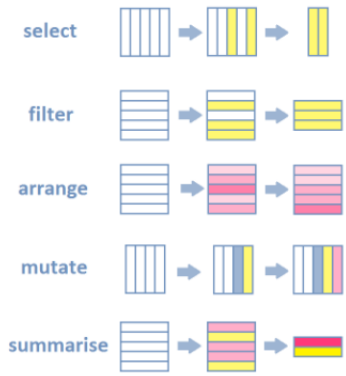
```
 iris %>%
  subset(Sepal.Length > 5) %>%
  aggregate(.~Species,., mean)
  # equival a dos passos:
sub.iris<-subset(iris,Sepal.Length>5)
aggregate(.~Species,data=sub.iris,mean)  # fixem-nos que
  #el primer punt s'ha de posar igual, no té a veure amb la pipa

rnorm(200) %>%
   round(digits=1) %>%
   sort() %>%
   matrix(.,ncol=10)

# la instrucció %T>% és per continuar la pipa però donar la sortida d'una
#    instrucció intermitja (la de després de %T>%)
rnorm(200) %>%
  matrix(ncol = 2) %T>%
  plot %>%
  colSums
```

**Treballar amb les pipes en general, sense la resta de Tidyverse.** El paquet `magrittr` permet utilitzar la instrucció `%>%` per fer pipes. També les podem utilitzar carregant només el paquet `tidyr` del Tidyverse.

**Pràctica:** Escriu el codi del final de la pàgina 95 amb pipes. Només cal crear abans `var.nums` i carrgar lattice perquè funcioni el `levelplot`. El `df` a què es refereix és el de datos empleados de la pràctica anterior

## 10.3 El paqquet dplyr

Les principals funcions per treballar de forma ràpida i fàcil amb Tidyverse són:




Es poden utilitzar altres funcions de `tidyr` per fer el subset (la funció `select` per seleccionar columnes i la funció `filter` per filtrar casos) tal com hem fet al primer exemple:

```
mtcars %>%
  filter(am==0) %>%
```

```
  select(c('mpg','disp')) %>%
  summary()
```

A la funció `filter`, separats per comes podem afegir més condicions. Farem la intersecció de totes. La sortida de la funció `filter` és un data frame.

```
load("GastoHogaresEduc.RData") # de la pràctica 9
str(fichero_salida)  # les variables MCL i C11A tenen NA
fichero_salida<-fichero_salida %>% filter(!is.na(MCL),!is.na(C11A))
```

Una altra funció és `mutate` que ens pot anar bé per transformar una variable en factor

```
fichero_salida<-fichero_salida %>%
         mutate(SEXO=factor(SEXO,labels=c("dona","home")))
# podíem haver escrit, amb magrittr carregat
fichero_salida %<>% mutate(SEXO=factor(SEXO,labels=c("dona","home")))
```

**Atenció:** Quan es fan pipes molt llargues, que ocupen més d'una línia, la instrucció `%>%` ha d'anar al final de la línea o dins d'una línea, mai a l'inici.

## 10.4 Altres recursos disponibles. Els cheatsheets.

Per aprendre com emprar els paquets a la pràctica, els fulls de referència ("xuletes" o "cheatsheets") que trobareu a la següent adreça són un bon recurs: `https://rstudio.com/resources/cheatsheets/`

També podeu consultar el web `https://www.tidyverse.org/` i per aprendre el Tidyverse més a fons teniu el llibre `https://es.r4ds.hadley.nz/`, a banda de les consultes habituals que es poden fer a R.

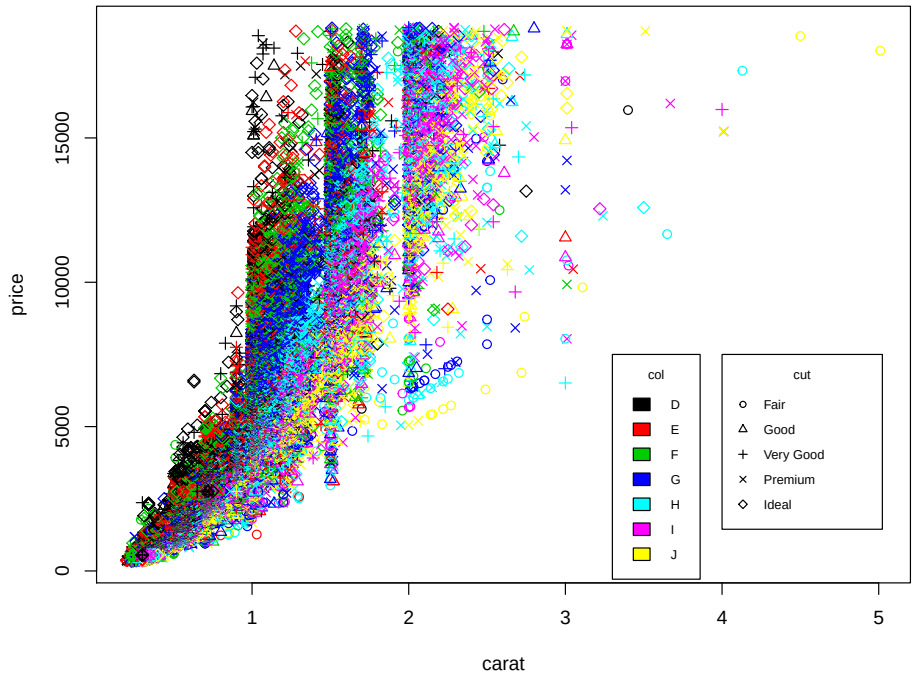
## 10.5 El paquet ggplot2

El paquet `ggplot2` queda instal·lat i carregat quan s'instal·la i es carrega el grup de paquets del Tidyverse però també es pot carregar de manera independent.

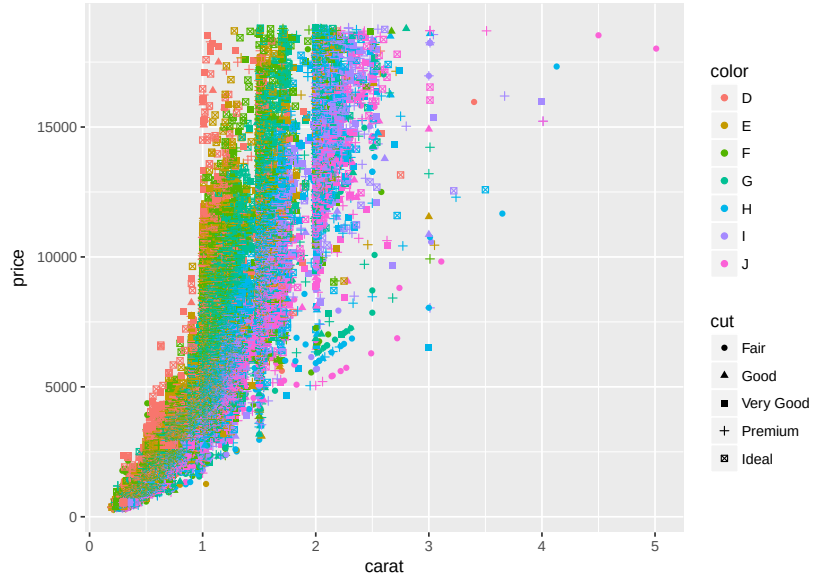
El paquet `ggplot2` ens permet fer de forma fàcil gràfics més bonics i la seva funció principal és `ggplot`. En aquest cas les diferents funcions que es relacionen amb `ggplot` es concatenen amb el símbol "+" i no pas amb les "pipes" `%>%`.

Vegem la diferència entre la funció plot d'R basic i la funció ggplot del Tidyverse amb un exemple:

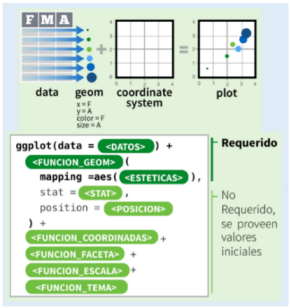
```
library(ggplot2)
data(diamonds) # és un conjunt de dades del paquet ggplot2

# gràfic amb R bàsic
plot(diamonds$carat,diamonds$price,col=diamonds$color,
     pch=as.numeric(diamonds$cut),xlab='carat',ylab="price")
legend(3.3,7500,legend=levels(diamonds$color),fill=1:7,
     title="col",cex=0.7)
legend(4,7500,legend=levels(diamonds$cut),pch=1:5,cex=0.7,
     title = "cut")
```




```
library(ggplot2)
# gràfic amb ggplot2
ggplot(diamonds,aes(carat,price,col=color,shape=cut))+geom_point()
```



### Què cal tenir en compte per fer un gràfic amb ggplot2?



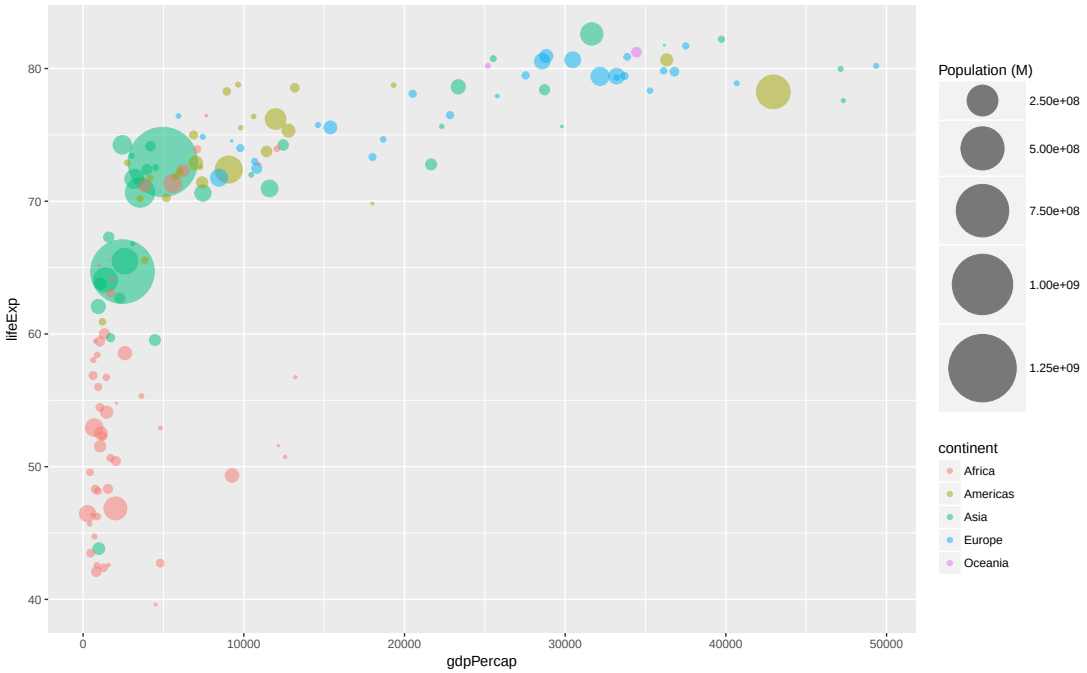

Data que s'ha de visualitzar
Geometria dels objectes que es mostren al gràfic
Aesthetic: estètica de les dades als components visuals
Statistics: transforma les dades de camí a la visualització
Coordinates: organitza la localització dels objectes geomètrics
Facets: agrupacions en subgràfics
Scales: defineix els rangs de valors a mostrar per estètica
Themes: aplica conjunts d'estils

Si voleu veure un exemple pas a pas: `https://pkg.garrickadenbuie.com/gentle-ggplot2/#39`

**Exemple:**

```
library(ggplot2)
library(dplyr) # a l'aula l'haureu d'instal.lar
library(gapminder)
data <- gapminder %>% filter(year=='2007') %>% dplyr::select(-year)
data %>%
  arrange(desc(pop)) %>%
  mutate(country = factor(country, country)) %>%  # converteix country a factor
  ggplot(aes(x=gdpPercap, y=lifeExp, size=pop, color=continent)) +
    geom_point(alpha=0.5) +
    scale_size(range = c(.1, 20), name='Population (M)')
```

podeu veure'n ampliacions a `https://www.r-graph-gallery.com/320-the-basis-of-bubble-plot.html`




Al web `https://www.r-graph-gallery.com/` pots trobar els scripts per diferents gràfics que només hauràs d'adaptar a les teves dades.

A continuació mostrem alguns exemples, extrets de `https://es.r4ds.hadley.nz` (recomanem els capítols 3, 7 i 28).

Executa el codi i observa el gràfic que obtens:

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

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

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

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

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

ggplot(data = mpg) + aes(x=displ,y=hwy)+
  geom_point(col=4)+
  geom_smooth(col=2)

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

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

ggplot(data=mpg)+
  aes(x=displ,y=hwy,col=drv)+
  geom_point()+
  geom_smooth(mapping=aes(linetype=drv),col=1)

ggplot(data=mpg)+
  aes(x=displ,y=hwy,col=drv)+
  geom_point()+
  geom_smooth(mapping=aes(col=drv),method="lm")+
  scale_color_manual(values =c(3,4,6))
```

Diagrames de barres

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

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

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

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

Altres diagrames

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

# boxplots
data(ToothGrowth)
# convertim la variable dose en factor amb mutate del paquet dyplr
ToothGrowth<-ToothGrowth %>%
  mutate(dose=as.factor(dose))

p <- ggplot(ToothGrowth, aes(x=dose, y=len)) +
  geom_boxplot()
p
# Rotem el gràfic
p + coord_flip()
ggplot(ToothGrowth, aes(x=dose, y=len)) +
  geom_boxplot(notch=TRUE)
ggplot(ToothGrowth, aes(x=dose, y=len)) +
  geom_boxplot(outlier.colour='red', outlier.shape=8,
               outlier.size=4)
p + geom_dotplot(binaxis='y', stackdir='center', dotsize=1)
p + geom_jitter(shape=16, position=position_jitter(0.2),size=3)
```

Afegir títol i modificar límits eixos

```
ggplot(data = diamonds) +
  stat_summary(
    mapping = aes(x = cut, y = depth),
    fun.ymin = min,
    fun.ymax = max,
    fun.y = median
  )+
  coord_cartesian(ylim = c(0,80))+
  labs(x="tall",title="Max, min i mediana",y="profunditat")
```

Representar per grups

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

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

### Guardar gràfics amb ggsave()

Els arguments més habituals de la funció `ggsave()` per guardar gràfics són el nom, el format que volem per al fitxer, l'alçada i l'amplada (pot resultar molt útil en un treball, per uniformitzar els gràfics).

```
ggsave('nom_fitxer.pdf', width = 20, height = 15, units = 'cm')
ggsave('nom_fitxer.png', width = 20, height = 15, units = 'cm')
```

**Pràctica:** Importa el fitxer `Notes_assign_comuns_BAT.xlsx` que trobaràs al moodle.
Al web `https://www.r-graph-gallery.com/` trobaràs l'script per fer un gràfic d'aranya.
a) Fes un gràfic d'aranya amb les notes mitjanes de les assignatures.
b) "Tuneja" el gràfic per a que es vegi més bonic: canvia la mida de les etiquetes per a que quedi millor, pinta l'àrea interior d'un blau claret, fes les línies del gràfic en gris, ...
c) Contesta:

1. Quina assignatura té la millor nota?
2. I la pitjor?

d) Fes un gràfic amb les notes mitjanes de les diferents assignatures per als nois, les noies i el total.
Posa la línia dels homes en verd, la de les dones en blau i la del total en negre i més gruixuda.
Afegeix també la llegenda.
En quines assignatures són millor les noies?
e) Fes un gràfic amb les notes mitjanes per als diferents tipus de batxillerat, amb una línia més gruixuda i vermella que marqui la nota de l'aprovat (5). Analitza el gràfic.