# Introducció a ggplot2 i ggmap

PID_00249220

Carlos J. Gil Bellosta

Temps mínim de dedicació recomanat: 2 hores

Universitat Oberta de Catalunya

# Índex

# 1. Introducció a ggplot2

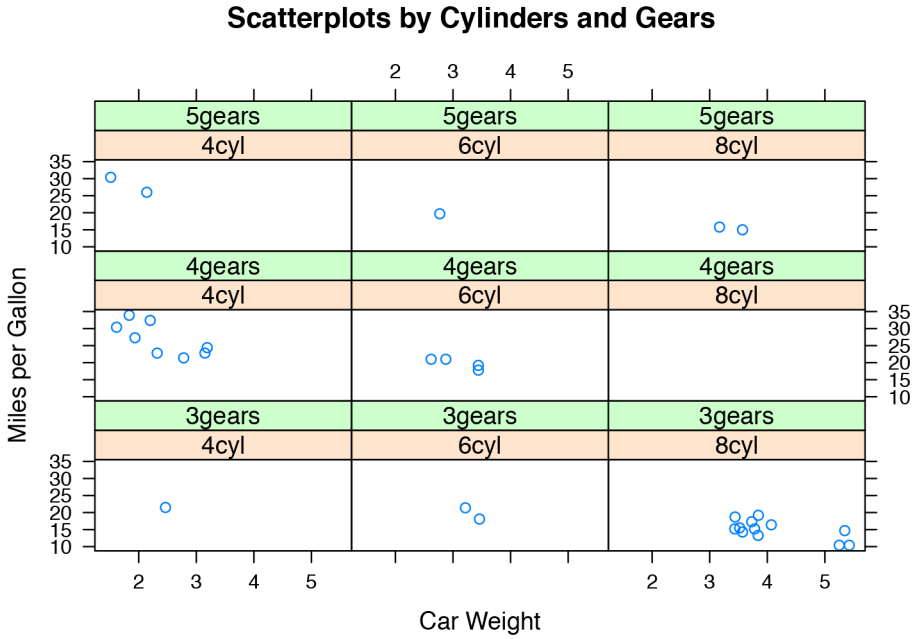
R és un llenguatge per a l'anàlisi estadística de dades. Com a tal, una de les seves característiques més destacades és la de la generació de gràfics. En bona part dels llenguatges de programació, la capacitat de crear gràfics la proporcionen llibreries addicionals alienes al seu nucli. No obstant això, en R, els gràfics són natius.

Existeixen dos *motors* gràfics en R. Un motor gràfic és un conjunt de funcions que permeten fer manipulacions gràfiques bàsiques: generar llenços (o *canvas*), traçar línies, dibuixar punts, etc. En general un usuari de R no manipula aquestes funcions directament: utilitza funcions d'*alt nivell*, com `plot`. La funció `plot` és l'encarregada d'invocar aquestes funcions de baix nivell que pinten els segments, cercles, etc. que conformen un gràfic estadístic, amb els seus eixos, les seves etiquetes, etc.

Funcions de R com ara `plot`, `hist`, `barplot`, `boxplot` i altres es basen en el motor tradicional de R. El motor tradicional de R és suficient per a aquestes finalitats. Tanmateix, queda curt per a construir un altre tipus de gràfics més avançats. Per això, el 2001 Paul Murrell va desenvolupar un motor gràfic alternatiu, `grid`. Un dels objectius de Paul Murrell era facilitar la generació en R d'uns tipus de gràfics coneguts com de Trellis, de gelosia o de petits múltiples.

```
## The following object is masked from package:ggplot2:
##
##     mpg
```

```
## The following objects are masked from mtcars (pos = 10):
##
##     am, carb, cyl, disp, drat, gear, hp, mpg, qsec, vs, wt
```




Els gràfics de Trellis permeten seguir el comportament d'unes variables d'interès a través dels diferents nivells d'altres i disposar la informació en una retícula que facilita el descobriment de patrons per inspecció visual. El gràfic anterior mostra la relació entre el pes i el consum de gasolina d'una sèrie de vehicles en funció del seu nombre de cilindres i el nombre de marxes. La relació és més evident fent servir un gràfic de Trellis que, per exemple, utilitzant colors (o formes) per a representar el nombre de cilindres o marxes en un únic gràfic de dispersió.

Hi ha moltes funcions i paquets que creen gràfics a partir del motor gràfic tradicional. Altres utilitzen `grid`. Dos dels més coneguts són `lattice` (amb el qual està generat el gràfic anterior) i `ggplot2`. De fet, `lattice` i `ggplot2` s'encavalquen funcionalment i la majoria dels usuaris de R es decanten per un o altre i el fan servir predominantment.

`ggplot2` és uns anys posterior a `lattice` però, malgrat això, més popular. `ggplot2` és una implementació de les idees recollides en l'article de «The Language of Graphics»*, escrit per Leland Wilkinson i altres el 2000. Aquest article recollia una sèrie d'idees noves sobre què és la representació gràfica d'informació estadística i com s'hauria de fer. Els gràfics tradicionals de R tenen un important element d'adhoquisme: les funcions per a representar diagrames de dispersió, de caixes, de barres, etc. utilitzen dades amb diferents formats, tenen paràmetres no sempre coincidents en nom, etc. El revolucionari del plantejament de l'article és posar de manifest que tots aquests tipus de gràfics (i altres) es poden generar mitjançant un *llenguatge* més o menys regular, amb la seva sintaxi, la seva semiòtica, etc. De la mateixa manera que el llenguatge natural organitza sons mitjançant certes regles comunes, conegudes i regulars per a generar missatges amb significat, és possible construir una sèrie de regles comunes, conegudes i regulars per a crear representacions visuals de dades d'interès estadístic.

*https://www.cs.uic.edu/ wilkinson/Publications/gpl.pdf

D'aquest llenguatge, implementat en el paquet `ggplot2`, s'ocupen els subapartats següents.

### 1.1. Instal·lació de ggplot2

El paquet `ggplot2` s'instal·la com qualsevol altre paquet de R: o bé des de línia d'ordres amb un `install.packages(ggplot2)` o bé utilitzant els menús (`Tools` > `Install Packages`, etc.) de la interfície de RStudio. `ggplot2` depèn, com ha quedat clar més amunt, del paquet `grid`, però aquest últim ve instal·lat en R per defecte sempre.

El paquet té altres dependències, que R sap instal·lar pel seu compte si no se'n disposa. Dues d'aquestes dependències, els paquets `plyr` i `reshape2`, tenen certa rellevància: són del mateix autor que `ggplot2`, Hadley Wickham, i el seu ús per a organitzar convenientment les dades per a la seva representació gràfica encaixa en la filosofia de `ggplot2`. És recomanable familiaritzar-s'hi.

Òbviament, per a utilitzar `ggplot2` una vegada instal·lat cal importar-lo així:

```
library(ggplot2)
```

### 1.2. Elements d'un gràfic en `ggplot2`

Un gràfic en `ggplot2` es construeix combinant una sèrie d'elements bàsics i comuns a molts tipus de gràfics diferents mitjançant una sintaxi senzilla. Aquest subapartat descriu aquesta sintaxi i els elements que articula.

#### 1.2.1. Dades

Un dels elements més importants d'un gràfic són les dades que es volen representar. Una particularitat de `ggplot2` és que només accepta un tipus de dades: `data.frames`. Altres funcions gràfiques (per exemple, `hist`) admeten vectors, llistes o altres tipus d'estructures. `ggplot2` no.

```
p <- ggplot(iris)
```

El codi anterior crea un objecte, `p`, que ve a ser un protogràfic: conté les dades que farem servir, les del conjunt de dades `iris` (a `?iris` n'hi ha una descripció). Òbviament, el codi anterior és insuficient per a crear un gràfic: encara no hem indicat què volem fer amb `iris`.

### 1.2.2. Estètiques

En un conjunt de dades hi ha columnes: edat, alçada, ingressos, temperatura, etc. En un gràfic hi ha, en la terminologia de `ggplot2`, *estètiques*. Estètiques són, per exemple, la distància horitzontal o vertical, el color, la forma (d'un punt), la grandària (d'un punt o el gruix d'una línia), etc. Igual que en parlar associem a un conjunt de sons (per exemple, t-a-u-l-a) un significat (l'objecte que coneixem com a taula), en fer un gràfic n'associem un a elements sense significat propi (per exemple, els colors): el que correspon a una columna determinada de les dades.

> En `ggplot2`, dins del llenguatge dels gràfics que implementa, és molt important aquesta associació explícita de significats a significants, és a dir, de columnes de dades a *estètiques*.

En el codi

```
p <- p + aes(x = Petal.Length, y = Petal.Width, colour = Species)
```

s'està afegint a `p` informació sobre les estètiques que ha de fer servir i quines variables de `iris` ha d'utilitzar:

- La distància horitzontal, `x`, vindrà donada per la longitud del pètal.
- La distància vertical, `y`, per la seva amplària.
- El color, per l'espècie.

Cal fer notar la sintaxi del codi anterior, bastant particular i pròpia del paquet `ggplot2`. Al *protogràfic* se li han sumat les estètiques. En els subapartats següents se li *sumaran* altres elements addicionals. L'important és recordar com la suma és el signe que combina els elements que componen el llenguatge dels gràfics.

De totes maneres, és habitual combinar els dos passos en una única expressió:

```
p <- ggplot(iris, aes(x = Petal.Length, y = Petal.Width, colour = Species))
```

L'objecte `p` resultant encara no és un gràfic ni es pot representar. Li falten capes, que és l'objecte del subapartat següent. Ara bé, es pot inspeccionar així:

```
summary(p)
```

```
## data: Sepal.Length, Sepal.Width, Petal.Length, Petal.Width,
##   Species [150x5]
## mapping:  x = Petal.Length, y = Petal.Width, colour = Species
## faceting: facet_null()
```

Aquí estan indicades les dades que farà servir i la relació (o *mapatge*) entre estètiques i columnes de les dades.

Quantes estètiques hi ha? Al voltant d'una dotzena, encara que se n'utilitzen, generalment, menys:

- `x` i `y` , distàncies horitzontal i vertical.
- `colour`, per al color.
- `size`, per a la grandària.
- `shape`, que indica la forma dels punts (quadrats, triangles, etc.) dels punts o del traç (continu, puntejat) de les línies.
- `alpha` per a la transparència: els valors més alts tindrien formes opaques i els més baixos, gairebé transparents. D'aquí la utilitat del canal alfa: dona pes i importància a les observacions que la mereixen.
- `fill`, per al color de farciment de les formes sòlides (barres, etc.).

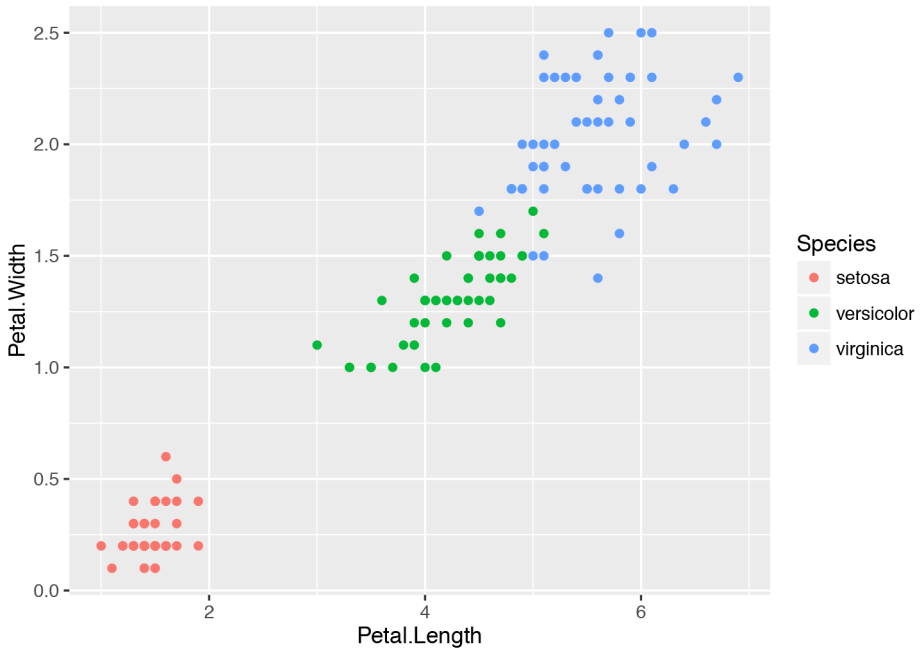
**Estètiques potents**

Cal advertir que no totes les *estètiques* tenen la mateixa potència en un gràfic. L'ull humà percep fàcilment longituds diferents, però té problemes per a comparar àrees (que és el que regula l'estètica `size`) o intensitats de color. Es recomana usar les estètiques més potents per a representar les variables més importants.

### 1.2.3. Capes

Les capes (o `geoms` per a `ggplot2`) són els verbs del llenguatge dels gràfics. Indiquen què s'ha de fer amb les dades i les estètiques triades, com s'han de representar en un llenç. I, en efecte, el codi següent crea el gràfic corresponent.

```
p <- p + geom_point()
p
```

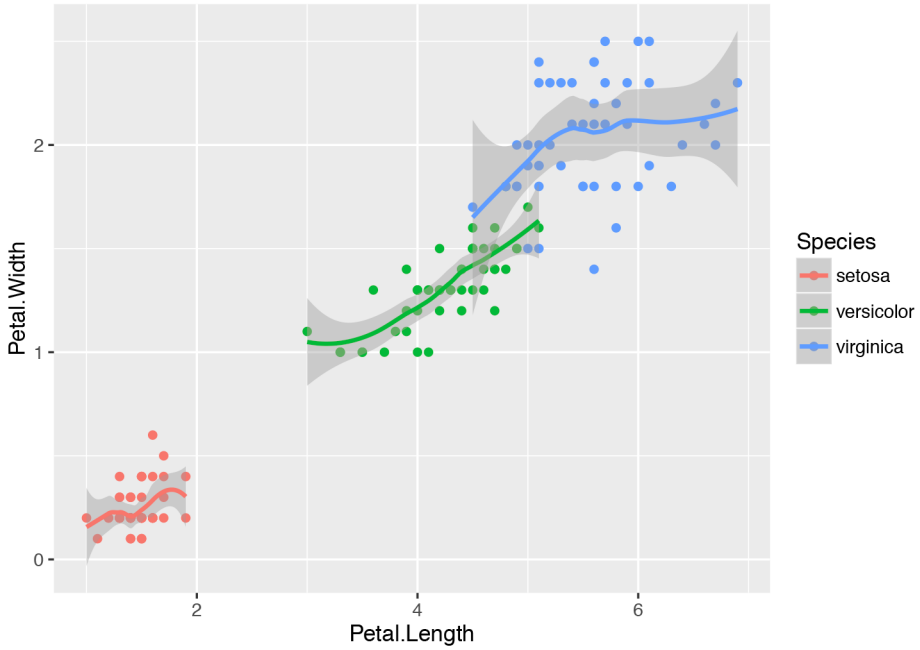


Una vegada afegida una capa al gràfic, es pot pintar (que és el que passa en invocar `p`). S'obté el mateix resultat fent, en una única línia,

```
ggplot(iris, aes(x = Petal.Length, y = Petal.Width, colour = Species)) + geom_point()
```

Una característica de les capes, i d'aquí el seu nom, és que es poden superposar. Per exemple, el codi següent afegeix al gràfic una corba suavitzada (amb els seus intervals de confiança en gris).

```
ggplot(iris, aes(x = Petal.Length, y = Petal.Width, colour = Species)) +
  geom_point() + geom_smooth()
```




Hi ha molts tipus de capes. Els més usuals són `geom_point`, `geom_line`, `geom_histogram`, `geom_bar` i `geom_boxplot`, però n'hi ha més.

A la pàgina http://docs.ggplot2.org/current/ es mostra una llista dels disponibles (en la versió més actualitzada de `ggplot2`). En aquesta pàgina s'indica quina *geom* cal utilitzar en funció d'una representació esquemàtica del tipus de gràfic que es vol construir. A més, hi ha capes específiques que exigeixen estètiques especials. Per a algunes té sentit, per exemple, `shape`. Per a altres, no. Aquestes especificitats estan indicades en aquesta pàgina, que és més útil que l'ajuda general de R.

Una vegada creat un gràfic, és possible exportar-lo a PNG, JPG, etc. La funció `ggsave` desa en un fitxer l'últim gràfic generat amb `ggplot2`. Ho fa, a més, en el format indicat en el nom del fitxer que es vol generar. Així,

```
ggplot(iris, aes(x = Petal.Length, y = Petal.Width, colour = Species)) + geom_point()
ggsave("mi_grafico.png")
```

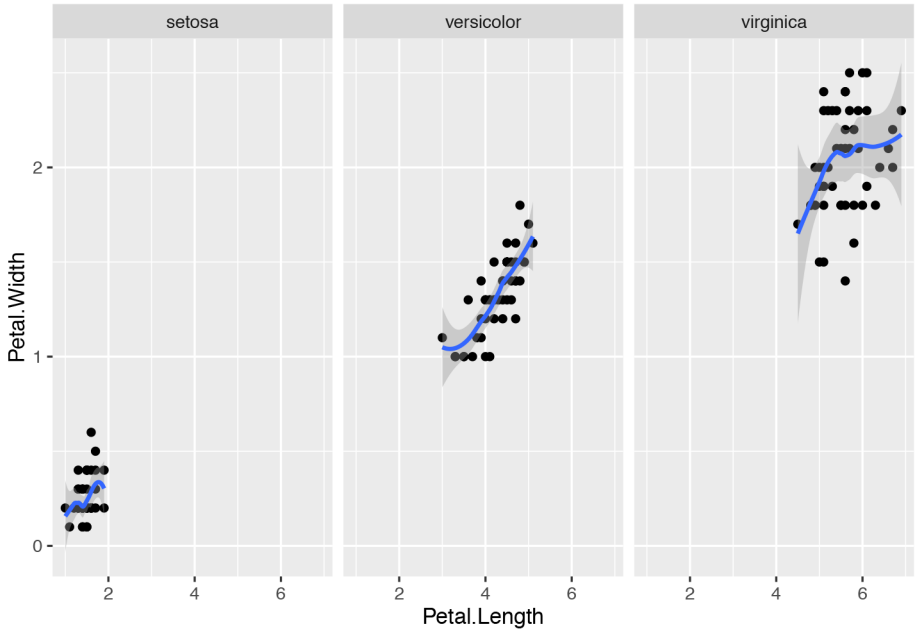
fa que es guardi la figura creada en la primera línia en format PNG en el fitxer `mi_grafico.png` del directori de treball.

### 1.2.4. Facetes

Molts dels gràfics que es poden generar amb els elements anteriors es poden reproduir sense gaire esforç (exceptuant, potser, qüestions d'aspecte) fent servir els gràfics tradicionals de R. Els que permeten l'ús de facetes, no.

Les facetes implementen els gràfics de Trellis esmentats abans. Per exemple, el codi següent crea tres gràfics disposats horitzontalment que comparen la relació entre l'amplària i la longitud del pètal de les tres espècies d'iris. Una característica d'aquests gràfics, que és crítica per a poder fer comparacions adequades, és que comparteixen eixos.

```
ggplot(iris, aes(x = Petal.Length, y = Petal.Width)) +
  geom_point() + geom_smooth() +
  facet_grid(~ Species)
```




Els gràfics es poden disposar verticalment canviant `facet_grid(~ Species)` per `facet_grid(Species ~ .)` en el codi anterior. En cas d'haver-hi moltes categories (per exemple, província), es pot fer servir la funció `facet_wrap` per a distribuir els subgràfics en una quadrícula.

### 1.2.5. Més sobre estètiques

Les estètiques es poden etiquetar amb la funció `labs`. A més, es pot afegir un títol al gràfic fent servir la funció `ggtitle`. Per exemple, en el gràfic anterior (subapartat 1.2.3.) es poden reetiquetar els eixos i la llegenda fent

```
p <- p + ggtitle("Petal length and width") +
         labs(x = "Petal length",
              y = "Petal width",
              colour = "Species")
```

### 1.2.6. Temes

Els *temes* de `ggplot2` permeten modificar aspectes estètics del gràfic que no tenen a veure amb les dades en si. Això inclou els eixos, les etiquetes, els colors de fons, la grandària dels marges, etc. No és habitual (i es desaconsella als usuaris menys experts) haver d'alterar els temes que `ggplot2` fa servir per defecte. Només es fa necessari quan els gràfics s'han d'adequar a una imatge corporativa o cal atenir-se a algun criteri de publicació exigent.

Un tema és una col·lecció d'elements (per exemple, `panel.background`, que indica el color, la transparència, etc., del llenç sobre el qual es representa el gràfic) modificables. El tema que usa `ggplot2` per defecte és `theme_grey`. En escriure `theme_grey()` en la consola de R, es mostren al voltant de quaranta elements modificables i els seus atributs tal com els defineix aquest tema.

Què es pot fer amb els temes? Una primera opció és triar-ne un altre. Per exemple, es pot reemplaçar l'habitual per altres de disponibles en el paquet com `theme_bw` (o `theme_classic`) fent

```
p <- p + theme_bw()
```

És possible utilitzar tant els temes que inclou `ggplot2` per defecte com altres de creats per la comunitat. Alguns, per exemple, tracten d'imitar l'estil de publicacions reconegudes, com *The Economist* o similars. Alguns estan recollits en paquets com, per exemple, `ggthemes`.

De manera alternativa (o addicional), és possible modificar un tema donat en un gràfic. Per exemple, fent

```
p <- p +
  theme_bw() +
  theme(
    panel.background = element_rect(fill = "lightblue"),
    panel.grid.minor = element_line(linetype = "dotted")
  )
```

s'està modificant l'atribut de color del llenç d'un gràfic i el tipus de la línia amb la qual es dibuixa la malla.

**Personalització de temes**

És possible construir temes propis i personalitzats. Malgrat que no és un procés complicat, els detalls queden fora de l'abast d'aquest material.

## 1.3. Exemples

En aquest subapartat s'exploraran alguns dels gràfics estadístics més bàsics.

### 1.3.1. Diagrames de caixes (i de violí)

Els diagrames de caixa (*boxplots*) descriuen de manera crua la distribució d'una variable contínua en funció d'una altra de discreta. En l'exemple que apareix a continuació s'explora el conjunt de dades `iris`, que conté cinquanta observacions de característiques mètriques de cadascuna de les tres subespècies d'iris, una flor. És un conjunt de dades de llarga tradició en estadística i es va recopilar per a il·lustrar algorismes de classificació, és a dir, com es creen criteris per a distingir els tres tipus d'iris.

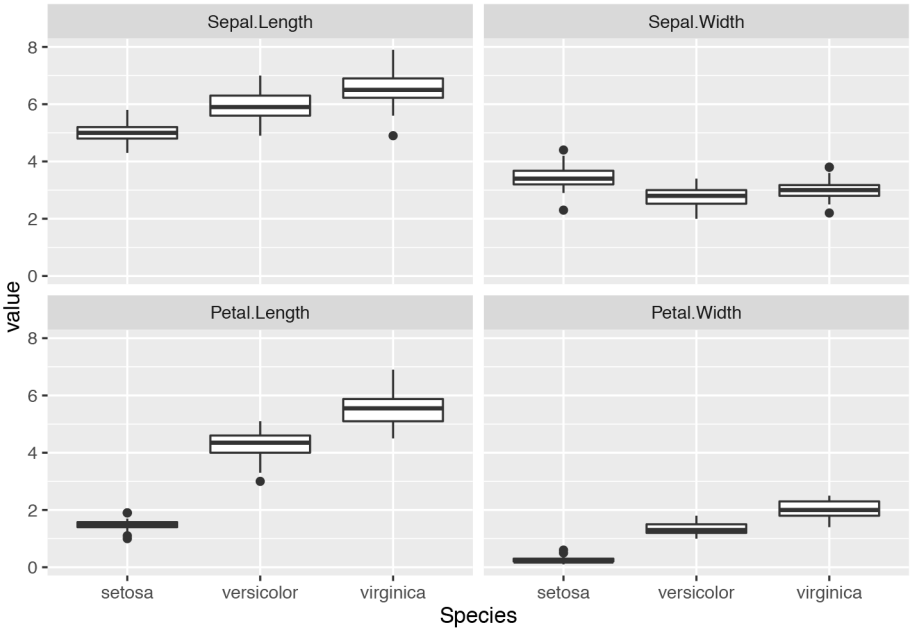
**Iris dataset**

https://archive.ics.uci.edu/ml/datasets/Iris

La funció `melt()` crea una nova taula a partir d'unes dades (que estan en el que es coneix com a format ample, amb un camp per a cada valor), de manera que hi hagi un registre per a cada un dels camps que es desitja recodificar (normalment tots aquells que no són factors), fent servir dues noves variables, una anomenada variable, que conté el nom del camp original, i una altra anomenada `value`, que conté el valor del camp original.

```
library(reshape2)
tmp <- melt(iris)
```

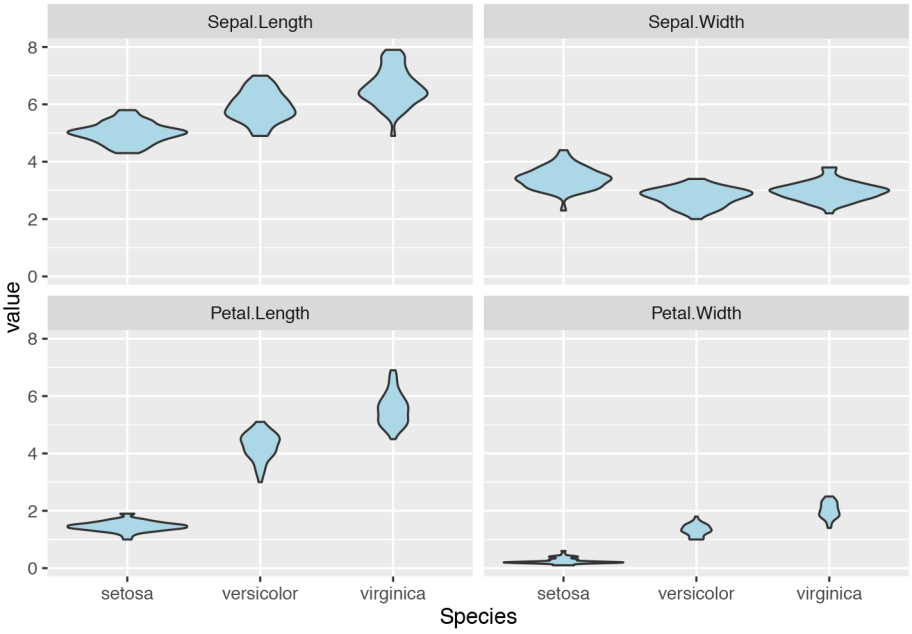
```
ggplot(tmp, aes(x = Species, y = value)) + geom_boxplot() + facet_wrap(~ variable)
```



El gràfic utilitza les facetes per a crear quatre panells, un per variable. Així resumeix ràpidament la informació continguda en el conjunt de dades i revela eficaçment els patrons que inclou: per exemple, com l'espècie setosa té el pètal sensiblement més estret i curt que les altres dues. Aquest tipus de gràfics és fonamental com a eina exploratòria prèvia a l'aplicació de tècniques d'anàlisi estadística (discriminant, en aquest cas).

Els gràfics de caixes són molt bàsics: amb prou feines mostren cinc punts característics d'una distribució: la mitjana, els extrems i els quartils. Són herència d'una època en la qual amb prou feines hi havia recursos, principalment informàtics, per a fer representacions més sofisticades. Una versió moderna dels gràfics de caixes és la de gràfics de violí. Com els de caixes, resumeixen la distribució de les variables, però en lloc d'una representació succinta, tracten de dibuixar la distribució real de les dades: són veritables gràfics de densitat, només que disposats d'una altra manera per a facilitar la comparació.

```
ggplot(tmp, aes(x = Species, y = value)) +
  geom_violin(fill = "lightblue") +
  facet_wrap(~ variable)
```

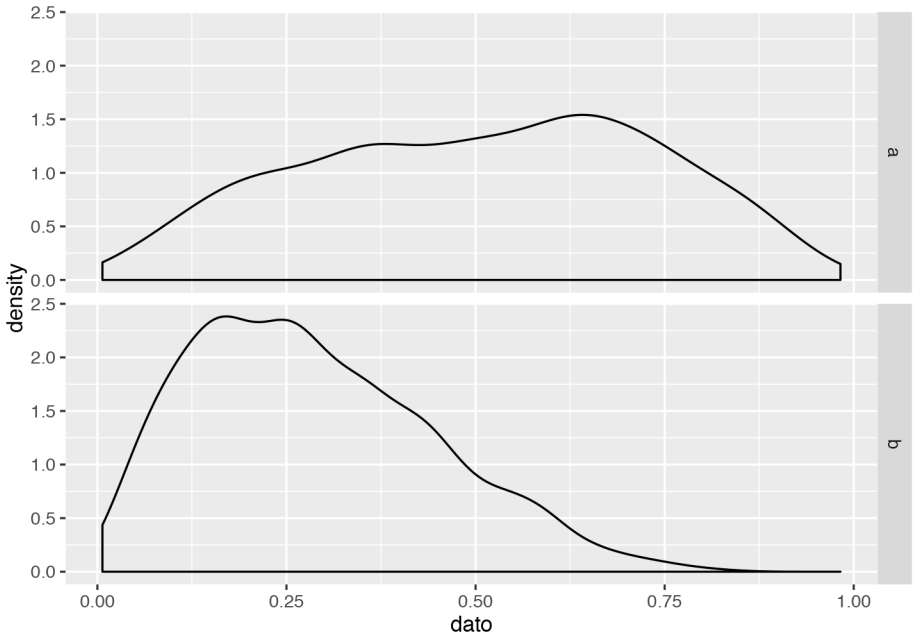



### 1.3.2. Comparació de dues densitats

L'anàlisi de dades exigeix de vegades comparar dues distribucions contínues. Es poden usar gràfics de caixes o de violí, com a dalt, però també es pot dibuixar la distribució completa, com en el gràfic següent:

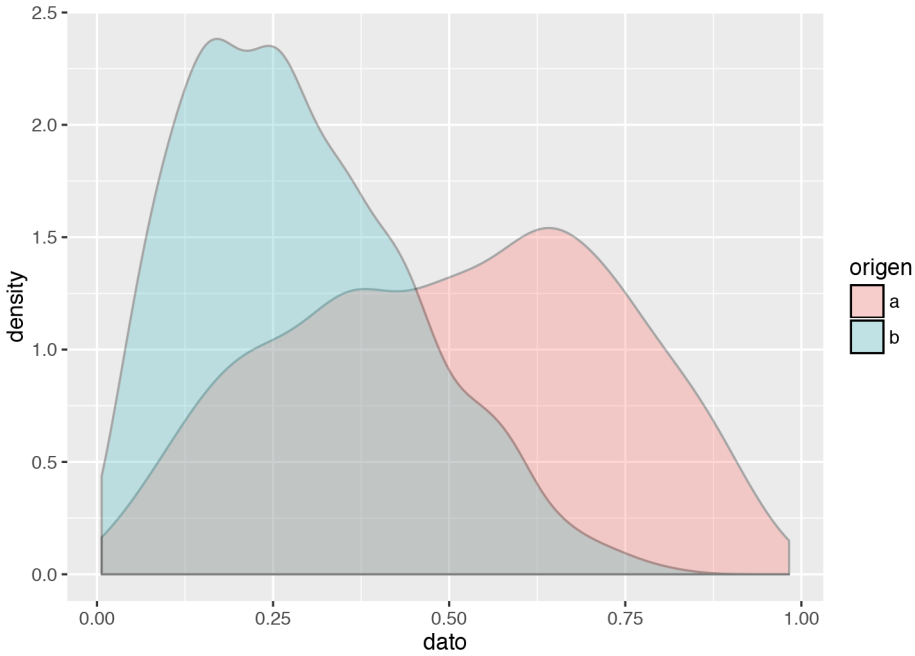
```
# dades (simulades)
a <- rbeta(1000, 2, 2)
```

```
b <- rbeta(2000, 2, 5)

# construcció d'un dataframe a partir d'ells
tmp <- rbind(data.frame(origen = "a", dato = a),
             data.frame(origen = "b", dato = b))

ggplot(tmp, aes(x = dato)) + geom_density() + facet_grid(origen ~ .)
```




Alternativament, es poden encavalcar les dues distribucions. L'ús del paràmetre `alpha`, que controla la transparència, és fonamental en aquest cas:

```
ggplot(tmp, aes(x = dato, fill = origen)) + geom_density(alpha = 0.3)
```



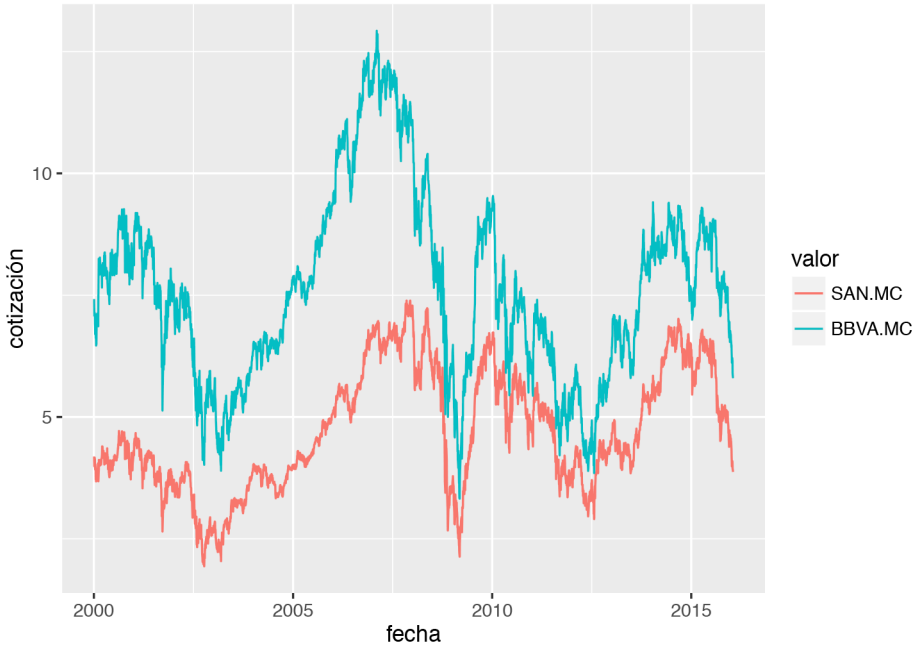
### 1.3.3. Sèries temporals

ggplot2 entén certs tipus de dades particulars, com ara sèries temporals. En l'exemple següent es baixen les cotitzacions borsàries de dos bancs espanyols directament de Yahoo! Finance i es representen gràficament:

```
# instal·lar-los prèviament si és necessari.
library(tseries)
library(zoo)
library(reshape2)

# funció per a baixar les cotitzacions
cotizaciones <- function(valor){
  res <- get.hist.quote(valor, provider = "yahoo", quote = "AdjClose")
  colnames(res) <- valor
  res
}

# combinació de les dues sèries temporals
res <- merge(cotizaciones("SAN.MC"),
             cotizaciones("BBVA.MC"))


## time series starts 2000-01-03
## time series starts 2000-01-03


# construcció d'un dataframe con un índex de tipus data
res <- data.frame(fecha = index(res), as.data.frame(res))
res <- melt(res, id.var = "fecha")

ggplot(res, aes(x = fecha, y = value)) + geom_line() + facet_grid(variable ~ .)
```

Una versió alternativa del gràfic és la que superposa les sèries fent servir colors per a distingir-les:

```
ggplot(res, aes(x = fecha, y = value, colour = variable)) + geom_line() +
  labs(colour = "valor", y = "cotización")
```




### 1.4. Dades mitjanes grans

Cada vegada és més freqüent enfrontar-se al problema de la representació gràfica de dades massives (*big data*). Moltes de les representacions gràfiques tradicionals són impossibles per diversos motius:

- Exigeixen un preprocessament de dades (per exemple, si es vol fer servir `geom_smooth`) molt pesat.
- Aquests algorismes no estan pensats per a dades massives; és a dir, hi podria haver implementacions d'aquests algorismes que podrien valer per a dades massives, però les existents no serveixen.
- Però, sobretot, perquè les funcions gràfiques són molt lentes: representar milions de cercles sobre el llenç (o *canvas*) d'un fitxer `.png` és molt costós computacionalment.

Tanmateix, sempre es pot recórrer a representacions gràfiques que agreguin dades d'alguna manera determinada. O, alternativament, fer la preagregació abans d'aplicar les funcions gràfiques pròpiament dites.

En el que segueix, es faran algunes exploracions gràfiques sobre part d'un conjunt de dades molt popular per a fer proves en entorns de dades massives: el dels retards de les companyies aèries*, que conté informació sobre tots els vols comercials als Estats Units entre 1998 i 2008, indicant-ne origen, destinació, temps de vol, distància de vol, temps de retard, incidències, etc. S'utilitzarà el corresponent al 2008, que conté informació de poc més de set milions de vols, ocupa 650 MB i es pot baixar de l'enllaç anterior.

*http://stat-computing.org/dataexpo/

En primer lloc, es procedirà a carregar-lo (per a fer-ho es recomana la funció `fread` del paquet `data.table`, molt més eficient, encara que també menys flexible, que l'habitual `read.table` per a dades massives). També se n'extraurà un subconjunt *mitjà* (100.000 files) a l'atzar.

```
library(data.table)

raw <- fread("2008.csv")
small <- raw[sample(1:nrow(raw), 100000),]
```

És temptador tractar d'executar

```
ggplot(small, aes(x=Distance, y = AirTime)) + geom_point()
```

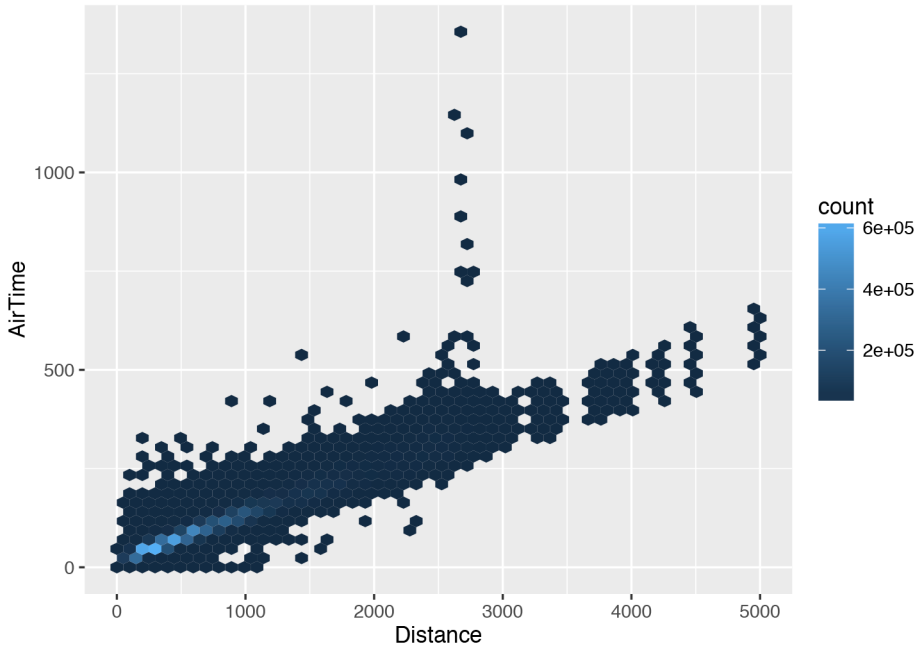
amb el conjunt mitjà de dades. No obstant això, a diferència del que passa amb conjunts més petits, l'opacitat dels punts, juntament amb el seu gran nombre, produeixen un efecte de taca negra que amaga els detalls. En aquests casos és recomanable modular la transparència així

```
ggplot(small, aes(x=Distance, y = AirTime)) +
  geom_point(alpha = 0.01) + geom_smooth()
```

Gràcies a la transparència, es detecten fàcilment les zones de més densitat de dades. Ara bé, aquest tipus de subterfugis fallen quan es vol representar el conjunt de dades complet amb els seus set milions de files. De tots els *geoms* que ofereix `ggplot2` només uns pocs, els que fan preagregacions de dades, es poden aplicar efectivament: histogrames, tessel·lacions hexagonals (`geom_hex`), diagrames de caixes, etc.

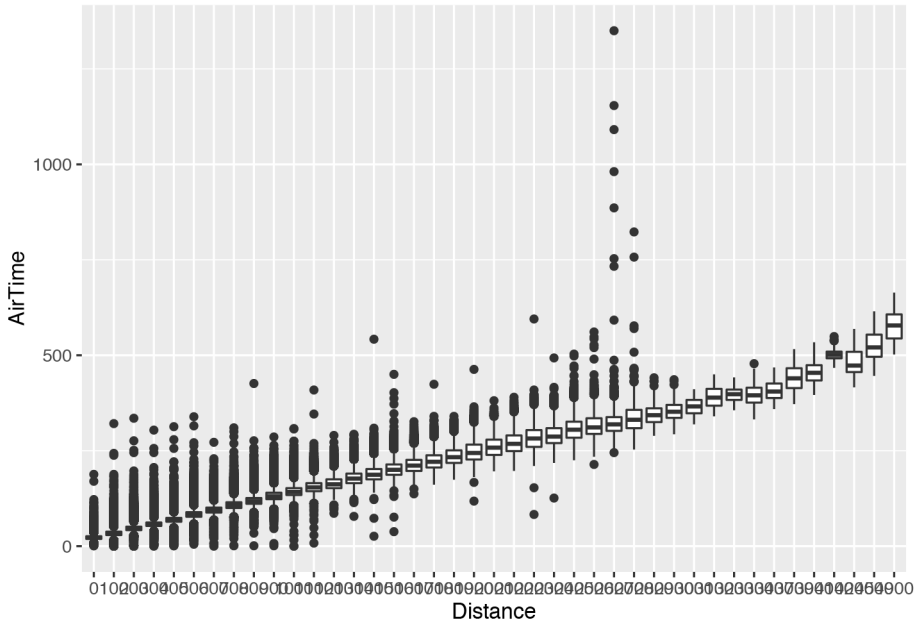
Per exemple,

```
ggplot(raw, aes(x=Distance, y = AirTime)) + geom_hex(bins = 50)
```



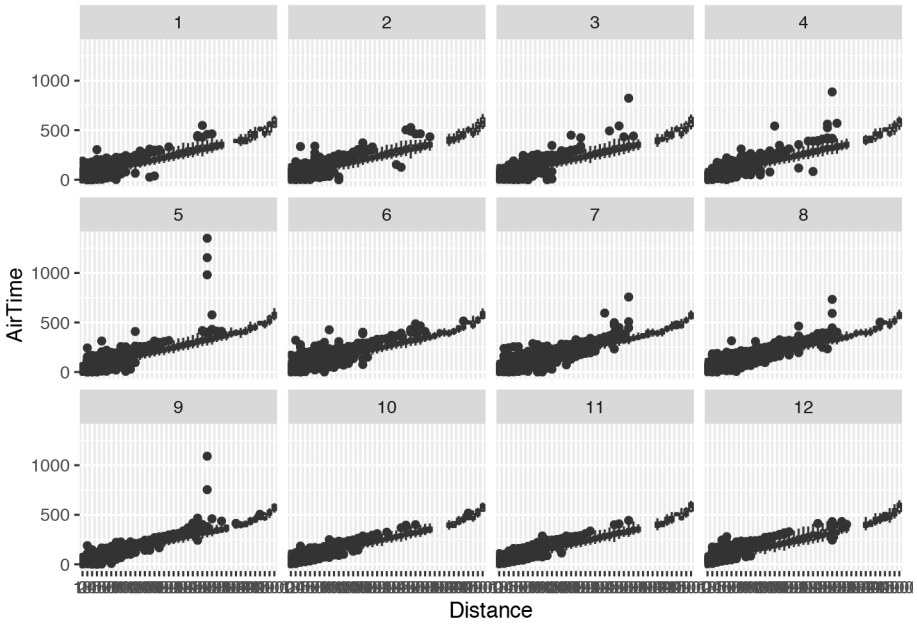

La binarització d'una variable contínua permet també utilitzar diagrames de caixa (noteu que cada caixa *resumeix* informació relativa a un conjunt potencialment gran de dades) per a fer visualitzacions efectives:

```
raw$DistanceBreaks <- factor(100* floor(raw$Distance / 100))
ggplot(raw, aes(x = DistanceBreaks, y = AirTime)) + geom_boxplot() +
  xlab("Distance")
```




I, per descomptat, l'ús de facetes no suposa un increment inassumible de la complexitat computacional:

```
ggplot(raw, aes(x = DistanceBreaks, y = AirTime)) + geom_boxplot() +
  facet_wrap(~ Month) + xlab("Distance")
```




Cal notar que l'eix de les X en tots dos gràfics resulta il·legible, donada la gran quantitat de dades a mostrar. Amb `ggplot2` també és possible determinar l'etiquetatge de cada un dels eixos, resolent aquest problema.

Hi ha consideracions addicionals a l'hora de representar conjunts de dades molt grans. El cas anterior, per exemple, posa de manifest com pocs retards aïllats (dins d'un conjunt de milions de vols) agafen un protagonisme excessiu: per tant, valdria la pena filtrar-los prèviament per a recalcar l'estructura general de les dades? O són precisament aquests valors atípics els que exigeixen la nostra atenció? Depenent de les respostes a aquestes preguntes es podrà plantejar la possibilitat de fer algun tipus de filtre previ.

## 1.5. Resum

`ggplot2` és un paquet modern per a crear gràfics estadístics avançats. Utilitza una sintaxi particular que tracta d'unificar la manera de generar gràfics. Exigeix un cert esforç per part dels qui s'inicien en el seu ús, però permet arribar molt més lluny que els gràfics tradicionals de R.

## 2. Introducció a ggmap

Amb `ggplot2` es poden construir molts tipus de gràfics d'interès estadístic, però els seus autors van voler traslladar l'arquitectura del paquet a un altre àmbit: el de la representació d'informació georeferenciada. `ggplot2` permet representar informació geogràfica (punts, segments, etc.): n'hi ha prou que les estètiques `x` i `y` es corresponguin amb la longitud i la latitud de les dades. El que permet fer `ggmap` és, en essència, afegir als gràfics ja coneguts una capa cartogràfica addicional. Per a això, empra recursos disponibles al web a través d'API (de Google i altres).

Un exemple senzill il·lustra els usos de `ggmap`. En primer lloc, es carrega (si s'ha instal·lat prèviament) el paquet:

```
library(ggmap)
```

Hi ha diversos proveïdors que proporcionen API de geolocalització. Un és Google: donat el nom més o menys normalitzat d'un lloc, l'API de Google en retorna les coordenades. Aquest servei té una versió gratuïta, que permet fer un determinat nombre de consultes diàries (2.500 actualment); per a usos més intensius, cal adquirir una llicència. La funció `geocode` encapsula la consulta en aquesta API i retorna un objecte (un `data.frame`) que conté les coordenades del lloc d'interès:

```
unizar <- geocode('Universitat de Saragossa, Saragossa, Espanya',
                  source = "google")
```

La funció `get_map` consulta un altre servei d'informació cartogràfica (Google Maps en l'exemple següent) i baixa un mapa (que és, essencialment, una imatge *raster*). La funció exigeix una sèrie d'arguments: el nivell de *zoom*, si es vol un mapa de carreteres o del terreny, etc. Són, de fet, els paràmetres que un pot manipular amb els controls de la interfície habitual de Google Maps.

```
map.unizar <- get_map(location = as.numeric(unizar),
                      color = "color",
                      maptype = "roadmap",
                      scale = 2,
                      zoom = 16)
```

És obvi que per a poder invocar les dues funcions anteriors fa falta una connexió a internet. Ara bé, la resta de les operacions que es duran a terme s'executen localment. Es pot, per exemple, representar el mapa directament (fent `ggmap(map.unizar)`). O també s'hi pot marcar a sobre el punt d'interès:

```
ggmap(map.unizar) + geom_point(aes(x = lon, y = lat),
                               data = unizar, colour = 'red',
                               size = 4)
```


Com es pot apreciar, la sintaxi és similar a la de `ggplot2`. Una diferència notable és que, ara, les dades es passen a la capa, és a dir, en aquest cas, a la funció `geom_point`.

### 2.1. Funcions de `ggmap`

`ggmap` inclou moltes funcions, que es poden classificar en tres categories àmplies:

- Funcions per a obtenir mapes (de diversos tipus i de diferents orígens: Google, Stamen, OpenStreetMap).
- Funcions que utilitzen API de Google i altres. Per exemple, `revgeocode`, `geocode` i `route` consulten la informació que tenen diferents proveïdors de serveis per mitjà d'API sobre les coordenades d'un lloc determinat; indiquen el lloc al qual es refereixen unes coordenades i, finalment, troben rutes entre dos punts. És convenient recordar que les consultes als serveis

de Google Maps exigeix l'acceptació de les condicions d'ús i que hi ha un límit diari en el nombre de consultes gratuïtes.

- Funcions que pinten mapes i que representen determinats elements addicionals (punts, segments, etc.) en mapes.

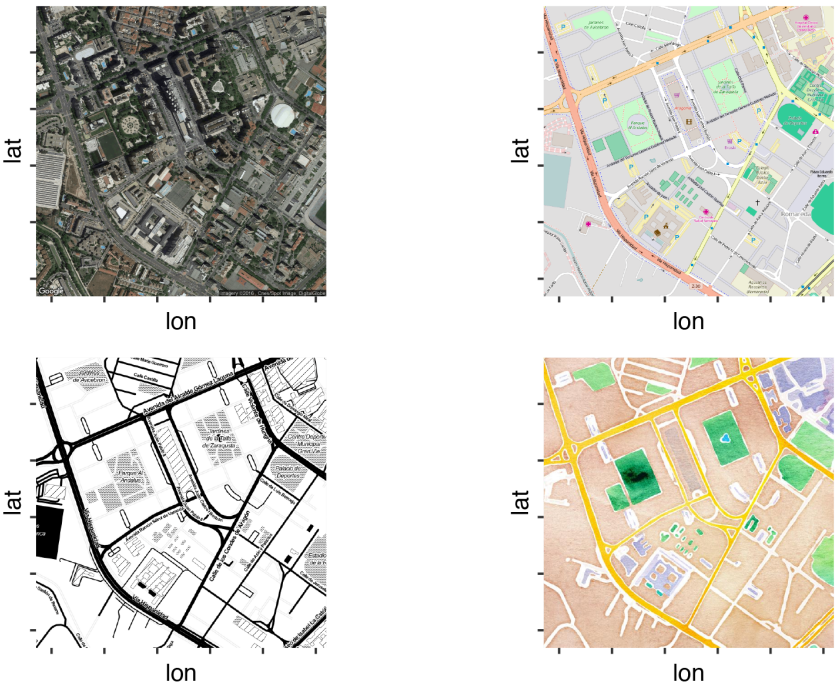
En aquest subapartat es presentaran els tres tipus de funcions de `ggmap`.

### 2.1.1. Funcions per a obtenir mapes

`ggmap` obté els seus mapes, per defecte, de Google Maps. No obstant això, hi ha altres proveïdors de mapes lliures, com OpenStreetMap (OSM) o Stamen. Cada proveïdor exigeix una sèrie de paràmetres diferents i, per exemple, un *zoom* de 8 pot significar una escala diferent en Google Maps que en OSM. Tanmateix, els autors de `ggmap` s'han pres la molèstia d'homogeneïtzar els arguments d'invocació perquè siguin aproximadament equivalents en tots els proveïdors.

`ggmap` inclou funcions específiques per a cada proveïdor, com són, per exemple, `get_googlemap` o `get_stamenmap`, però llevat per a usos avançats, és recomanable usar la funció `get_map`, que ofereix un punt d'entrada únic i homogeni per a la resta.

El gràfic següent mostra quatre mapes obtinguts de diferents proveïdors i amb diverses opcions. En la fila superior, una capa de Google en mode imatge de satèl·lit i una altra d'OSM. En la inferior, dos mapes de Stamen, un en mode *toner* i un altre en mode *watercolor* o aquarel·la. Són només quatre dels molts als quals la funció `get_map` pot accedir.



### Funcions per a consultar API cartogràfiques

Molts serveis d'informació cartogràfica proporcionen API per a fer consultes. Les API es consulten, típicament, amb URL convenientment construïts.

> Per exemple, l'URL següent consulta el servei de geolocalització de Google Maps i retorna les coordenades de la Universitat de Saragossa (així com altra informació rellevant en format JSON):
> http://maps.googleapis.com/maps/api/geocode/json?address=Universitat+de+Saragossa
> La funció `geolocate` de `ggmap` facilita la consulta a aquest servei: pren el seu argument (el nom d'un lloc), construeix internament l'URL, duu a terme la consulta (cal tenir connexió a internet), llegeix la resposta i li dona un format convenient (en aquest cas, un `data.frame` de R).

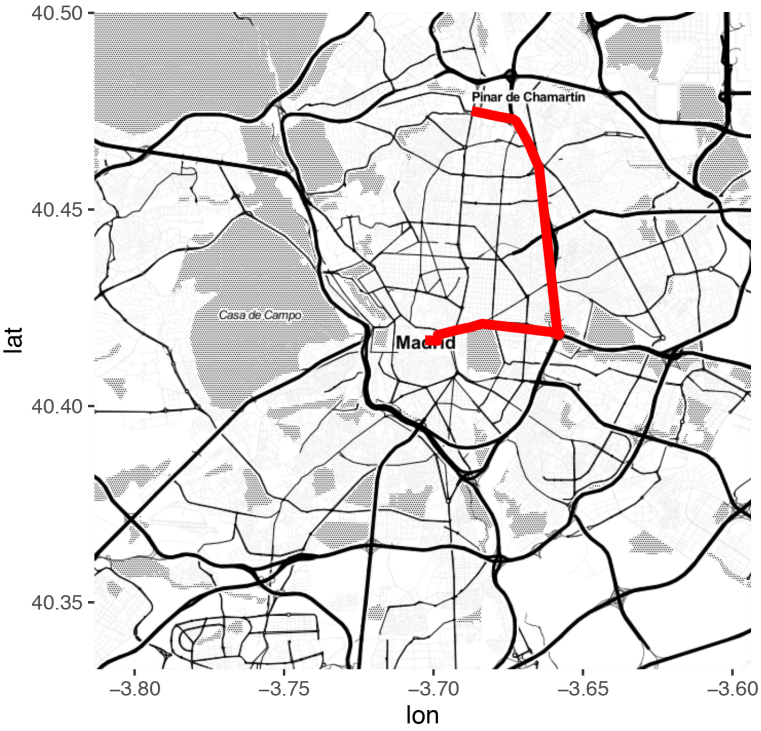
La de geolocalització no és l'única API que permet consultar `ggmap`. També permet invertir la geolocalització, és a dir, donades unes coordenades, retornar el nom del lloc al qual es refereixen:

```
revgeocode(as.numeric(unizar))
```

```
## [1] \comillas Av. Ramón Sainz de Varanda, 112, 50009 Saragossa,
Saragossa, Espanya\textquotedblright
```

Finalment, `route` permet obtenir la ruta entre dos punts diferents:

```
mapa <- get_map("Madrid", source = "stamen", maptype = "toner", zoom = 12)
ruta <- route(from = "Puerta del Sol, Madrid", to = "Plaça de Castilla, Madrid")
ggmap(mapa) +
  geom_path(aes(x = startLon, y = startLat, xend = endLon, yend = endLat),
            colour = "red", size = 2, data = ruta)
```



En el mapa anterior la ruta triada per Google Maps per a anar de la Puerta del Sol fins a la plaça de Castilla (dues places de Madrid) està marcada en vermell sobre un mapa de Stamen de tipus *toner*.

### 2.1.2. Funcions per a representar mapes

Com s'ha vist en els subapartats anteriors, la funció `ggmap` permet representar un mapa baixat prèviament. A més, a aquesta capa subjacent se li poden afegir elements (punts, segments, densitats etc.) usant les funcions ja conegudes de `ggplot2`: `geom_point`, etc.

En `ggplot2` existeix una funció, `geom_path` que dibuixa camins (seqüències de segments). Es pot utilitzar en `ggmap` per a dibuixar rutes, encara que aquest paquet proporciona una funció especial, `geom_leg`, que té la mateixa finalitat, encara que amb algunes diferències menors: per exemple, els segments tenen les puntes arrodonides per a millorar el resultat gràfic.

## 2.2. Exemples

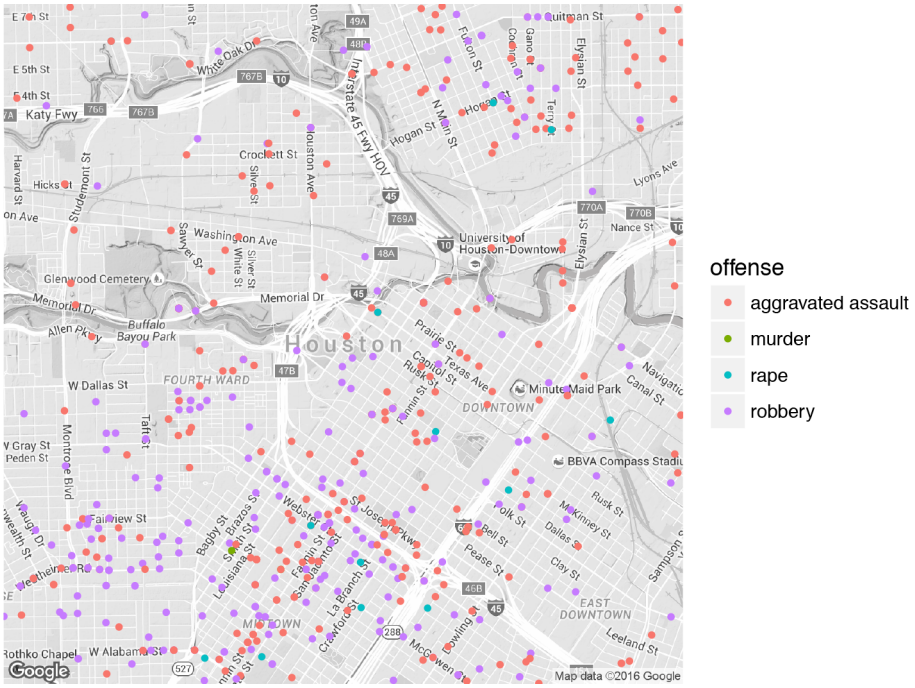
En els exemples que segueixen es farà servir el conjunt de dades `crimes`, que forma part del paquet `ggmap` i que inclou informació geolocalitzada de crims comesos a la ciutat de Houston. En realitat, només considerarem els crims *seriosos*, és a dir, descomptarem els robatoris:

```
crimes.houston <- subset(crime, ! crime$offense %in% c("auto theft", "theft", "burglary"))
```

### 2.2.1. Punts sobre mapes

El tipus de mapes més simple és el que es limita a representar punts sobre una capa cartogràfica.

```
HoustonMap <- qmap("houston", zoom = 14, color = "bw")
HoustonMap +
  geom_point(aes(x = lon, y = lat, colour = offense), data = crimes.houston, size = 1)
```

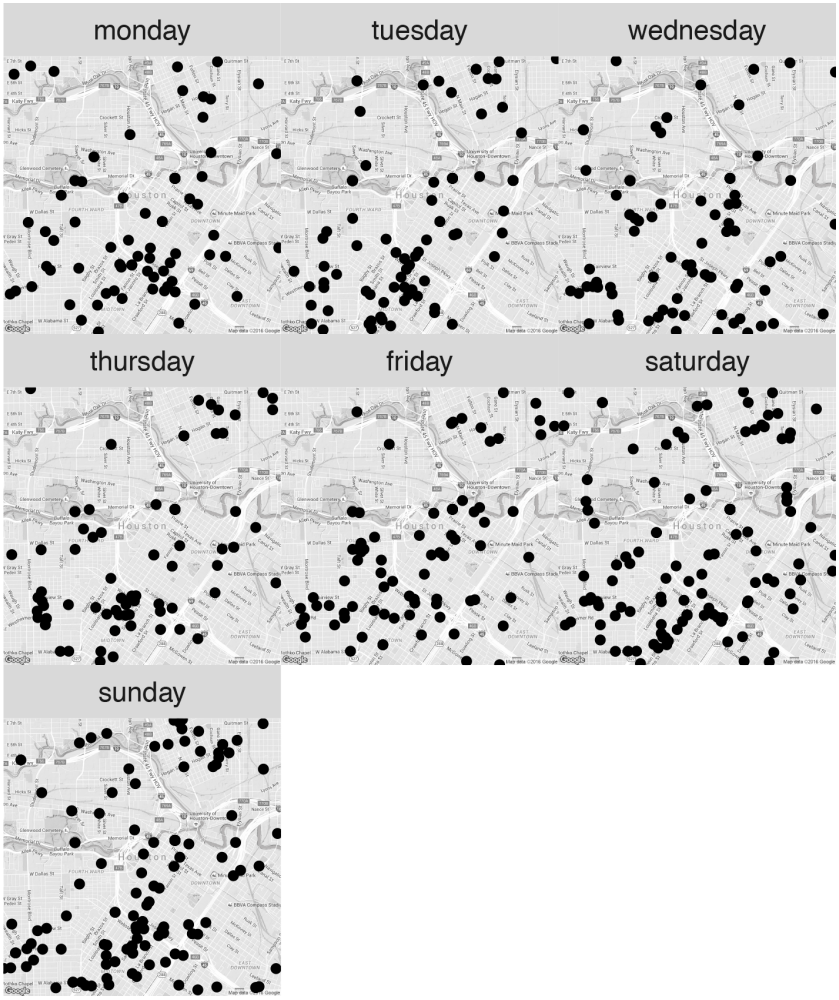



Els mecanismes coneguts de `ggplot2`, com les facetes, estan disponibles a `ggmap`: és possible crear una retícula de mapes usant `facet _wrap`. En aquest primer cas, descomponent el gràfic anterior per tipus de crim.

```
HoustonMap +
  geom_point(aes(x = lon, y = lat), data = crimes.houston, size = 1) +
  facet_wrap(~ offense)
```

O, alternativament, per dia de la setmana.

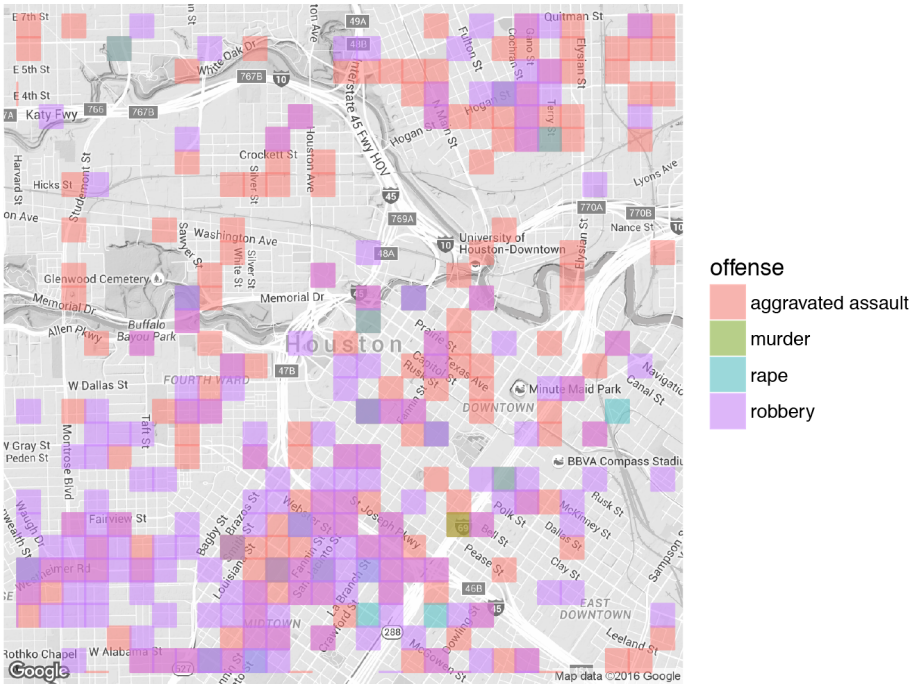
```
HoustonMap +
  geom_point(aes(x = lon, y = lat), data = crimes.houston, size = 1) +
  facet_wrap(~ day)
```




### 2.2.2. Més enllà dels punts: densitats i retícules

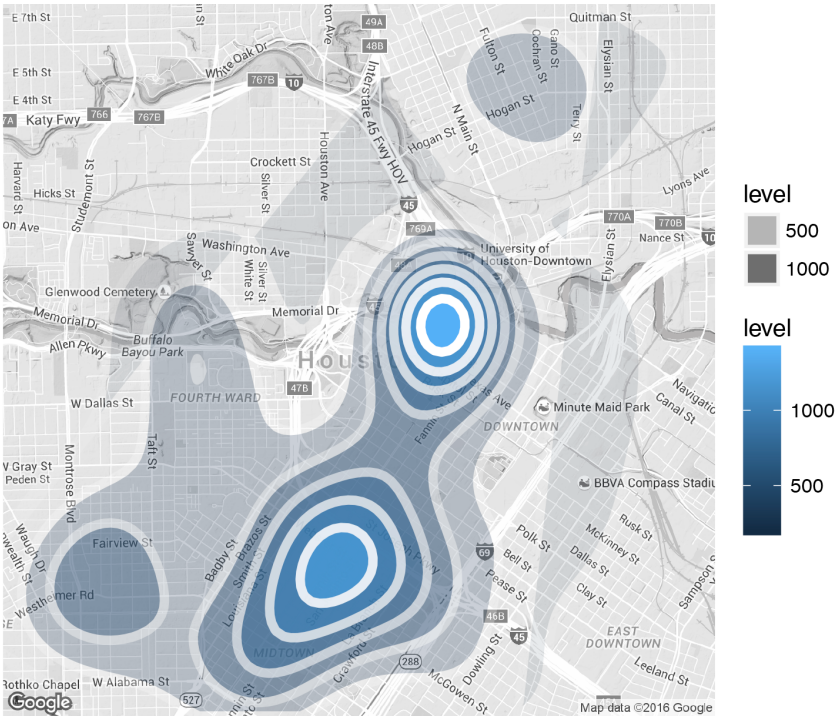
A més de `geom_point`, també estan disponibles altres tipus de capes de `ggplot2`, com `stat_bin2d`, que compta el nombre d'esdeveniments que tenen lloc en regions quadrades d'una grandària predefinida.

```
HoustonMap +
  stat_bin2d(
    aes(x = lon, y = lat, colour = offense, fill = offense),
    size = .5, bins = 30, alpha = 1/2,
    data = crimes.houston
  )
```




O es pot usar `stat_density2d`, que representa intensitats, per a identificar les zones de més criminalitat.

```
HoustonMap +
  stat_density2d(aes(x = lon, y = lat, fill = ..level.., alpha = ..level..),
                 size = 2, data = crimes.houston,
                 geom = "polygon"
)
```



## 2.3. Resum

`ggmap` és una extensió del paquet `ggplot2` per a representar informació cartogràfica. A més de funcions merament gràfiques, en disposa d'altres que permeten fer operacions d'interès geogràfic mitjançant consultes a API de proveïdors externs, com ara Google Maps.