# R for Data Science (I): Visualization

Alex Sanchez, Miriam Mota, Ricardo Gonzalo and Mireia Ferrer

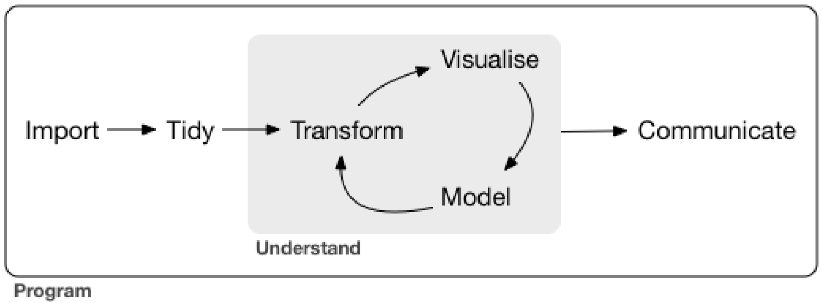
Statistics and Bioinformatics Unit. Vall d'Hebron Institut de Recerca

## Readme

## Outline: Data Exploration

- The Data Science Approach in R
- Data Visualization
- Data Transformation
- Exploratory Data Analysis

## Recall: The Data Science Approach in R



## Data Visualization

## Graphics in the `tidyverse`

- Traditionally graphics in R are relatively complicated because they are based in functions with many parameters.
- Improving a graphic or overimposing distinct plots is also a non-trivial task.
- The `tidyverse approach` provides a distinct way to draw plots which is, at the same time, **intuitive, flexible and powerful**.
- This is made possible because it implements the so-called *grammar of graphics* which was introduced by Hadley Wickam in his paper A layered grammar of graphics.

## The grammar of graphics

- Graphics are treated as a set of elements which can be combined to produce the final plot.
- The idea consists of *working with distinct layers* starting with a first one that sets the data to be plotted.
- Successive layers are added, for instance to change colors, add annotations, overimpose other plots, etc.

## The ggplot2 package

- This package implements the grammar of graphics within the tidyverse.
- The package does not belong to the standard R distribution, so it has to be installed.
  - This can be done when installing the tidyverse or separately (only for this package).
- Option 1:

```
install.packages('tidyverse')
```

- Option 2:

```
install.packages('ggplot2')
```

## Creating a plot with `ggplot2`

- A (gg)plot is obtained by combining several elements which produce distinct layers in the same plot:

1. The data to be represented, stored in a data frame.

## Creating a plot with `ggplot2`

- A (gg)plot is obtained by combining several elements which produce distinct layers in the same plot:

1. The data to be represented, stored in a data frame.
2. Geometric objects (`geoms`) which define the global aspect of the layer (bars, points, lines. . . ).

## Creating a plot with `ggplot2`

- A (gg)plot is obtained by combining several elements which produce distinct layers in the same plot:

1. The data to be represented, stored in a data frame.
2. Geometric objects (`geoms`) which define the global aspect of the layer (bars, points, lines. . . ).
3. Esthetic attributes (`aesthetics`), visual properties of the geoms such as *position*, *color of line*, *shapes of points*, etc.

## Creating a plot with `ggplot2`

- A (gg)plot is obtained by combining several elements which produce distinct layers in the same plot:

1. The data to be represented, stored in a data frame.
2. Geometric objects (`geoms`) which define the global aspect of the layer (bars, points, lines...).
3. Esthetic attributes (`aesthetics`), visual properties of the geoms such as *position*, *color of line*, *shapes of points*, etc.
4. A statistical summary of the data (`stats`) (*counting*, *smoothing*, ...). It is usually associated to the type of geom used.

## Creating a plot with `ggplot2`

- A (gg)plot is obtained by combining several elements which produce distinct layers in the same plot:

1. The data to be represented, stored in a data frame.
2. Geometric objects (`geoms`) which define the global aspect of the layer (bars, points, lines. . . ).
3. Esthetic attributes (`aesthetics`), visual properties of the geoms such as *position*, *color of line*, *shapes of points*, etc.
4. A statistical summary of the data (`stats`) (*counting*, *smoothing*, . . . ). It is usually associated to the type of geom used.
5. `facets` and `scales` allow to visualize different subsets of the data and control the representation in space.

## Creating a plot with `ggplot2`

- A (gg)plot is obtained by combining several elements which produce distinct layers in the same plot:

1. The data to be represented, stored in a data frame.
2. Geometric objects (`geoms`) which define the global aspect of the layer (bars, points, lines. . . ).
3. Esthetic attributes (`aesthetics`), visual properties of the geoms such as *position*, *color of line*, *shapes of points*, etc.
4. A statistical summary of the data (`stats`) (*counting*, *smoothing*, . . . ). It is usually associated to the type of geom used.
5. `facets` and `scales` allow to visualize different subsets of the data and control the representation in space.
6. Different elements can be included in the graph with the operator `+`.

## Creating a plot in practice

The basic steps to create a plot are:

1. Create a `ggplot` object providing the data and some `aesthetics`
2. Add one or more `geoms` using the + operator to define and shape the plot type.

## Example 1. The data

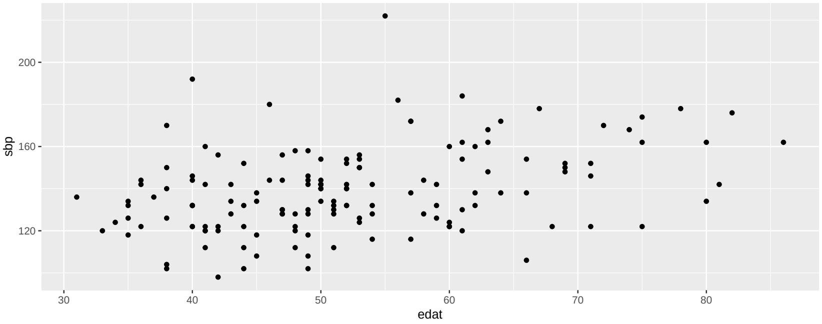
- First we need the data

```
library(readxl)
diabetes <- read_excel("datasets/diabetes.xls")
head(diabetes)
```

```
## # A tibble: 6 x 11
##   numpacie mort  tempsviu  edat   bmi edatdiag tabac   s
##      <dbl> <chr>    <dbl> <dbl> <dbl>    <dbl> <chr> <db
## 1        1 Vivo      12.4    44  34.2       41 No f~    1
## 2        2 Vivo      12.4    49  32.6       48 Fuma~    1
## 3        3 Vivo       9.6    49  22         35 Fuma~    1
## 4        4 Vivo       7.2    47  37.9       45 No f~    1
## 5        5 Vivo      14.1    43  42.2       42 Fuma~    1
## 6        6 Vivo      14.1    47  33.1       44 No f~    1
```

## Example 1. Build the plot

```
library(ggplot2)
ggplot(diabetes)+geom_point(aes(x=edat,y=sbp))
```



## Variations on the theme

- Calls to ggplot can be combined differently

```
ggplot(data=diabetes,aes(x=edat,y=sbp))+geom_point()
```

or

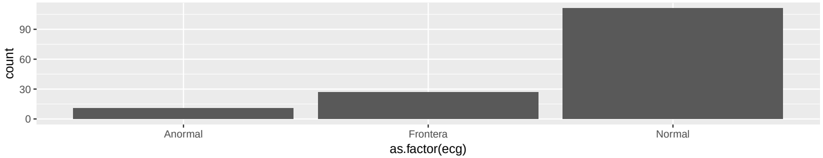
```
ggplot()+geom_point(data=diabetes,aes(x=edat,y=sbp))
```

## Aesthetics

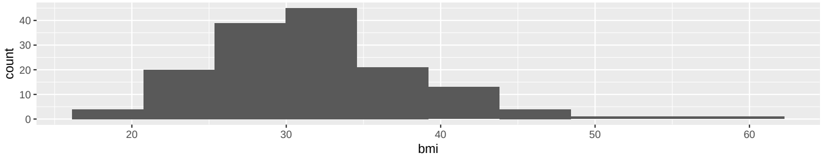
- In a ggplot aesthetic `aes ()` refers to *what we can see*, that is, visual properties of an object.
  - x, y: what goes on the axes
  - color: exterior color
  - ll: color of the interior
  - shape: shape of the points
  - linetype: type of line
  - size: size
  - alpha: transparency (1: opaque; 0: transparent)
- Each type of geometry accepts a subset of the possible options.
- One of the most used functions is to define groups through various aesthetics variables or directly with the option on "group".

## Example 2: Aesthetics

```
ggplot(diabetes)+geom_bar(aes(x=as.factor(ecg)))
```

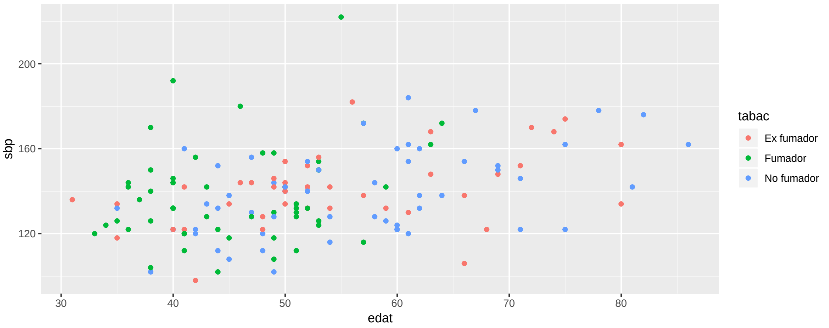


```
ggplot(diabetes)+geom_histogram(aes(x=bmi),bins=10)
```



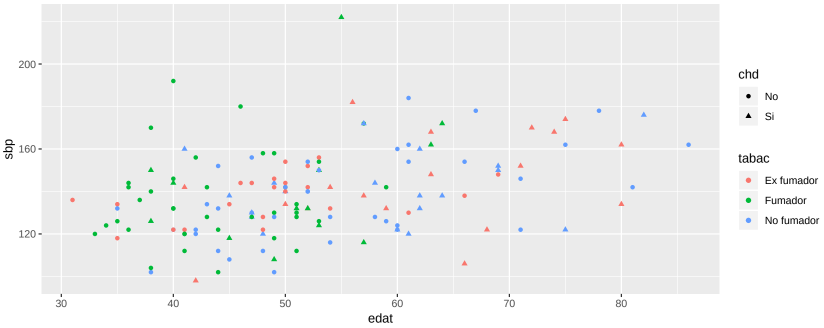
## Distinguishing between groups using `aes()` (1)

```
ggplot(diabetes)+geom_point(aes(x=edat,y=sbp, col=tabac))
```



## Distinguishing between groups using `aes()` (2)

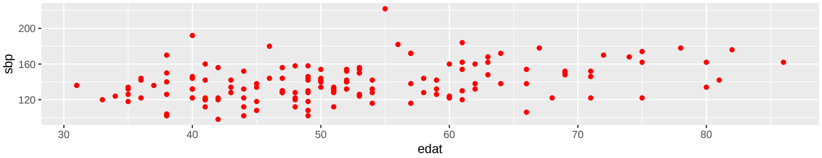
```
ggplot(diabetes)+geom_point(aes(x=edat,y=sbp, col=tabac, sh
```



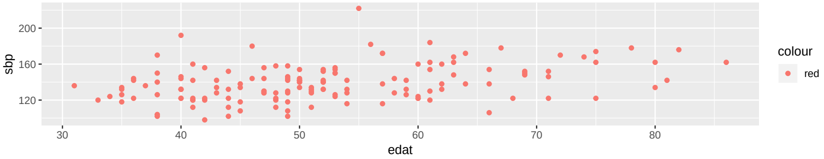
## aes properties that do not depend on variables

Notice the difference between these plots.

```r
ggplot(diabetes)+geom_point(aes(x=edat, y=sbp), col='red')
```



```r
ggplot(diabetes)+geom_point(aes(x=edat, y=sbp, col='red'))
```



## Geometric Objects

## Modifying plots by adding `geoms`

- Geometric objects are the actual marks we put on a plot. Examples include:
  - points (`geom_point`, for scatter plots, dot plots, etc)
  - lines (`geom_line`, for time series, trend lines, etc)
  - boxplot (`geom_boxplot`, for, well, boxplots!)
- A plot **must have at least one** `geom`; there is no upper limit.
  - You can add a geom to a plot using the + operator
- You can get a list of available geometric objects using the code below: `help.search("geom_", package = "ggplot2")`

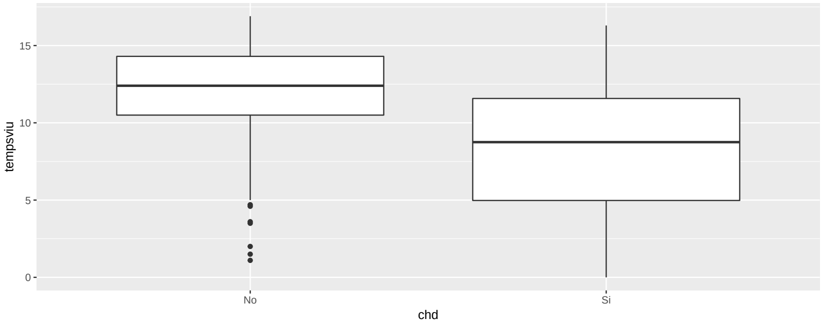
## Drawing plots incrementally

- In the console run the follow instructions one after the other

```
(p <- ggplot(diabetesF, aes(x=edat, y=sbp)))
(p<- p + geom_point())
(p<- p + geom_smooth(method='lm'))
```

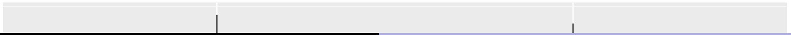
## Do not forget boxplots!

```
(p<- ggplot(diabetes, aes(x=chd, y=tempsviu)) +   geom_boxp
```



```
(p<- p+ ggtitle("Relation between temps viu and chardiac di
```

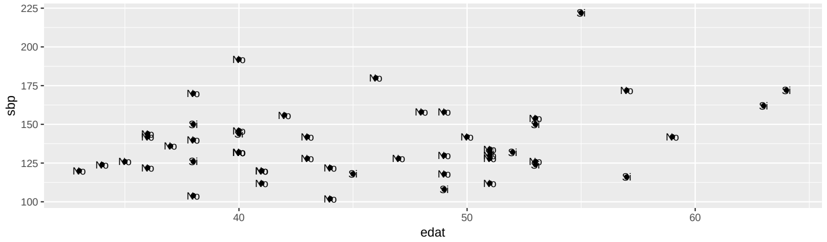
Relation between temps viu and chardiac disease



## Adding labels to your plot

- It is straightforward with the geom_text() which accepts a labels mapping.
- An alternative is using geom_label

```
ggplot(diabetesF, aes(x=edat, y=sbp))+ geom_point() + geom_
```



## ggplot extensions: the ggrepel package

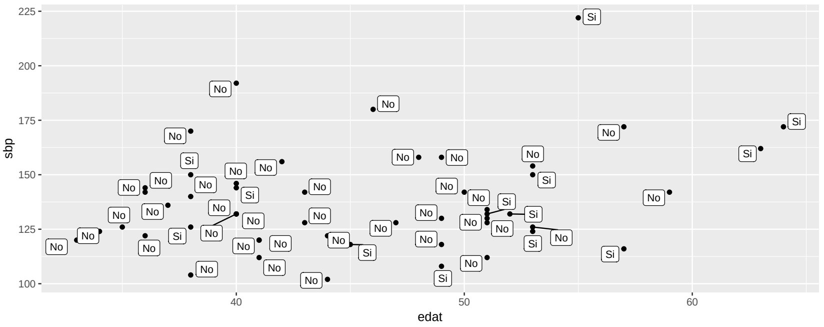
Use this package to avoid overlapping of labels and points

```r
install.packages("ggrepel")
```

```r
require(ggrepel)
ggplot(diabetesF, aes(x=edat, y=sbp))+ geom_point() +
  geom_label_repel(aes(label=chd), size = 3)
```

## ggplot extensions: the ggrepel package

```
## Loading required package: ggrepel
```



## Exercise I

- The data for this exercise, stored in the file `EconomistData.csv`.
- They consist of Human Development Index and Corruption Perception Index scores for several countries.

1. Create a scatter plot with CPI on the x axis and HDI on the y axis.
2. Color the points blue.
3. Map the color of the the points to Region.
4. Make the points bigger by setting size to 2
5. Map the size of the points to HDI.Rank

## Facets

## Facets

- Faceting is ggplot2 parlance for small multiples
- The idea is to create separate graphs for subsets of data
- ggplot2 offers two functions for creating small multiples:
  - `facet_wrap()`: define subsets as the levels of a single grouping variable
  - `facet_grid()`: define subsets as the crossing of two grouping variables
- Faceting facilitates comparison among plots, not just of geoms within a plot

## The housings dataset

For te following examples we will use a database on housing prices.

```
require(readr)
```

```
## Loading required package: readr
```

```
housing <- read_csv("datasets/landdata-states.csv")
```

```
## Parsed with column specification:
## cols(
##   State = col_character(),
##   region = col_character(),
##   Date = col_double(),
##   Home.Value = col_integer(),
##   Structure.Cost = col_integer(),
```

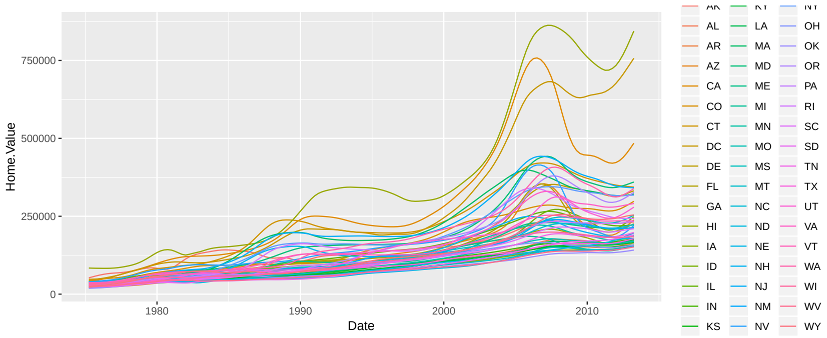
## What is the trend in housing prices in each state?

We can start with what we know how to do: map State to color.

```
p5 <- ggplot(housing, aes(x = Date, y = Home.Value))
p5 + geom_line(aes(color = State))
```

## Housing prices trends by states (1)

```
p5 <- ggplot(housing, aes(x = Date, y = Home.Value))
p5 + geom_line(aes(color = State))
```

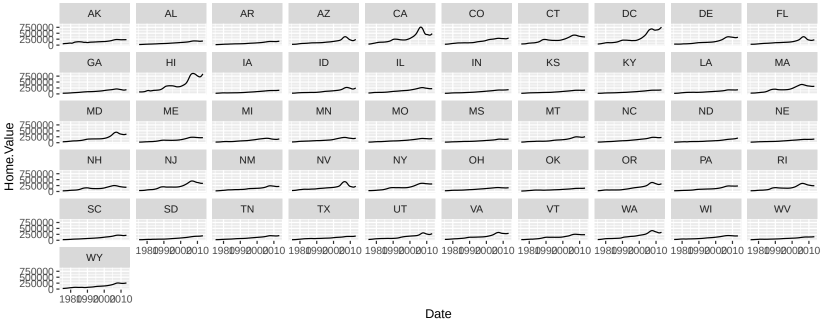


## Housing prices by states (2)

- Visibility of distinct trends depending on state can be improved if we plot each state in a separate graphic.

```
p5 <- ggplot(housing, aes(x = Date, y = Home.Value))
(p5 <- p5 + geom_line() +
   facet_wrap(~State, ncol = 10))
```

## Housing prices by states (2)



## Exercise

- Interpret the result of the following instructions:

```
 ggplot(mtcars,aes(x=wt,y=mpg))+geom_point()+
+ geom_smooth()+
+ facet_grid(as.factor(am)~as.factor(gear))
```

- What happens if we try to separate based on a continuous variable?
- How can this be solved?

## Statistical transforms

## Statistics

- Some plot types (such as scatterplots) do not require transformations-each point is plotted at x and y coordinates equal to the original value.
- Other plots, such as boxplots, histograms, prediction lines etc. require statistical transformations:
  - for a boxplot the y values must be transformed to the median and 1.5(IQR)
  - for a smoother smother the y values must be transformed into predicted values

## One stat per each geom

- Each geom has a default statistic, but these can be changed.

| geom | stat |
|---|---|
| geom_bar() | stat_count() |
| geom_col() | stat_identity() |
| geom_pol() | stat_identity() |
| geom_smooth() | stat_smooth() |

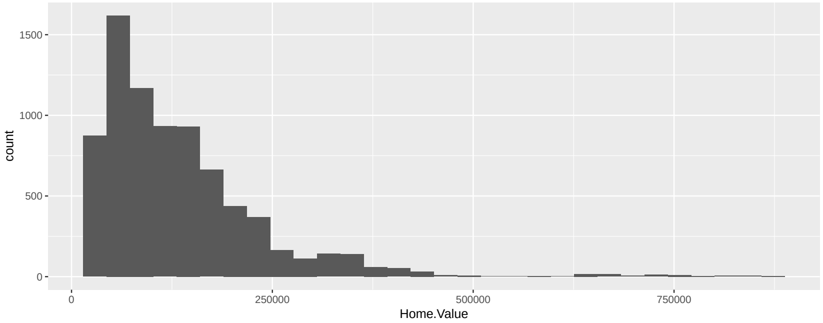
- The “stat” is an argument of the “geom” and the “geom” is an argument from the “stat”.
- Compare the outputs from:

```
args(geom_histogram)
args (stat_bin)
```

## Seeing the effect of stats (1: default)

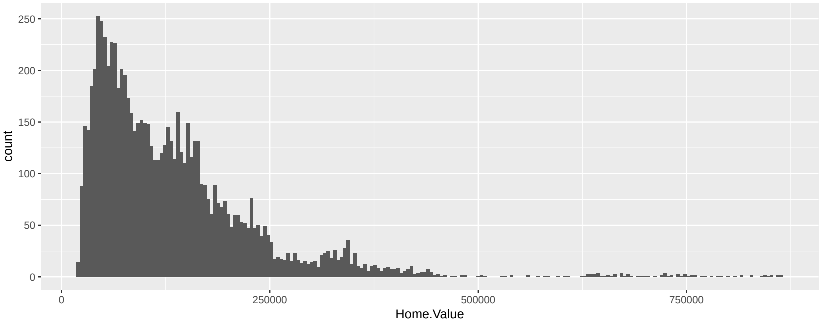
```
p <-ggplot(housing, aes(x = Home.Value))
(p<-p + geom_histogram())
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `
```



## Seeing the effect of stats (2: change values)

```
p<- ggplot(housing, aes(x = Home.Value))
(p<-p+geom_histogram(stat= 'bin', binwidth=4000))
```



## That's (not) all

- There are other things you can do with your plots.
- Start trying what we have done with your data or check the final exercise in this workshop:
    - A workshop on R graphics with ggplot2
- And do not forget the cheatsheet!:
    - ggplot2-cheatsheet-2.1-Spanish.pdf