The Language of Graphs: from Bertin to GoG to ggplot2

Michael Friendly

Psych 6135

http://euclid.psych.yorku.ca/www/psy6135/
Topics

• Idea: Graphs as visual language
  ▪ Early attempts at standardization of graphs
• Jacques Bertin: *Semiology of Graphics*
  ▪ Mapping of visual properties to data relations
• Graphics programming languages:
  ▪ Goal: power & elegance
• Lee Wilkinson: *Grammar of Graphics*
• Hadlely Wickham: ggplot2
Metaphor: Graphs as visual language

- Playfair, Guerry, Minard and others described their fundamental insight that **graphical displays** convey quantitative data more directly than **numbers**.

- **Playfair (1802)**
  - “Regarding numbers and proportions, the best way to catch the imagination is to speak to the eyes”

- **Minard (1861)**
  - “The aim of my carte figurative is ... to convey promptly to the eye the relation not given quickly by numbers requiring mental calculation.”
Émile Cheysson (1890) took this further:

- "When a law is contained in figures, it is buried like metal in an ore; it is necessary to extract it. This is the work of graphical representation.

- It points out the coincidences, the relationships between phenomena, their anomalies, and we have seen what a powerful means of control it puts in the hands of the statistician to verify new data, discover and correct errors with which they have been stained."
From ~ 1870—1910, statistical albums of official statistics on topics of population, trade, moral & political issues became widespread throughout Europe and the U.S.

- France: *Album de Statistique Graphique*: 1879-1899 (trade, commerce & other topics)
- USA: Census atlases: 1870/80/90--
- Switzerland: *Atlas graphique de la Suisse*: 1897, 1914
Need for standardization

• Beautiful graphics: Yes, but all separate designs
  ▪ Can anything be compared across countries?
• Émile Cheysson (1878)
  ▪ “The time will come when Science has to lay down general principles and decide on well-defined standards. We can no longer tolerate this sort of anarchy”
• International statistical meetings (ISI)
  ▪ 1852 (Brussels), 1857 (Vienna), 1869 (The Hague), 1872 (St. Petersburg), 1876 (Budapest) …
  ▪ Participants: Quetelet, Cheysson, Levasseur (France), Ernest Engel, Gustav von Mayr, Hans Schwabe (Germany), Francis Walker (U.S.), …
No consensus

• St. Petersburg (1872) resolutions:
  ▪ “The Congress accepts that it is not worth going into details about the choice of methods or facts for graphical representation”.
  ▪ “no strict rule can be imposed on authors, because the only real problem is that of applying the graphical method to data that is comparable”.
  
• Most of the debate had to do with thematic maps
  ▪ number of class intervals for a quantitative variable
  ▪ number and variety of shading colors

• Yet, the idea of a visual language had been accepted, along with the need for some theory of graphs

Standardize the data before the graphs!
Bertin: *Semiology of graphics* (1967)

- Defines a system of “grammatical elements” of graphs and relations among visual attributes that give **meaning** (semantics) from perceptual features
  - Planar variables: (x,y) coordinates
  - Retinal variables: shape, size, color, ...

![Image of Bertin](image.png)
Bertin: *Semiology of graphics*

- Defines a system of mapping of retinal variables to properties of data variables for perception of *relations*
  - Association (≡) – marks are perceived as *similar*
  - Selection (≠) – marks are perceived as *forming classes*
  - Order (O) – marks are perceived as *showing order*
  - Quantity (Q) – marks are perceived as *proportional*

- This is the first theory of graphs relating visual attributes (encoding) to perceptual characteristics (decoding).

- It comprises nearly all known graph and thematic map types in a *general system*
The retinal variables and relationship types can be implanted in various symbol types in the plane (X,Y)

<table>
<thead>
<tr>
<th>VARIABLES OF THE IMAGE</th>
<th>POINT</th>
<th>LINE</th>
<th>AREA (ZONE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>XY 2 DIMENSIONS OF THE PLANE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Z</td>
<td>SIZE</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>VALUE</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>OQ</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>OQ</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>O</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| DIFFERENTIAL VARIABLES | | | |
| TEXTURE | | | |
| | O | | |
| | X | | |
| COLOR | | | |
| ORIENTATION | | | |
| | | | |
| | | | |
| | | | |
| SHAPE | | | |
| | | | |
| | | | |
| | | | |
Visual variables & data characteristics

Visual variables differ in the kinds of information they can convey

<table>
<thead>
<tr>
<th>Visual Variables</th>
<th>Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Selective</td>
</tr>
<tr>
<td>Position</td>
<td><img src="image" alt="Position" /></td>
</tr>
<tr>
<td>Size</td>
<td><img src="image" alt="Size" /></td>
</tr>
<tr>
<td>Shape</td>
<td><img src="image" alt="Shape" /></td>
</tr>
<tr>
<td>Value</td>
<td><img src="image" alt="Value" /></td>
</tr>
<tr>
<td>Color</td>
<td><img src="image" alt="Color" /></td>
</tr>
<tr>
<td>Orientation</td>
<td><img src="image" alt="Orientation" /></td>
</tr>
<tr>
<td>Texture</td>
<td><img src="image" alt="Texture" /></td>
</tr>
</tbody>
</table>

(≠) (≡) (Q) (O)
Some recommendations

Various authors have used Bertin’s system to make recommendations for the best attributes to use with different symbol types.

<table>
<thead>
<tr>
<th>Visual Variables</th>
<th>Points</th>
<th>Lines</th>
<th>Areas</th>
<th>Best to show</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Shape</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><img src="image" alt="Points Shape" /></td>
<td><img src="image" alt="Lines Shape" /></td>
<td><img src="image" alt="Areas Shape" /></td>
<td>qualitative differences</td>
</tr>
<tr>
<td><strong>Size</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><img src="image" alt="Points Size" /></td>
<td><img src="image" alt="Lines Size" /></td>
<td><img src="image" alt="Areas Size" /></td>
<td>quantitative differences</td>
</tr>
<tr>
<td><strong>Color Hue</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><img src="image" alt="Points Color Hue" /></td>
<td><img src="image" alt="Lines Color Hue" /></td>
<td><img src="image" alt="Areas Color Hue" /></td>
<td>qualitative differences</td>
</tr>
<tr>
<td><strong>Color Value</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><img src="image" alt="Points Color Value" /></td>
<td><img src="image" alt="Lines Color Value" /></td>
<td><img src="image" alt="Areas Color Value" /></td>
<td>quantitative differences</td>
</tr>
<tr>
<td><strong>Color Intensity</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><img src="image" alt="Points Color Intensity" /></td>
<td><img src="image" alt="Lines Color Intensity" /></td>
<td><img src="image" alt="Areas Color Intensity" /></td>
<td>qualitative differences</td>
</tr>
<tr>
<td><strong>Texture</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><img src="image" alt="Points Texture" /></td>
<td><img src="image" alt="Lines Texture" /></td>
<td><img src="image" alt="Areas Texture" /></td>
<td>qualitative &amp; quantitative differences</td>
</tr>
</tbody>
</table>
Retinal variables allow several variables to be encoded. Bertin’s system provides a general framework for thematic mapping, allowing multiple variables to shown simultaneously in a single map.

Map for height, hair color and cephalic index distribution

For Bertin, the legend is a symbolic description of the coordinate system and the variables displayed.

Legend:
GEO: (x,y)
T, V, OR: ordered
Various maps of France, encoding quantitative and categorical variables in a wide number of different ways.

This semiology is **productive**, as is the semiology of language.

Allows one to think of new graphic encodings.
Decoding: Reading a graphic

How successful is a graph for transmitting information?
Bertin defines three stages for reading a graphic:

• **External**: What is the overall context?
  ▪ Graph title, axis labels

• **Internal**: What visual variables are used to represent the components in the graphic?
  ▪ points, lines, ...
  ▪ size, shape, color: hue, color: intensity, texture, ...

• **Relationships**:
  ▪ How are these components related?
  ▪ What questions can I ask of this graphic?
  ▪ What can I learn?

Research topic: Have there been any studies of this ordering in graph perception?
Questions a graph should answer:

- Elementary: find some specific value
- Intermediate: make comparisons, see a trend
- Overall: what is the general message or overall trend?

These ideas provided the beginnings of a theory of graphs related to graph perception.
Graph from the NY Times, Feb. 3, 2016

Reading tasks:
• Elementary: “How many guns were sold in January of 2013?”
• Intermediate: “What’s the trend in gun sales since President Obama was elected?”
• Overall: “What’s the overall trend in gun sales in America since the year 2000?”

From: https://medium.com/@karlsluis/before-tufte-there-was-bertin-63af71ceaa62
A data table: objects by characteristics

Encode each value by visual attributes

Both rows and columns are reorderable (≠≠)

Overall relation can be discovered by permuting rows, cols
The reorderable matrix

Permute rows and columns to put like with like

This is an early example of what I called “effect ordering” for data display

Interpret row/col order & clusters
A physical device implementing matrix reordering

This was used by Bertin and others in a large number of applied projects.

Bertin was to visual data analysis in France what Tukey was to EDA in N. America.
Bertifier: A web app implementing Bertin’s idea of the reorderable matrix
See: http://www.aviz.fr/bertifier

Table: Attitudes and attributes by country
Values encoded by size and shape
Sorted and grouped by themes and country regions

Watch: Youtube video of Bertifier, http://youtu.be/tJxAF_a_yBQ
Heatmaps are a re-invention of Bertin’s ideas:
- Cluster analysis to reorder rows/cols
- Shading cells to show some variable

This example shows a microarray analysis of 128 leukemia patients using 12625 genes.
- The goal is to distinguish two types of leukemia
- The shading variable is a z-score for how well a given gene distinguishes the two types.
- Several clusters of high association are discovered!

Image source: https://warwick.ac.uk/fac/sci/moac/people/students/peter_cock/r/heatmap/
There are many implementations of “heatmaps”
They differ importantly in the details of: clustering, shading scheme

This example shows a data set of 11 measures on 32 cars from the 1974 Motor Tends magazine
- Each variable was converted to z-scores
- The value was shaded using a bipolar color scheme
- Clusters of cars are slightly separated
- The very high and low values stand out

Menu-driven graphics provide a wide range of graph types, with options

What’s wrong with that?

**WYSIAYG:** What you see is **all** you get. No way to do something different
Not reproducible: Change the data → Re-do manually from scratch
Often designed by programmers with little sense of data vis
Programming languages: Power & elegance

- **CS view**: All programming languages can be proved to be equivalent (to a Turing machine)

- **Cognitive view**: Languages differ in:
  - expressive power: ease of translating what you want to do into the results you want
  - elegance: how well does the code provide a human-readable description of what is done?
  - extensibility: ease of generalizing a method to wider scope
  - learn-ability: your learning curve (rate, asymptote)
## Programming languages: Power & elegance

<table>
<thead>
<tr>
<th>Language</th>
<th>Features: Tools for thinking?</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FORTRAN</strong></td>
<td>Subroutines – reusable code</td>
</tr>
<tr>
<td></td>
<td>Subroutine libraries (e.g., BLAS)</td>
</tr>
<tr>
<td><strong>APL, APL2STAT</strong></td>
<td>N-way arrays, nested arrays</td>
</tr>
<tr>
<td></td>
<td>Generalized reduction, outer product</td>
</tr>
<tr>
<td></td>
<td>Function operators</td>
</tr>
<tr>
<td>Logo</td>
<td>Turtle graphics</td>
</tr>
<tr>
<td></td>
<td>Recursion, list processing</td>
</tr>
<tr>
<td><strong>Lisp, LispStat, ViSta</strong></td>
<td>Object-oriented computing</td>
</tr>
<tr>
<td></td>
<td>Functional programming</td>
</tr>
<tr>
<td>Perl</td>
<td>Regular expressions</td>
</tr>
<tr>
<td></td>
<td>Search, match, transform, apply</td>
</tr>
<tr>
<td><strong>SAS</strong></td>
<td>Data steps, PROC steps, BY processing</td>
</tr>
<tr>
<td></td>
<td>SAS macros, Output Delivery system</td>
</tr>
<tr>
<td><strong>R</strong></td>
<td>Object-oriented methods, tidyverse: dplyr, ggplot2, ...</td>
</tr>
</tbody>
</table>
Features:

- Based on Lisp, but tuned to young minds
- Turtle graphics: draw by directing a turtle, not by \((x,y)\) coordinates
  - Analytic geometry rests on a coordinate system.
  - Turtle geometry is "body syntonic": Tell turtle what to do.
- Data types:
  - words, lists, arrays, property lists
- Lists & list processing: inherited from Lisp, but with gentler syntax.
  - Lists are infinitely expandable & nestable.
- Recursion rather than iteration is the natural method to process lists
- Extensions:
  - multiple, animated turtles (sprites);
  - object-oriented programming (message passing) -> SmallTalk
Logo : Turtle graphics

Turtle primitives: forward, back, left, right, penup, pendown, ...

> to spiral :size :angle
if :size > 100 [stop]
forward :size
right :angle
spiral (:size + 2) :angle
end

> spiral 0 90
> spiral 0 91

Logo procedures: teach the turtle a new word

> to square :side
repeat 4 [fd :side rt 90]
end

> square 100

Recursive procedures:

> to square :side
repeat 4 [fd :side rt 90]
end

> square 100
Logo was more than just pretty pictures
It was graphics & mathematics for young minds: A language for learning

```
to Hilbert0 :turn :size
right :turn
forward :size
left :turn
forward :size
left :turn
forward :size
right :turn
end
```

Start with some basic shape
What happens if you replace each line with a smaller copy of the basic shape?
What happens if you continue this process?
What happens if you choose a different basic shape?
**Logo : Hilbert curves**

**Hilbert curve**: A continuous, space-filling fractal, of Hausdorff dimension 2

**Theorem** (Hilbert, 1891): The euclidean length of the n-th depth Hilbert curve, $H_n$ is $2^n - \frac{1}{2^n}$

**Proof** (by enumeration): Redefine forward to calculate total turtle path length

```logo
to forward.length :size
  make "total.length :total.length + :size
forward :size
end
```

```
to Hilbert :depth :turn :size
  if :depth = 0 [stop]
    right :turn
      Hilbert (:depth-1) -:turn :size
    forward :size
  left :turn
    Hilbert (:depth-1) :turn :size
  forward :size
  Hilbert (:depth-1) :turn :size
  left :turn
  forward :size
  Hilbert (:depth-1) -:turn :size
right :turn
end
```
Logo: Tower of Hanoi

Move N disks from one pole to another, with no disk ever resting on a disk smaller than itself.

to Hanoi :n :start :goal :spare
  if :n=0 [stop]
  Hanoi :n-1 :start :spare :goal
  move :n :start :goal
  Hanoi :n-1 :spare :goal :start
end

# move disks 1:n from START to GOAL
# are we done?
# move disks 1:n-1 from START to SPARE
# move disk n from START to GOAL
# move disks 1:n-1 from SPARE to GOAL

The Tower of Hanoi problem has an elegant solution in Logo. Change the ‘move’ instruction to render on screen or by a robot!
Graphics programming languages: SAS

• SAS: procedures + annotate facility + macros
  □ PROC GPLOT (x,y plots), PROC GCHART, PROC GMAP, ...
  □ Annotate: data set with instructions (move, draw, text, fonts, colors)
  □ Macros: Create a new, generic plot type, combining PROC steps and DATA steps.

```
data class;
input age sex ht wt;
datalines;
20 M 75 152
22 F 69 132
proc glm data=class;
  class sex;
  model wt = ht sex;
  output out=results p=predict r=resid;
results
proc gplot data=results;
  plot wt * ht = sex;
symbol1 ...
symbol2 ...
```

Why I gave up SAS: This works well, and is very powerful, but lacks elegance
SAS thinking: many languages

Output delivery system (ODS)
- template language

%macro language

proc iml
- matrix language, graphics

SAS/Graph:
- procs, Annotate language

Base SAS, SAS/STAT
- data step, proc steps
• Natural language:
  ▪ **Grammar/syntax**: What are the *minimal, complete* set of rules to describe *all* well-formed sentences?
    • John ate the big red apple ✔
    • John big apple red apple ate the ✗
  ▪ **Semantics**: How to distinguish meaning, nonsense, poetry in well-formed sentences?
    • Large green trucks carry garbage ✔
    • Colorless green ideas sleep furiously ??

• How to apply these ideas to graphics?
  ▪ **Grammar**: Algebra, scales, statistics, geometry, ...
  ▪ **Semantics**: Space, time, uncertainty, ...
  ▪ Needed: a complete *formal theory* of graphs & computational graphics language
Wilkinson: Grammar of Graphics

• A complete system, describing the components of graphs and how they combine to produce a finished graphic
  ▪ “The grammar of graphics takes us beyond a limited set of charts (words) to an almost unlimited world of graphical forms (statements)” (Wilkinson, 2005, p. 1).
  ▪ “… describes the meaning of what we do when we construct statistical graphics … more than a taxonomy”
  ▪ “This system is capable of producing some hideous graphics … This system cannot produce a meaningless graphic, however.”

• This is a general theory for producing graphs.
  ▪ the foundation of most modern software systems;
  ▪ not connected with a theory for reading graphs à la Bertin.
Wilkinson: Grammar of Graphics

• Components:
  ▪ **specification**: a formal language for composing graphs
  ▪ **assembly**: coordination of attributes
    • internal: a data structure for a graphical “object”
  ▪ **rendering**: producing a graphic on a display system
    • low level: device drivers for screen, PDF, PNG, SVG, ...

```
ELEMENT: point(position(x*y)),
COORD: rect(dim(1,2))
SCALE: linear(dim(1))
SCALE: linear(dim(2))
GUIDE: axis(dim(1), label("Sepa
GUIDE: axis(dim(2), label("Sepa
```

code  data structure  graphical output
Grammar of Graphics: Specification

- **Algebra**: combine variables into a data set to be plotted
  - cross (A*B), nest (A/B), blend (A+B), filter, subset, ...

- **Scales**: how variables are represented
  - categorical, linear, log, power, logit, ...

- **Statistics**: computations on the data
  - binning, summary (mean, median, sd), region (CI), smoothing
Grammar of Graphics: Specification

- **Geometry**: Creation of geometric objects from variables
  - Functions: point, line, area, interval, path, ...
  - Partitions: polygon, contour,
  - Networks: edge
  - Collision modifiers: stack, dodge, jitter

- **Coordinates**: Coordinate system for plotting
  - Transformations: translation, rotation, dilation, shear, projection
  - Mappings: Cartesian, polar, map projections, warping, Barycentric
  - 3D+: spherical, cylindrical, dimension reduction (MDS, SVD, PCA)
Grammar of Graphics: Specification

• **Aesthetics**: mapping of qualitative and quantitative scales to sensory attributes (extends Bertin)
  - **Form**: position, size, shape (polygon, glyph, image), rotation, ...
  - **Surface**: color (hue, saturation, brightness), texture (pattern, orientation), blur, transparency
  - **Motion**: direction, speed, acceleration
  - **Sound**: tone, volume, rhythm, voice, ...
  - **Text**: label, font, size, ...

• **Facets**: Construct multiplots (“small multiples”) by partitioning, blending or nesting

• **Guides**: Allow for reading the encodings of variables mapped to aesthetics
  - **scales**: axes, legend (labels: size, shape, color, ...)
  - **annotations** (title, footnote, line, arrow, ellipse, text, ...)
Wilkinson illustrates the GoG with a programming language (GPL: the Graphics Production Language)

GPL statements

- **DATA**: expressions that create variables to display from data sets
- **TRANS**: variable transformations prior to plotting (e.g., ranking the data points)
- **ELEMENT**: define graphical elements (e.g., points, lines, ...) and their aesthetic attributes (e.g., shape, color, ...) to use in the display
- **SCALE**: apply scale transformations to the plot (e.g., square root or log)
- **COORD**: select the coordinate system for use in the graphic (e.g., Cartesian, polar)
- **GUIDE**: guides to aid interpretation (axes, legends)
A simple scatterplot of the Iris data, points colored by species

```plaintext
DATA: x = "SepalLength"
DATA: y = "SepalWidth"
DATA: z = "Species"
TRANS: x = x
TRANS: y = y
ELEMENT: point(position(x*y), color(z))
COORD: rect(dim(1,2))
SCALE: linear(dim(1))
SCALE: linear(dim(2))
GUIDE: axis(dim(1), label("Sepal Length"))
GUIDE: axis(dim(2), label("Sepal Width"))
```

TRANS, SCALE, COORD and GUIDE all show the defaults & aren’t necessary here. The key one is ELEMENT, specifying points, positioned by (x*y) and colored by z

SPSS graphics now use GPL as the backend (syntax) for their graphics engine
A smoothed contour plot of birth rate vs. death rate for selected countries

**ELEMENT:** `point(position(birth*death), label(country))`
**ELEMENT:** `contour(position(smooth.kernel.density(birth*death)), color.hue())`
**GUIDE:** `form.line(position((0,0), (30,30)), label("Zero population growth"))`
**GUIDE:** `axis(dim(1), label("Birth rate"))`
**GUIDE:** `axis(dim(2), label("Death rate"))`

Wilkinson, *Grammar of Graphics*, Fig 1.1
The essential features of a graph are described by **ELEMENT**

- The geometrical objects (point, line, interval, ...) are specified within this
- Their visual properties (position, color) and statistical summaries are given as well

Some typical graph types:

<table>
<thead>
<tr>
<th>Graph</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>scatterplot</td>
<td><strong>ELEMENT</strong>: point((\text{position} (\text{d}*\text{r})))</td>
</tr>
<tr>
<td>line chart</td>
<td><strong>ELEMENT</strong>: line((\text{position} (\text{d}*\text{r})))</td>
</tr>
<tr>
<td>bar chart</td>
<td><strong>ELEMENT</strong>: interval((\text{position} (\text{d}*\text{r})))</td>
</tr>
<tr>
<td>hor. bar chart</td>
<td><strong>COORD</strong>: rotate((270))</td>
</tr>
<tr>
<td>clustered bar chart</td>
<td><strong>ELEMENT</strong>: interval.dodge((\text{position} (\text{d}*\text{r}), \text{color} (c)))</td>
</tr>
<tr>
<td>stacked bar chart</td>
<td><strong>ELEMENT</strong>: interval.stack((\text{position} (\text{summary.proportion} (r), \text{color} (c)) ))</td>
</tr>
<tr>
<td>histogram</td>
<td><strong>ELEMENT</strong>: interval((\text{position} (\text{summary.count} (\text{bin.rect} (y))) ))</td>
</tr>
</tbody>
</table>
Facets & frames

Tables of graphs:
- Facets: → graphs of subset
- Frames: → separate graphs

Linked micromap:
- Population density of US, divided in octiles
- States in each octile shown separately

GoG was a coherent language for specifying and producing nearly all known graphic forms.
• JSM 2017: Dinner with Lee Wilkinson, Howard Wainer, Paul Vellman, & others

• The great debate:
  - LW: The GoG is a complete theory, a formal mathematical model comprehending all graphs.
    "Beauty is truth, truth beauty,"--that is all Ye know on earth, and all ye need to know.
  - MF: There is more--
    - Implementation matters: translating a graphic idea into a finished graph should be facilitated by the language of graphic code.
    - A productive language for graphs should encompass the steps of data analysis
  - Pere Milán: A truly expressive graphic language should recommend the right graphic(s) to “get the message home”
Wickham: ggplot2

• ggplot2: *Elegant graphics for data analysis*
  ▪ a computational language for thinking about & constructing graphs
  ▪ sensible, aesthetically pleasing defaults
    • + themes: default, bw, journal, tufte, ...
  ▪ infinitely extendable
    • ggplot extensions: [https://exts.ggplot2.tidyverse.org/](https://exts.ggplot2.tidyverse.org/)

---

Facebook ggplot2 Theme
Implementation of GoG in R as layers of a graphic

- Basic layers:
  - Data,
  - Aesthetics (data → plot mapping)
  - Geoms (points, lines, bars, ...),
- Statistics: summaries & models
- Coordinates: plotting space
- Facets: partition into sub-plots
- Themes: define the general features of all graphical elements
ggplot2: data + geom = graph

- Every graph can be described as a combination of independent building blocks, connected by “+” (read: “and”)
  - data: a data frame: quantitative, categorical; local or database query
  - aesthetic mapping of variables into visual properties: size, color, x, y
  - geometric objects ("geom"): points, lines, areas, arrows, ...
  - coordinate system ("coord"): Cartesian, log, polar, map,
ggplot2: data + geom = graph

```r
ggplot(data=mtcars, 
       aes(x=hp, y=mpg, 
           color=cyl, shape=cyl)) + 
    geom_point(size=3)
```

In this call:
- **data=mtcars**: data frame
- **aes(x=, y=)**: plot X,Y variables
- **aes(color=, shape=)**: attributes
- **+ geom_point()**: what to plot
- the coordinate system is taken to be the standard Cartesian (x,y)
- a corresponding legend is automatically generated
Wow! I can really see something there.

How can I enhance this visualization?

Easy: add a `geom_smooth()` to fit linear regressions for each level of cyl

```r
ggplot(mtcars, aes(x=hp, y=mpg, color=cyl, shape=cyl)) +
  geom_point(size=3) +
  geom_smooth(method="lm", aes(fill=cyl))
```
ggplot2: GoG -> graphic language

• The implementation of GoG ideas in ggplot2 for R created a more expressive language for data graphs
  ▪ **layers**: graph elements combined with “+” (read: “and”)

```r
ggplot(mtcars, aes(x=hp, y=mpg)) +
  geom_point(aes(color = cyl)) +
  geom_smooth(method ="lm") +
```

▪ **themes**: change graphic elements consistently
ggplot2 facilitates graphical thinking by making a clear separation among:

- mapping data variables to plot features (`aes()`);
- geometric objects (`geom_()`);
- statistical summaries (`stat_()`).
Aesthetic attributes in the `ggplot()` call are passed to `geom_()` layers

Other attributes can be passed as constants (size=3, color="black") or with `aes(color=, ...)` in different layers

This plot adds an overall loess smooth to the previous plot

```r
ggplot(mtcars, aes(x=hp, y=mpg)) +
  geom_point(size=3, aes(color=cyl, shape=cyl)) +
  geom_smooth(method="lm", aes(color=cyl, fill=cyl)) +
  geom_smooth(method="loess", color="black", se=FALSE)
```
All the graphical attributes of ggplot2 are governed by themes – settings for all aspects of a plot.

A given plot can be rendered quite differently just by changing the theme.

If you haven’t saved the ggplot object, `last_plot()` gives you something to work with further.

```
last_plot() + theme_bw()
```
ggplot2: themes

Built-in ggplot themes provide a wide variety of basic graph styles

Other packages provide custom themes, or you can easily define your own
Facets divide a plot into separate subplots based on one or more discrete variables.

```r
plt <-
ggplot(mtcars, aes(x=hp, y=mpg, color=cyl, shape=cyl)) +
  geom_point(size=3) +
  geom_smooth(method="lm", aes(fill=cyl))
plt + facet_wrap(~gear)
```

Syntax: `facet_wrap(rowvar ~ colvar)`
ggplot2 provides a prototype system for implementing new geoms, stats, themes, ... Many of these are listed at https://exts.ggplot2.tidyverse.org/
ggplot2 provides a prototype system for implementing new geoms, stats, themes, ... Many of these are listed at https://exts.ggplot2.tidyverse.org/
The wide range of extensions indicates the power of ggplot2 as a general framework for data graphics.
Data science treats statistics & data visualization as parts of a larger process:

- Data import: text files, data bases, web scraping, ...
- Data cleaning → “tidy data”
- Model building & visualization
- Reproducible report writing
These ideas inspire a larger view of data analysis and graphics based on tidy principles.
Summary

• Graphical developers in the Golden Age recognized the idea of “graphic language,” but could not define it.

• Bertin first formalized the relations between graphical features (“retinal variables”), data attributes (O, Q, ≠, ≡), and “reading levels”

• Wilkinson, in GoG, created a comprehensive syntax and algebra to define any graph

• Wickham, in ggplot2, created an expressive language to ease the translation of graphic ideas into plots.

• Tidyverse ideas place data analysis & graphics within a communication-oriented, reproducible research framework.