Report

GP Team 2015

February 9, 2016

Contents

1 Introduction

What is a graphic? How can we succinctly describe a graphic? And how can we create the graphic that we have described?

These are important questions for the field of statistical graphics. One way to answer these questions is to develop a grammar . A good grammar will allow us to gain insight into the composition of complicated graphics, and reveal unexpected connections between seemingly different graphics.

A grammar provides a strong foundation for understanding a diverse range of graphics. A grammar may also help guide us on what a wellformed or correct graphic looks like, but there will still be many grammatically correct but nonsensical graphics.

This is easy to see by analogy to the English language: good grammar is just the first step in creating a good sentence. Grammar makes language expressive. A language consisting of words and no grammar (statement = word) expresses only as many ideas as there are words. By specifying how words are combined in statements, a grammar expands a language's scope.

In other hand grammar of graphics is a tool that enable us to concisely describe the components of graphic. Such a grammar allow us to move beyond named graphics scatterplot and gain insight into the deep structure that underlies statistical graphics .
The power of the grammar is illustrated with a selection of examples that explore different components and their interactions .

2 Literature review

2.1 Data Visualization

2.1.1 Definition

Data visualization is the presentation of data in a pictorial or graphical format. For centuries, people have depended on visual representations such as charts and maps to understand information more easily and quickly.

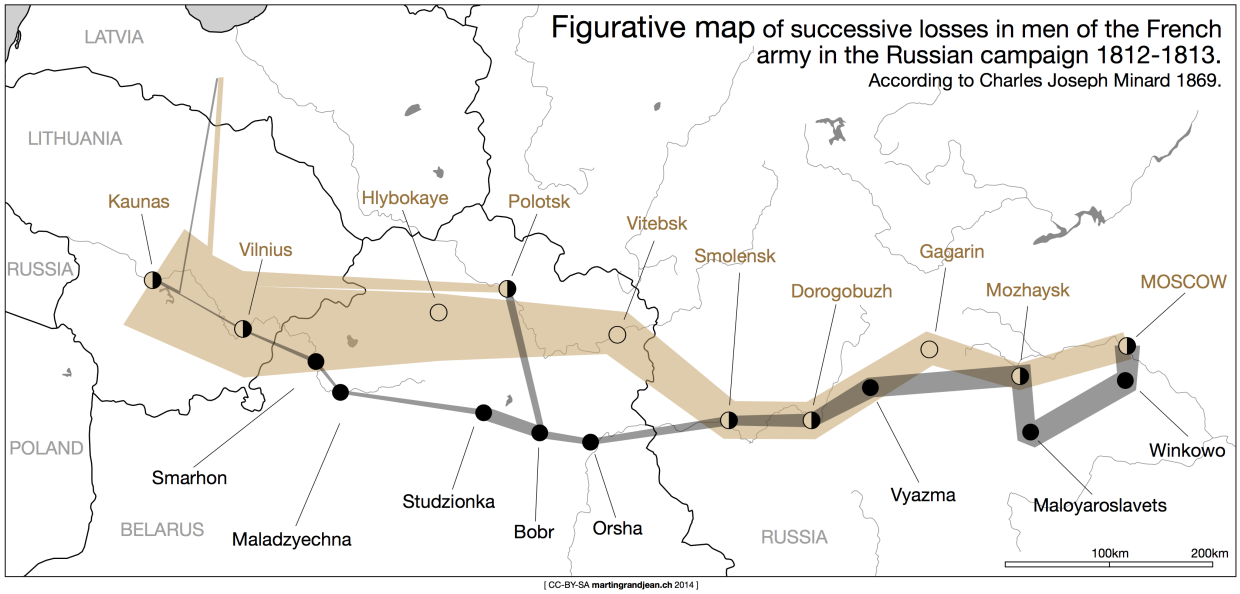
Because of the way the human brain processes information, it is faster for people to grasp the meaning of many data points when they are displayed in charts and graphs rather than poring over piles of spreadsheets or reading pages and pages of reports.

2.1.2 Data visualization importance

Visualizations help people see things that were not obvious to them before.

A spreadsheet cannot visually represent the information due to data presentation limitations, would spend hours searching among thousands of rows and columns of data with still no concrete answer about the relationship between two factors.

Figure 1: The path of Napoleon's troops across the Russian Empire of Alexander I



3 Grammar of graphics

3.1 What is the meaning of Grammar?

the whole system and structure of a language or of languages in general, usually taken as consisting of syntax and morphology (including inflections) and sometimes also phonology and semantics.
So if we think about the graphics as a language from the perspective of the grammar we should put in our minds those rules (grammar) that manage that manage these sentences or the components of the language (charts).

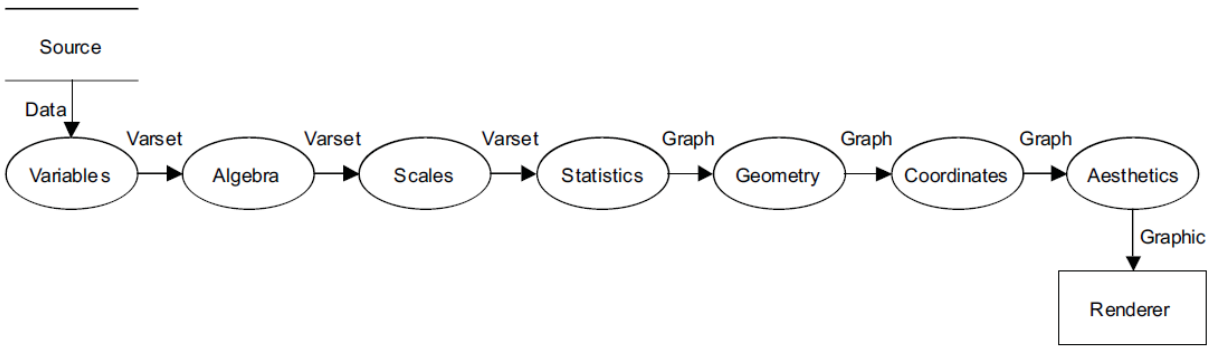
3.2 What is the Grammar of Graphics?

The regime of constructing the graphs depending on predefined rules, Hence The grammar of graphics takes us beyond a limited set of charts (words) to an almost unlimited world of graphical forms (statements).

3.3 Grammar of graphics features

- Orthogonal set of features describes all common charts, Virtually all uncommon charts.
- Language is flexible enough to
  - describe our known chart types
  - describe unknown chart types

Figure 2: Grammar of graphics layers



3.4 Brief explanation about GoG layers

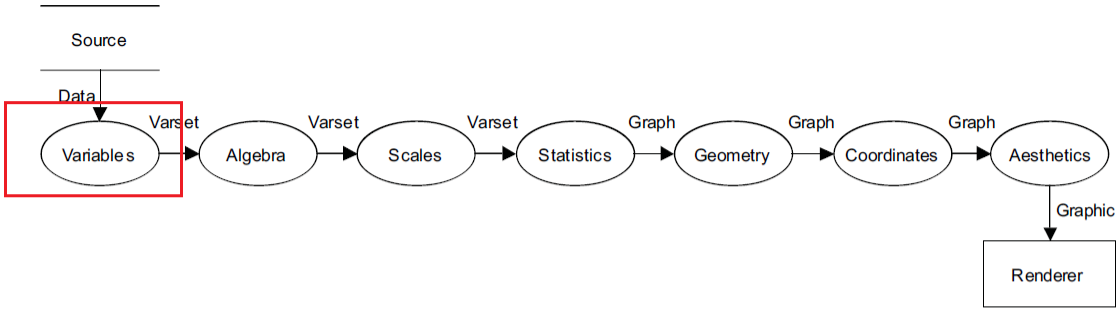
3.4.1 Variables

A variable gives us a method for associating a concept like income with a set of data. Variables are used together with operators in the syntactical portion of our specification language (e.g., Response & Gender).

A variable refers to a column in a rectangular cases-by-variables file. Many statistical packages assume rows are cases or observations that presents instances or samples of a variable represented by the column.

There is nothing in our definition of a variable which requires it to represent a row or column, however. The only requirement is that the variable mapping function return a single value in the range for every index.

Figure 3: Variables Layer



CaseID	Response
1	Frequently
2	Not Sure
3	Frequently
...	...
3834	Rarely
3835	Infrequently

CaseID	Gender
1	Male
2	Female
3	Male
...	...
3834	Male
3835	Female

3.4.2 Algebra

Algebra deals with restoring and balancing sets of variables in order to create the specification for the frames in which graphs are embedded.

- Algebra Syntax
  - Symbols
    A symbol is used to represent an entity operated on by an algebra. The symbols in varset algebra are varsets (e.g., Special variable (1), which represents the unity variable. Its range is one unity value).
  - Operators
    An operator is a method for relating symbols in an algebra. There are three operators in the graphical system, called cross, nest, and blend.
    * Cross (*)
      Cross joins the left varset with the right to produce a set of tuples.

$$A * B : (s,t) \longrightarrow A(s) \cap B(t)$$

$$V_{A*B} = \{(s,t), \forall s \in \forall_A, \forall t \in V_B\}$$

    * Nest (/)
      Nest produces a varset that looks like the result of a cross but the domain of a nest is not a subset of the domain of a cross, however.
      Even though they look the same, the tuples of a nest are colored or tagged and the tuples of a cross are not.

$$A/B : (s,t) \longrightarrow A(s) \cap B(t)$$

$$V_{A/B} = \{(s,t), \forall s \in \forall_A, \forall t \in V_B : A(s) \cap B(t) \neq \phi\}$$

    * Blend (+)
      Blend produces a union of varsets, Notice that we lose information about which value came from which column.

$$A + B : (s,t) \longrightarrow A(s) \cup B(t)$$

$$V_{A+B} = V_A \cup V_B$$

    * Shorthand
      For convenience, we occasionally make use of two operator aliases that reduce the complexity of graphics algebra expressions. The first is "exponentiation"

$$X^2 = X * X$$

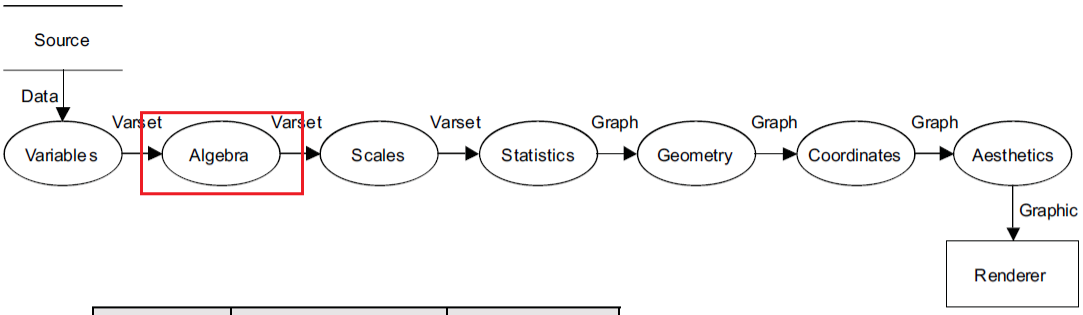
$$X^3 = X * X * X$$

The second is a "dot-cross" operation (assuming the arguments conform)

$$(X + Y) \bullet (A + B) = X * A + Y * B$$

Since our extract-variables process made two separate variables, we need to combine them into one variable so that the rows are associated with each other. We use the algebraic cross function to accomplish this. (e.g. cross(Response, Gender))

Figure 4: Algebra Layer

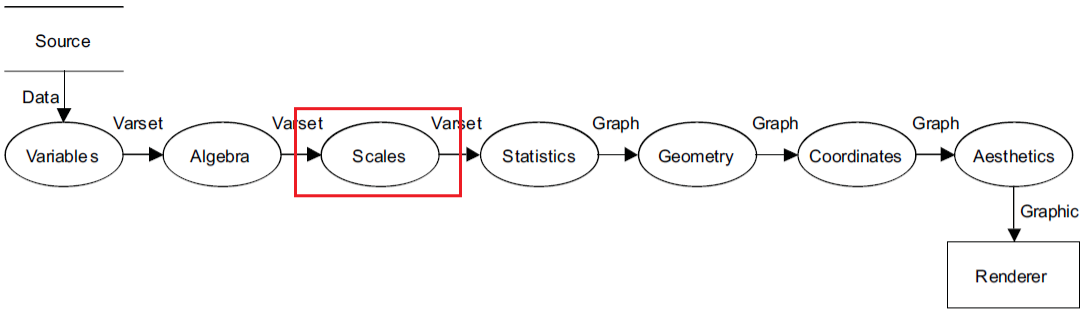


CaseID	Response	Gender
1	Frequently	Male
2	Not Sure	Female
3	Frequently	Male
...	...	...
3834	Rarely	Male
3835	Infrequently	Female

3.4.3 Scales

The categorical variables Response and Gender must have a defined order. A categorical scale transformation associates the values of a categorical variable with the set of integers. It may do this through a variety of possible methods. Any number of automatic categorical scale transformations can be devised, such as natural (alphabetical) order, relative frequency of the answer, or even the length of the string.
We will use a custom ordering that maps $Rarely \to 1, Infrequently \to 2, Occasionally \to 3, Frequently \to 4, NotSure \to 5$.
This is derived from the ordering of the frequency implied by the responses, where the Not Sure case isn't comparable and is arbitrarily placed last. We will use an alphabetical ordering for the Gender variable. The result appears in Figure 5.

Figure 5: Scales Layer



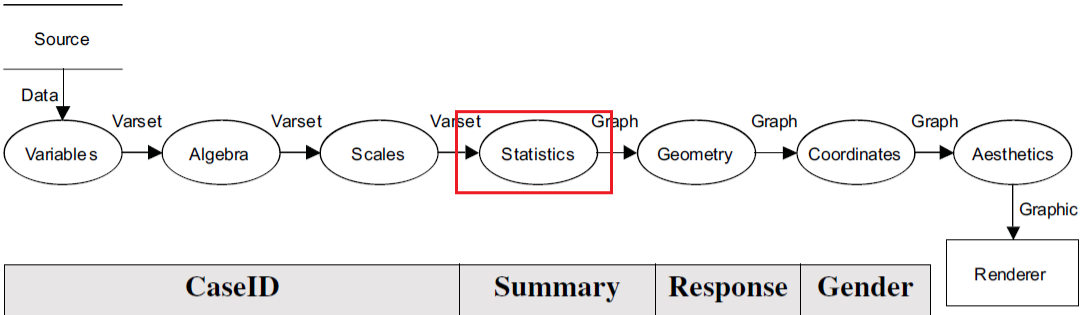
CaseID	Response	Gender
1	4	2
2	5	1
3	4	2
...	...	...
3834	1	2
3835	2	1

3.4.4 Statistics

Our system will ultimately make one graphical element per row of a varset. A pie chart of the data we have so far would therefore have 3,835 slices in it. This data layout isn't what we usually have in mind when we make a pie. In order to reduce the number of cases, we will need to do some statistics on the data.

A pie chart uses the pie wedges to represent the proportion of the whole for some category. The summary.proportion function is an aggregating statistic. It aggregates over columns constituting a whole pies domain in this case, each gender is a whole. The aggregated variable is Response. Figure 6 shows the result, using the summary.proportion() statistical method.

Figure 6: Statistics Layer



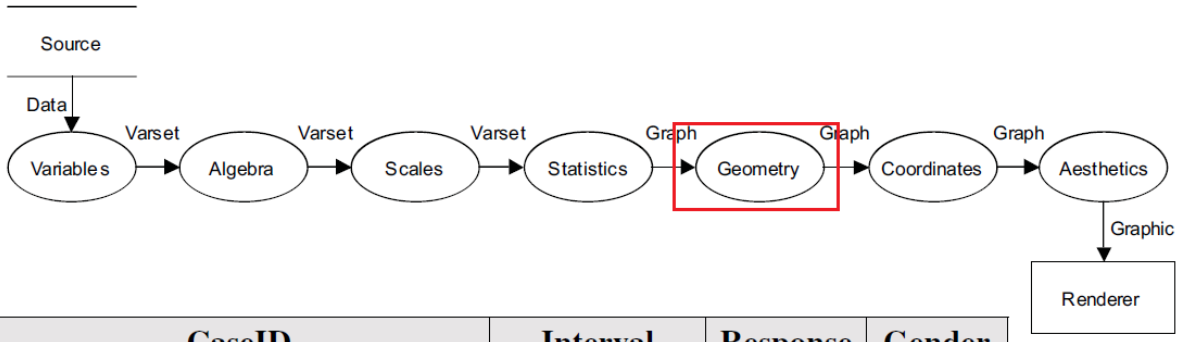
CaseID	Summary	Response	Gender
{females responding "rarely"}	0.08	1	1
{females responding "infrequently"}	0.11	2	1
{females responding "occasionally"}	0.17	3	1
{females responding "frequently"}	0.32	4	1
{females responding "not sure"}	0.32	5	1
{males responding "rarely"}	0.30	1	2
{males responding "infrequently"}	0.15	2	2
{males responding "occasionally"}	0.10	3	2
{males responding "frequently"}	0.07	4	2
{males responding "not sure"}	0.38	5	2

3.4.5 Geometry

The grammar of graphics has a variety of different geometric graphing operations.
In this case, we will use the interval graph.
The interval() function converts an interval into a (usually rectangular) geometric object with a length proportional to the interval and a nonzero (usually constant) width.

The interval.stack() function cumulates the intervals constructed by the interval geometric object. In other words, stack is a modifier method on the interval class.
Each interval is modified by incrementing its lower and upper bounds by the upper bound of the preceding interval in a sequence. Figure 7 shows the result.

Figure 7: Geometry Layer(interval.stack())



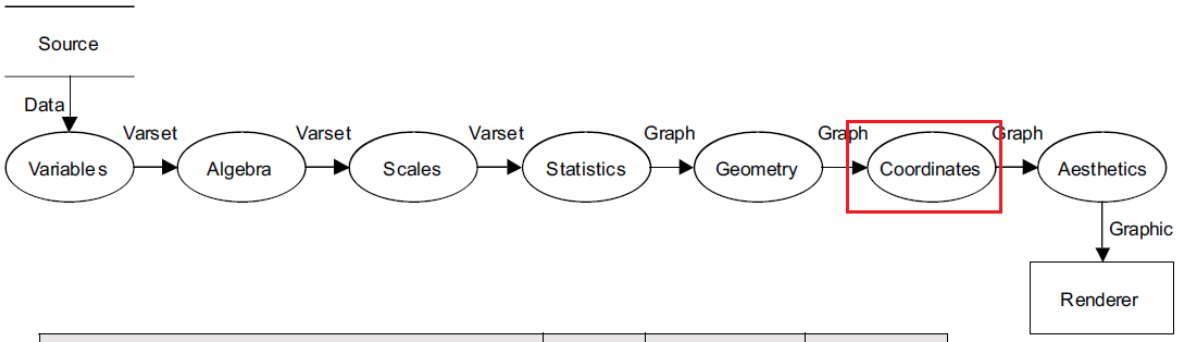
CaseID	Interval	Response	Gender
{females responding "rarely"}		1	1
{females responding "infrequently"}		2	1
{females responding "occasionally"}		3	1
{females responding "frequently"}		4	1
{females responding "not sure"}		5	1
{males responding "rarely"}		1	2
{males responding "infrequently"}		2	2
{males responding "occasionally"}		3	2
{males responding "frequently"}		4	2
{males responding "not sure"}		5	2

3.4.6 Coordinates

To make pie wedges, we apply a polar transformation to the shapes that were produced from the geometry. We send (x, y) to (r, theta) for the shapes.
Our program might accomplish this by digitizing the edges and sending each of the points through a simple transformation.

Figure 8 shows the result of the polar coordinate transformation.
We note that this transformation applies only to the position aesthetic. It is possible in the grammar of graphics to apply coordinate transformations to other aesthetics as well, e.g., COORD: polar.theta(aesthetic(color)).

Figure 8: Coordinates Layer



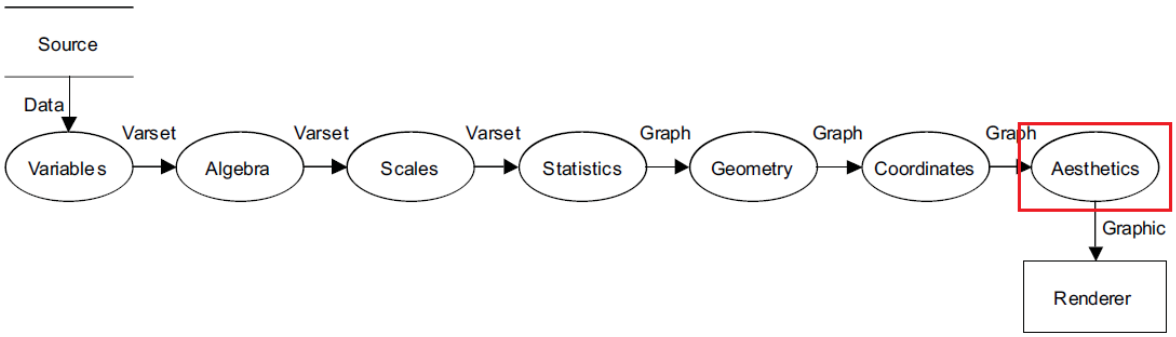
CaseID	Slice	Response	Gender
{females responding "rarely"}		1	1
{females responding "infrequently"}		2	1
{females responding "occasionally"}		3	1
{females responding "frequently"}		4	1
{females responding "not sure"}		5	1
{males responding "rarely"}		1	2
{males responding "infrequently"}		2	2
{males responding "occasionally"}		3	2
{males responding "frequently"}		4	2
{males responding "not sure"}		5	2

3.4.7 Aesthetics

We have created a graph, but a graph is a mathematical abstraction. We cannot sense mathematical abstractions. We must therefore give this abstraction perceivable form. Aesthetic functions translate a graph into a graphic.
Figure 9 adds three aesthetic functions position, color, and label.

The output of each aesthetic function is a graphic, which is a set of drawing instructions for a renderer. The position() aesthetic translates the tuples corresponding to the vertices of the objects to be drawn into a renderer coordinate system (usually pixel-based). The color() aesthetic function produces an index to a color table so that the renderer can apply the appropriate color. We have colored the wedges in the table to signify this. The label() aesthetic function associates strings with values.

Figure 9: Aesthetics Layer (position(Response,Gender), color(Response), label(Response))

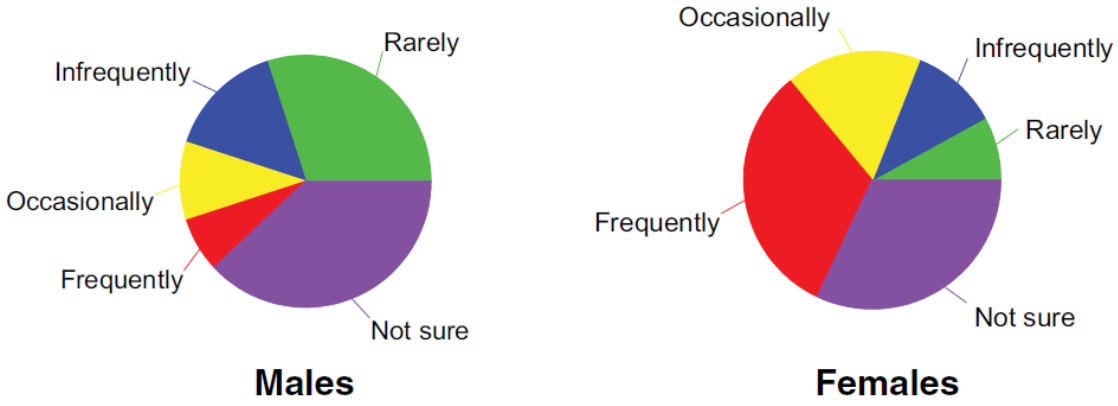


CaseID	Slice	Response	Gender
{females responding "rarely"}		1	1
{females responding "infrequently"}		2	1
{females responding "occasionally"}		3	1
{females responding "frequently"}		4	1
{females responding "not sure"}		5	1
{males responding "rarely"}		1	2
{males responding "infrequently"}		2	2
{males responding "occasionally"}		3	2
{males responding "frequently"}		4	2
{males responding "not sure"}		5	2

Figure 10 shows a possible result of a renderer working on the data in Figure 9, together with the formal specification for the graph.

DATA: response = Response
DATA: gender = Gender
SCALE: cat(dim(1),
values("Rarely","Infrequently","Occasionally","Frequently","Not Sure"))
SCALE: cat(dim(2), values("Female","Male"))
COORD: rect(dim(2), polar.theta(dim(1)))
ELEMENT:
interval.stack(position(summary.proportion(response*gender)),
label(response), color(response))

Figure 10: Output



4 Comparative study among GoG libraries

Table 1: Comparative Study

Library	ggplot2	ggvis	ggd3	vega
Implementation	Fully Implemented	Fully Implemented	Partially Implemented	Partially Implemented GOG
Language	R	R	JavaScript	JavaScript
Issue	No interactivity.	Limited Interactivity	-Have fixed set of geoms. -No interactivity.	Set of static geometrics.
Description	It takes care of many of the fiddly details that make plotting a hassle (like drawing legends) as well as providing a powerful model of graphics that makes it easy to produce complex multi-layered graphics.	-Declaratively describe data graphics with a syntax similar in spirit to ggplot2. -Create rich interactive graphics that you can play with locally in Rstudio or in your browser.	Stop development	Vega is a declarative format for creating, saving, and sharing visualization designs.

5 Our Approach

Design & Implement a library that is based on Grammar of graphics to be the baseline of continuous contribution in the field of data visualization.

1. GoG library.
    - built with javascript runs on Node.js environment
2. Application Layer.
    - built on top of Electron.
3. Package Manager.
    - which will manage application extensions and plugins.

6 Team members playrolls

6.1 Raafat Sobhy

- List & implement Coordinate systems
    - Number line
    - Cartesian coordinate system

- Polar coordinate system
- Cylindrical and spherical coordinate systems
- Homogeneous coordinate system

6.2 Sherif Embarak

- adding full documentation to all function
  - Example :

```
# Project ||GoGLib
#Module ||Stat Methods
#Author ||Sherif Emabrak
#Description ||Given a list of n nodes (n assumed even) represented by the
#              indices (1, 2, 3, ...) , the join method returns a list of edges
#              represented by the indices (1, 1 + n/2), (2, 2 + n/2), ...This method
#              is useful for blending two varsets of nodes because it pairs the
#              first node in the first column with the first node
#              in the second, and so on.
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

- partial implementation for link, bin, region and summary functions
  - link functions :
    - join.

```
join = (input_array...) ->
  n = input_array.length
  result = []
  for key,value of input_array
    result.push([value,value+n/2])
  result
```

    - sequence.

```
sequence = (input_array...) ->
  result = []
  for key,value of input_array
    temp=key
    key=value
    value=input_array[parseInt(temp)+1]
    result.push([key,value])
  result
```

- bin functions :
- summary functions :
  * sum.

```
sum = (input_array...) ->
  total=0;
  for count in input_array
    total+=count
  total
```

  * mean.

```
sum = (input_array...) ->
  total=0;
  for count in input_array
    total+=count
  total


mean = (input_array...) ->
  (sum input_array...)/input_array.length
```

  * mode.

```
mode = (input_array...) ->
  counter = 1
  max = 0;
  result = input_array[0]

  input_array.sort()

  for i in [1...input_array.length-1]
    if(input_array[i]==input_array[i+1])
      counter++
      if(counter > max)
        max=counter
        result = input_array[i]
      else
        counter = 1

  result
```

  * median.

```
median = (input_array...) ->
  size = input_array.length
  posetion = 0
  result = 0
  if size%2 isnt 0
    posetion = (size-1)/2
    result = input_array[posetion]
  else
    posetion = size / 2
    result = (input_array[posetion]+
    input_array[posetion-1])/2
  result
```

* log.

```
log = (input)  ->
  Math.log(input)
```

- region functions :
- smooth functions :
  * sum.
  * mean.
  * mode.
  * median.
  * log.
- importing csv files to js objects
  * Import CSV files and treat each row as JSON object
  * Show data in the browser as table
  * Add Type, Tag, or etc for table or column.

6.3 Ahmed Fouad

- List predefined figures in GoG book
- implementing geometric components (point, line, circle)

6.4 Yusuf Mohamed

- Package Manager for application layer.

* A cli utility tool which fetches & removes plugins (extensible piece of code) from npm and places it in specific folder. once the application bootstraps it will check all the existing plugins in the folder to add them which will provide feature rich application.
* why use a new directory for plugins?
  - The idea behind using a new directory other than `node_modules` is that the application will lookup plugins in it, and application shouldn't lookup plugins in the `node_modules` folders, as it will be another problem to differentiate between plugins and other node modules. So, the separation is needed to help the visualization application lookup plugins in a folders containing plugins only.
* Works on all major platforms like Windows, Linux and Mac.
* Took in consideration various plugin manager design like apm (`atom package manager`) and Brackets plugins.
* why didn't we use apm or brackets plugin manager?
  - APM : no documentation or how-to, limits the plugins to be hosted on github only.
  - Brackets : limits the developer with a specific structure and specific keywords to make his plugin up and running.
* Usage
  - Add Or Update a plugin.

    vispack install plugin_name

  - Remove a plugin

    vispack remove plugin_name

– Application Layer

* Made with electron formerly known as NW.js which is a cross platform application to distribute the application on all major platforms and photon kit for styling.
* Interprocess communication `ipc` to coordinate activities among different program process (main process and rendering process) via sending and receiving messages between the mentioned processes.

7 Gantt Chart

Figure 11: Gantt chart

8 Deliverable

Partial implementation of GoG library which will make us able to generate the following charts

- Scatter Plot
- Area Chart
- Line Chart
- Bar Chart
- Histogram
- Parallel Coordinates

References

[1] Vega `http://vega.github.io/`

[2] ggplot2 `http://ggplot2.org/`

[3] ggvis `http://ggvis.rstudio.com/`

[4] ggd3 `http://benjh33.github.io/ggd3/`