Visualization of the performance of baseball players. Final report.

Andrii Zakharchenko

May 2020

1 Visualisation overview

To visualise a performance of baseball players I chose a Trellis visualisation method. The Trellis visualisation allows to split data into multiple plots by certain fields. In case of statistics of baseball players using trellis layout will help to visualise development of multiple statistics in time in one figure.

I implemented 3 types of trellis visualization. The first type, shown in fig. 1, splits different performance statistics into separate plots, but at the same time shows development of all player for certain statistic in a same plot.

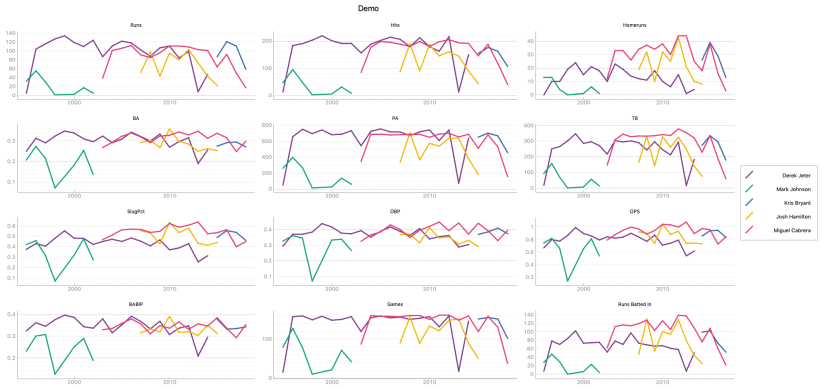

Figure 1: Trellis layout, split by statistics

In the figure above I show the comparison of 5 different players across 12 different statistics. We can see that each individual player has a unique color and that all players may have different set of points on x-axis. It is also easy to change a layout and add different number of statistics, as shown in fig. 2, we can split layout in 4 columns and add 9 statistics.

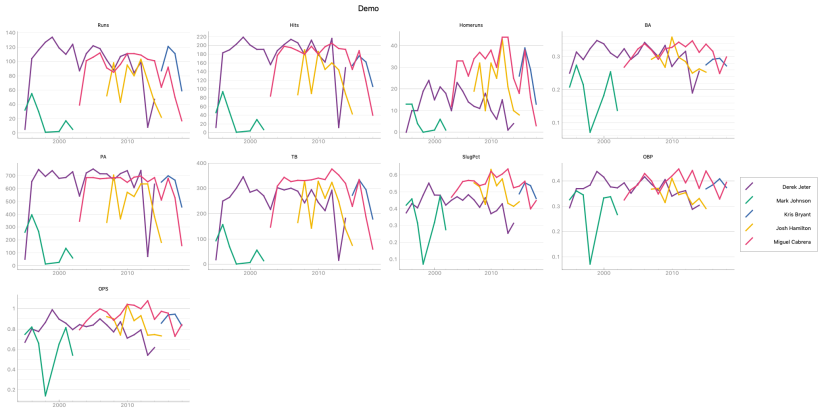

Figure 2: Trellis layout, 2 by 4

The second type of visualisation splits data by players and statistics. In fig. 3, each column represents an individual player and his statistics. However, with a growing number of statistics it's hard to place them on a screen, so I added a scrolling bar, however this cannot be seen in the image.

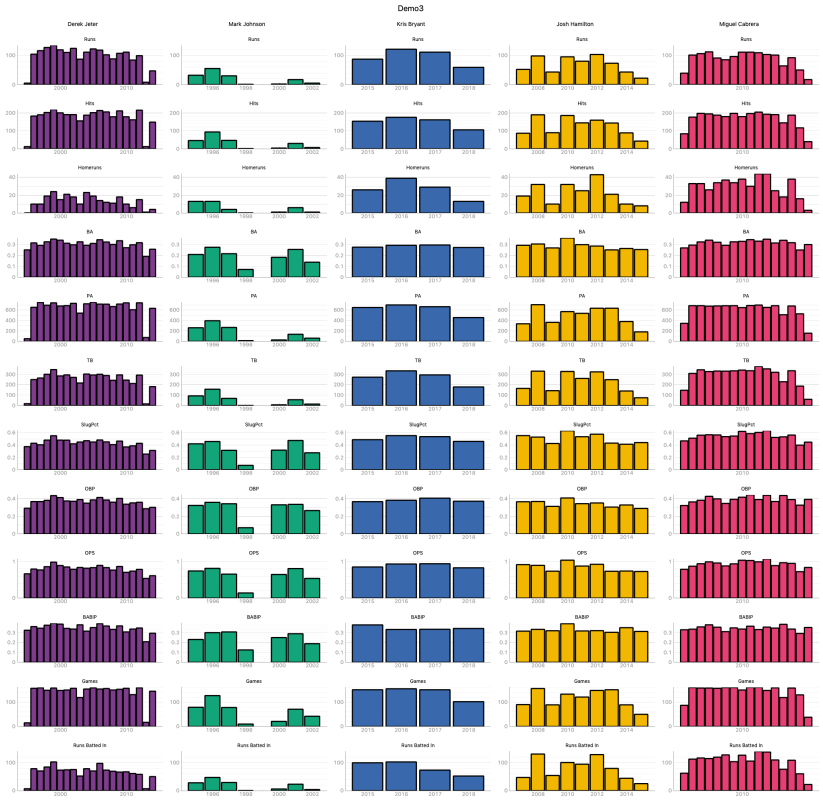

Figure 3: Full trellis by players and statistics

It is also possible to easily change a bar plot to a usual plot in this type of visualisation, as shown in fig. 4.

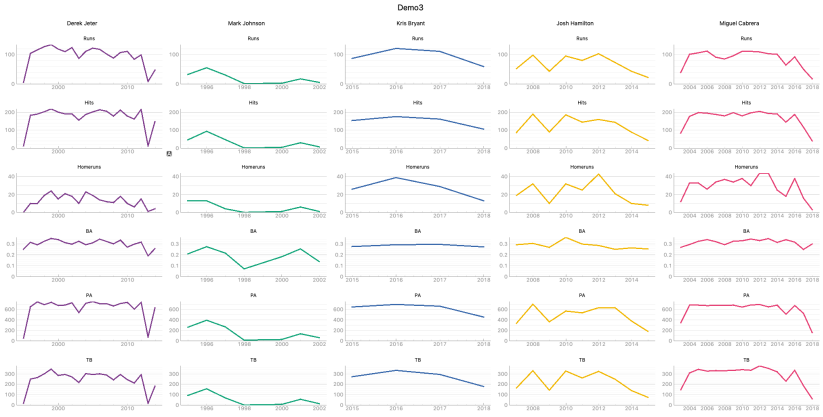

Figure 4

Also, important note here is that the y-axis of plots on a same line is ranged according to the maximum value of the given statistic across all players. This allows for more conviniante comparison of different players.

The third type of visualisation allows to compare players in a given year using a slider to select a year from range. Two figures below, show comparison of players for two selected years 1996 and 2010. In these plots, x-axis contains a name of player, instead of a year.

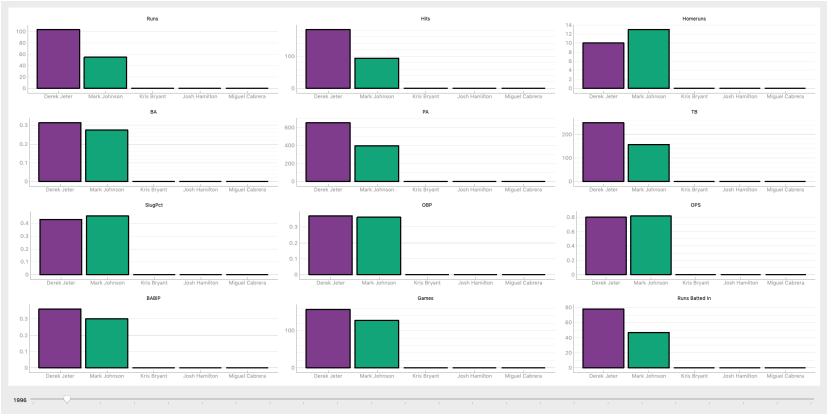

Figure 5: Comparison by year, 1996

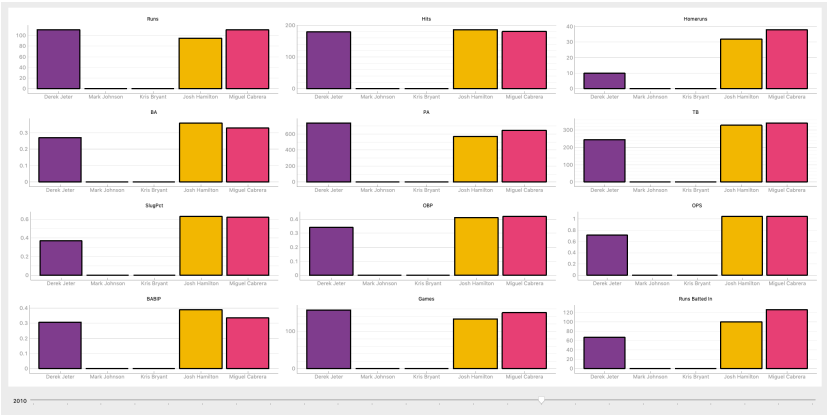

Figure 6: Comparison by year, 2010

2 Implementation overview

This visualisation was implemented in Python 3.8 and it is based on a **pyqt-graph** library. This library, in its turn, is based on a **PyQt** library and designed

for more convenient work with graphs in Qt environment. The fact that this library is based on Qt library allows to combine it with standard Qt widgets, such as scroll bars and sliders, which I used for this implementation. I also use **pandas**, **numpy** for data managment and **palettable** library, which is a simple library that contains a collection of different colors.

The **pyqtgraph** library contains a standard implementation of plots, however I re-implemented parts that are responsible for plotting data. My implementation consists of two modules **plot.py** and **vis.py**. The first one contains implementation of simple graphs such as plot and barplot and other. The **vis.py** module contains implementation of each Trellis visualisations.

The **plot.py** module contains next classes:

1. *CustomGraph* - a container for a graph object which contains other graphical objects such as Items and axes.
2. *BaseCustomItem* - base class for Item objects. Item objects can represent a single shape in a plot (e.g. bar in bar plot or line in plot).
3. *CustomBarItem* - object that implements a drawing of bars in bar plot
4. *CustomPlotItem* - object that implements a drawing of lines in plot
5. *CustomLegend* - a legend object to display legend in the first type of visualisation
6. *CustomLegendItem* - an Item object to draw symbol in a legend.
7. *Slider* - a slider widged for the third type of visualisation
8. *CustomAxis* - a custom axis object to display axis with names of players
9. *Style* - a data object that contains a style parameters for visualisation

The **vis.py** module contains these classes:

1. *TrellisStyle* - a data object to define a style of the visualisation (e.g. line width, palette etc)
2. *Trellis* - a base class for Trellis visualisation
3. *TrellisByStats* - implementation of first type of visualisation
4. *TrellisByPlayerAndStats* - implementation of second type of visualisation
5. *TrellisWithSlider* - implementation of third type of visualisation