Big Data Project

Analysis of LEGO brick prices over the years with PySpark

Chengcheng Guo (Matrikel-Nr.183090)

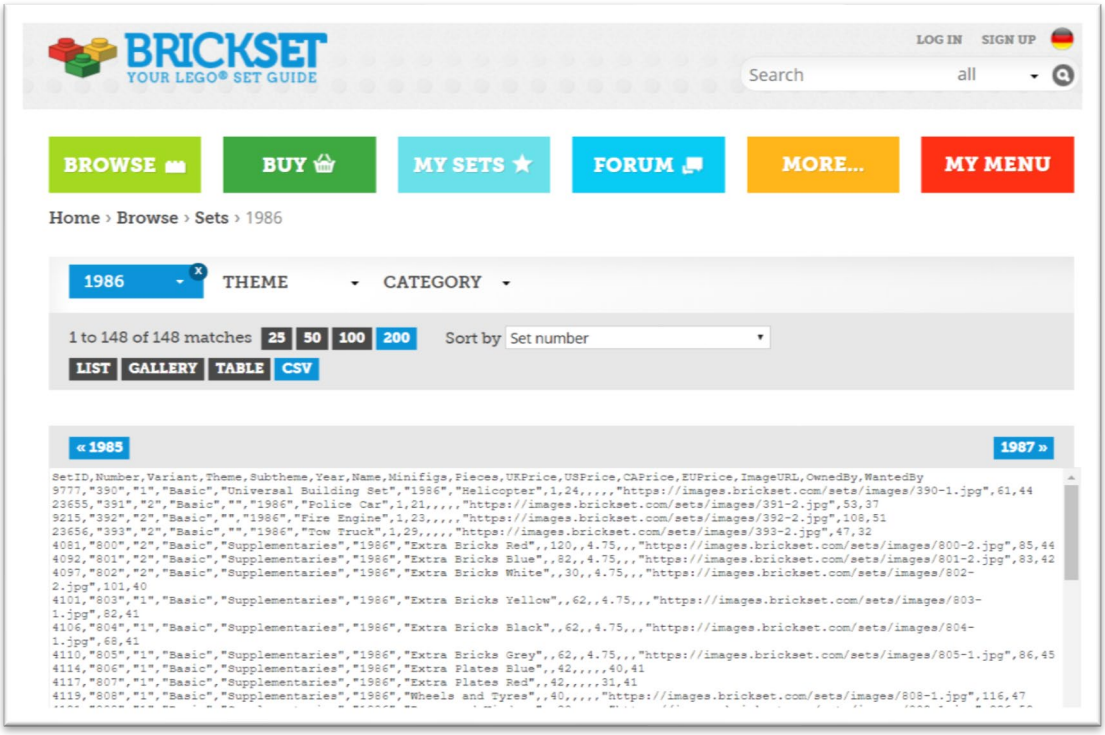
26.1.2020

Analysis of LEGO brick prices over the years with PySpark

1.Introduction

In my analysis, I want to get some information from data of LEGO sets prices, to evaluate the price history of LEGO sets, and the Mass of pasts LEGO sets.

Data is like:



I get my data from https://brickset.com/. This website offers historic retail prices of LEGO sets. I download the datasets. It is a series of csv files, contains informations about past launched LEGO sets from year 1949 to 2020. (informations refers to the unique ID of sets, Variant, Theme, launch year, how many pieces the set contains, how much is the set and a URL links to an image of the set). Notice, the data from 1949 to 1960 are partly incomplete, which means in this period, some 'price' or 'name' of sets are lacking. So, in my later analysis, I will clean the dataset or focus on the data after 1960's.

2.Data cleaning

I use PySpark to combine the separate data every separate year to a single DataFrame format table, further I filter the data by 2 steps to make the dataset cleaner.

```
#READ DATASET TO DATAFRAME AND DATA CLEANING
df=sqlContext.read.format('com.databricks.spark.csv').options(header='true',inferschema='true'). \
load('sortbyyear/*.csv')
dfnot01=df.filter(df.USPrice.isNotNull())
dfnot2=dfnot01.filter(dfnot01.Pieces>=25)#.show(25)
```

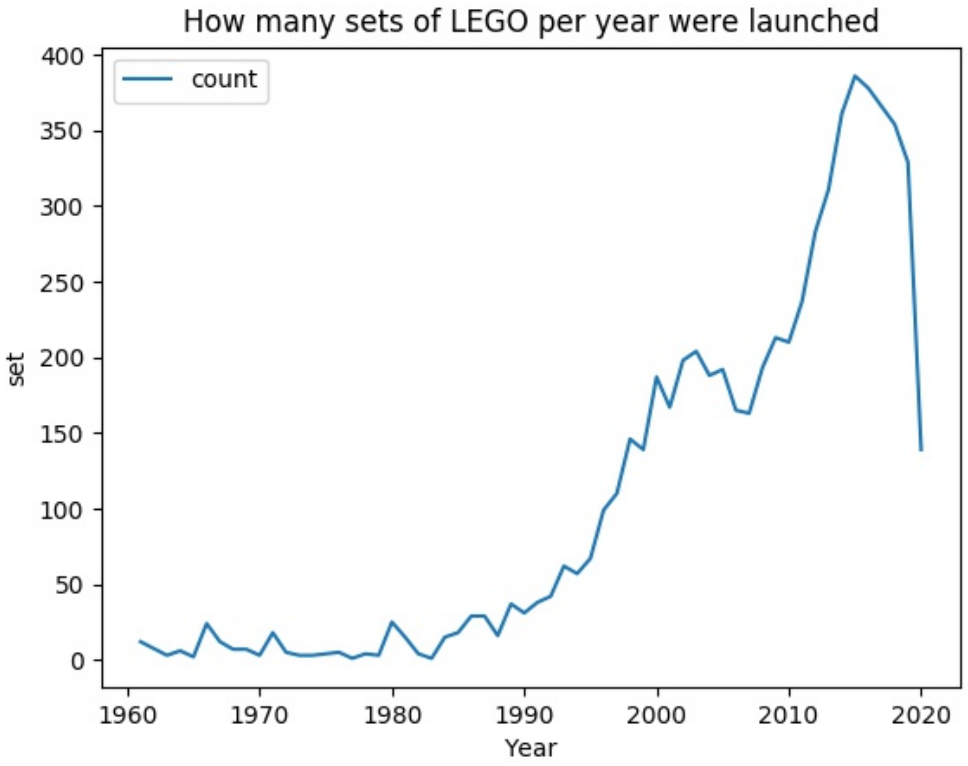
I filter the rows, which the set has no price. Moreover I filter the sets which has less than 25 pieces. There have been many promotional sets over the years which have very few pieces but carry a higher price because of their promotional status. These can range from keychains to individual minifigs to seasonal items. They are not representative of the typical price of a LEGO brick and therefore should not be included in the evaluation.

3.Analysis, findings and plots

1. Afterwards I calculate “How many sets of LEGOs per year were launched” with Spark.

```
30  #每年共多少套how many sets per year launch##############################
31  df_data1=dfnot0.groupby('Year').count().orderBy('Year')#.show(25)
32  df_data1.write.csv('legodc_data1.csv')
```

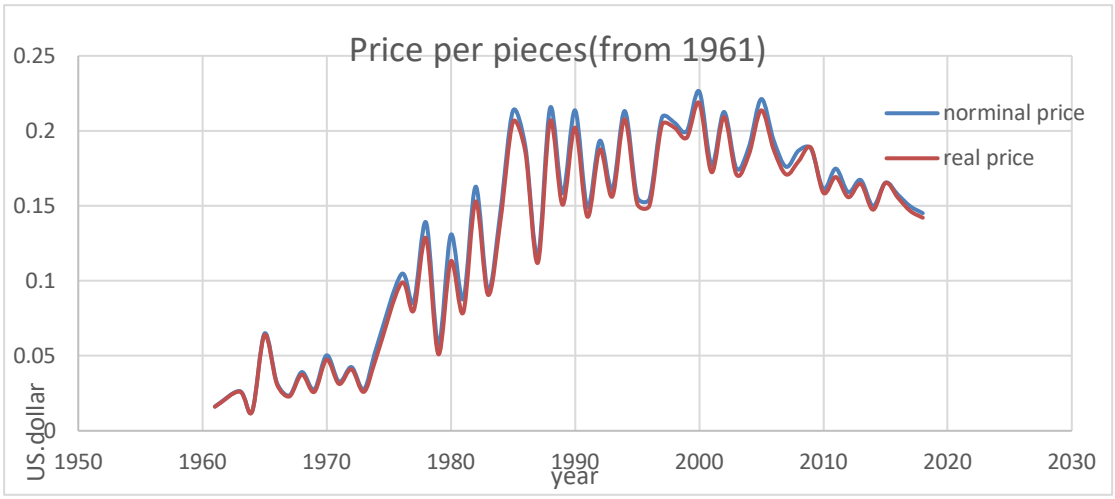
In the last few years, LEGO has had a Renaissance. It has obtained major licenses and broadened its appeal. There have also been changes to the manufacturing process that allows LEGO to expand its product lines and release more sets each year. Below is the chart of the number of sets released each year from 1962-2020 till now:



2. “how much costs a single brick per year”

```
#每年共多少套how many sets per year launch#################################
df_data1=dfnot0.groupby('Year').count().orderBy('Year')#.show(25)
df_data1.write.csv('legodc_data1.csv')
```

I then calculated the average price per piece of each set of each year and adjusted it for inflation. Afterwards I put it into graph form. *Figure 1* shows the results (prices are in US dollars).

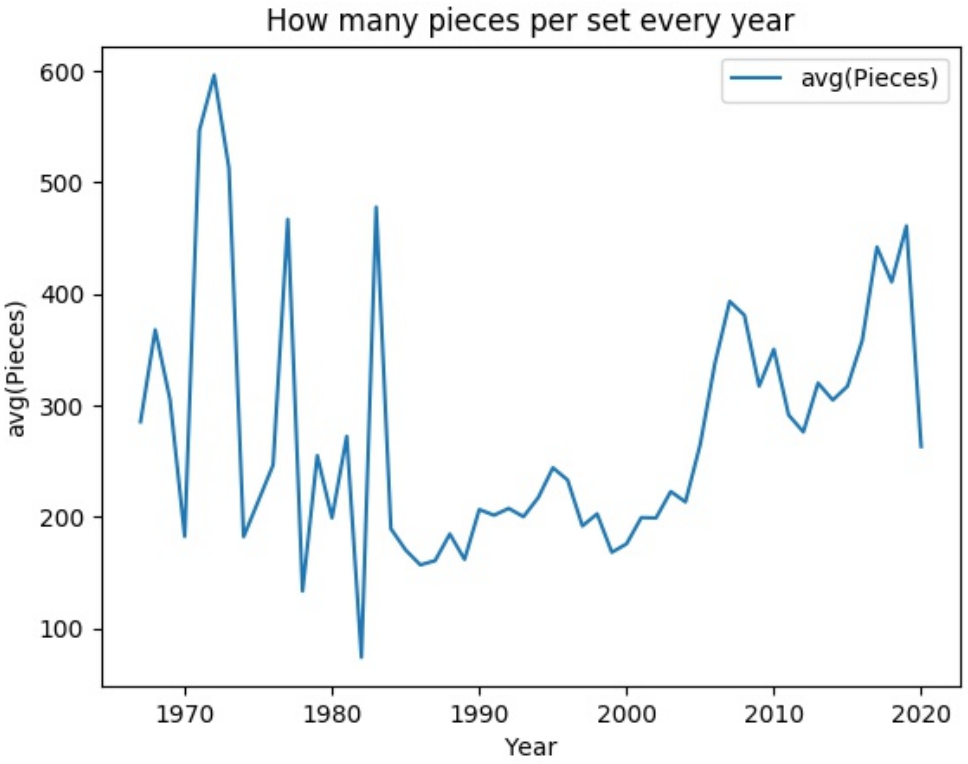


From our experience and intuition, we have the sense that LEGO is more expensive with time, but in this plot, we can see whether the normal price or the real price is

increase with time. From 1990 the price per LEGO piece is decreasing by year.

3. “how many pieces are there in average LEGO sets”

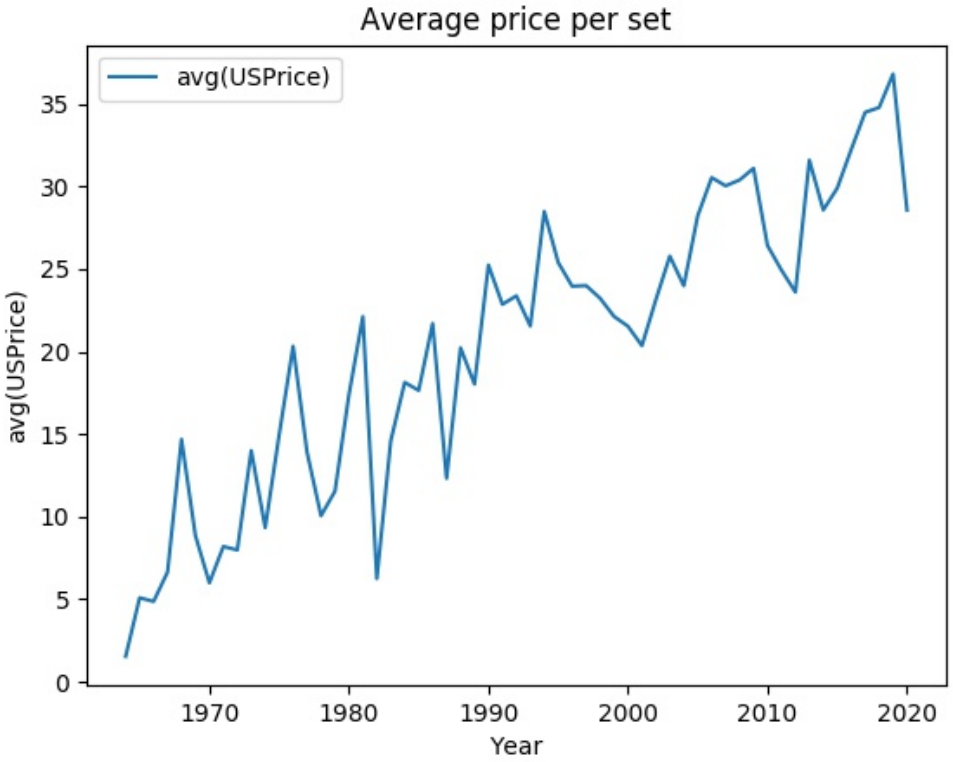
```
33  #how  many pieces per set per year每年每一套 多少块##############################
34  df_data2=dfnot0.groupby('Year').agg({'Pieces':'mean'}).orderBy('Year')#.show(25)
35  df_data2.write.csv('legodc_data2.csv')
```



As you can see on the chart above, the average size of sets released each year stayed somewhat constant from 1963-1985 until around 1990 which set sizes started to increase. The average set size seems to have peaked in 2008 (which saw the release of the Taj Mahal), but since then it has not fallen to its pre-2000 levels. It seems to have found a new normal around 300 pieces. The traditional boxes of bricks are pushed out of the way for the more profitable lines.

This increase in average piece count could be a factor in why LEGO is perceived to be more expensive now than in the past. LEGO sets have become larger and more complex. They have started to market directly to an older crowd with sets such as the Modular Buildings and the Architecture series. These new sets have rekindled interest in LEGO for an older generation but at the same time, it has introduced this same generation to the relatively high price of LEGO sets.

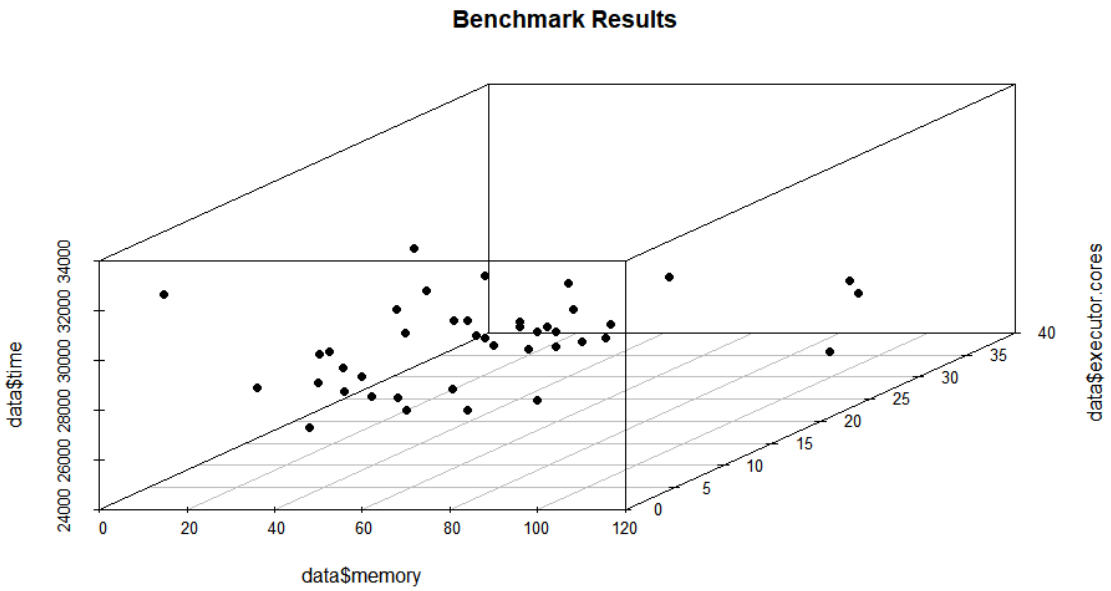
4.“Average price price per set”

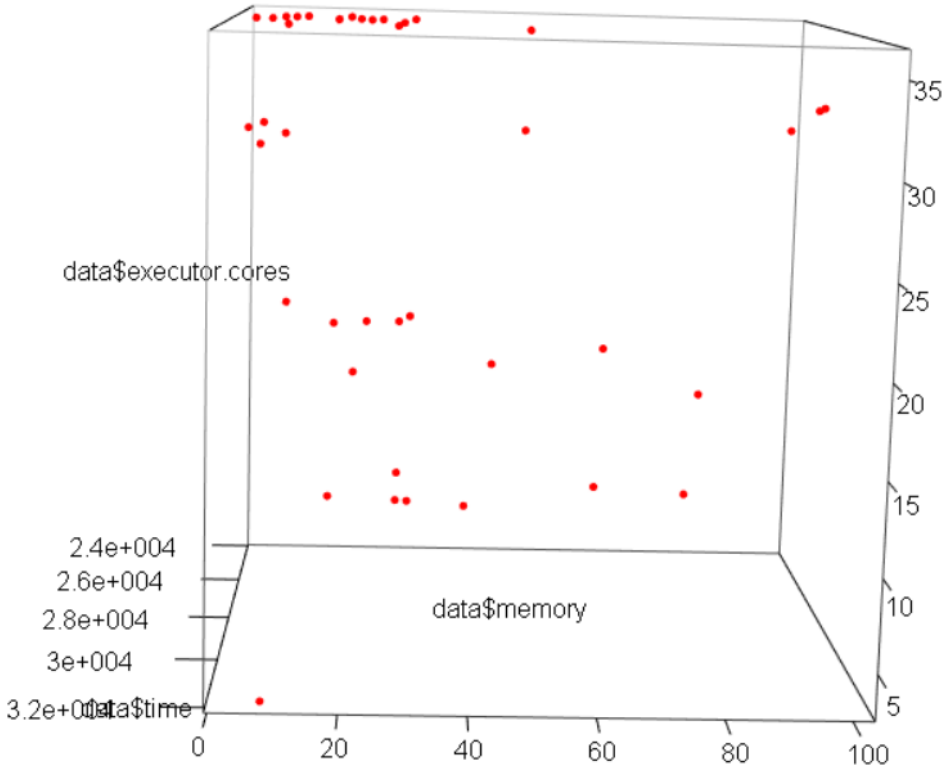


As we can see from the chart above, the average price of a set of LEGO has been increasing over time.

5.Benchmark results

I allocated different combinations of executor memory and executor cores and measured the time of them. I found that with more cores can save time, but with more executor memory can't save time. When I use executor memory more than 102G, it returns "Required executor memory overhead”.



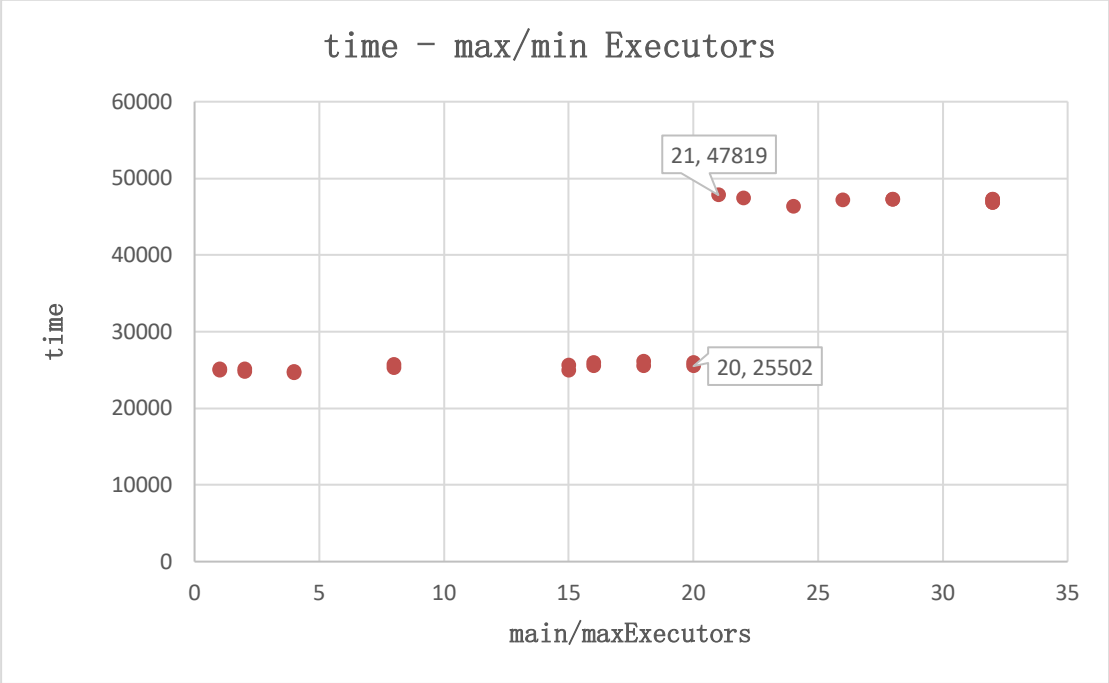


We can see the optimal combinations are when 36 cores, and memory under 40GB.

Processing time is between 24000-25000 millisecond.

6.minExecutors and maxExecutors configuration

Then I add the minExecutors and maxExecutors configuration options with fixed RAM and core combination (64GB RAM, 18 cores). I found that when the min/maxExecutors no more than 21, the processing time are always around 25000 When the min/maxExecutors reached 21, the processing time jumped 2 times as before.



I assume that , when use more than 20 executors ,the hardware resources are not enough, so spark have to use hyper threading to carry out the task, so it takes more time.

4What I learn and where I can improve with my analysis

In this project I learned basically processing data with PySpark. At first, I didn't work on ara-cluster, but on computer in Linux pool, so I faced problems, that brought by lower Spark version. Then I have following precious experience.

1. data frame format is really more convenient than RDD. RDD does not infer the schema of the ingested data, with dataframe I don't need to think how to filter the data with lambda function, to separate the data and put it in columns. Furthermore "orderby" "groupby" is really useful and convenient.

2. I learn to solve problem sentence by sentence with Pyspark shell not “spark-submit “. I faced a problem and I don’t know why, I turned to tutor and learned to run the script with Pyspark shell. Then I learned to debug my script sentence by sentence.
3. I always use “show ()” to see if the processed dataframe is what I want. In this project I learned to comment out the “show ()” after using it, because if I don’t, the following sentence will be calculate on what have showed not the whole dataframe.
4. I should list what are the 3 LEGO sets with most pieces every year, that may let me know the trends then. It would be interesting. Or I should order by price to see what are the most expensive LEGO sets each year. It would also meaningful.