BLACKROCK®

TopNotch

Systematically Quality Controlling Big Data

David Durst

Email: david.durst (at) blackrock (dot) com

TopNotch GitHub Repository: https://github.com/blackrock/TopNotch


https://pixabay.com/en/lightning-thunder-lightning-storm-1056419/

BlackRock at a Glance

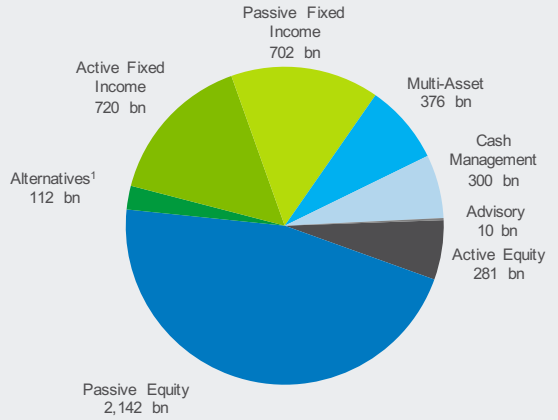
BlackRock Mission Statement

Create a better financial future for our clients by building the most respected investment and risk manager in the world

BlackRock facts*

- Established in 1988
- NYSE: BLK
- $4.64 trillion assets under management
- More than 12,000 employees
- More than 1,800 investment professionals
- Offices in over 30 countries
- 25 primary investment centers globally
- Clients in over 100 countries
- Over 750 iShares® ETFs
- Through BlackRock Solutions, the Firm provides risk management and enterprise investment services for over 200 clients
- Financial Markets Advisory business managed or advised on over $8 trillion in asset and derivative portfolios
- Transition Management team partners with clients to save costs and reduce risks when changing investment exposures

$4.64 trillion managed across asset classes



Assets as of 12/31/2015
[1]Includes commodity and currency mandates

*As of 12/31/2015

What I Do: Financial Modeling Group

- Following 2008 Crisis, more disclosures on assets backed by mortgages, credit card debt, etc.
- Modeling technology must adapt to larger data sets
- Models use a wide variety of large (1-10 TB) and noisy data sets
- Have significant resources: 3000 Executors, 12 TB of RAM

The Problem

How to quality control data sets for all models?

- **How to define and measure data quality?**
- **How to efficiently ensure quality across all data sets?**
- **How to institutionalize knowledge of data sets?**
- **Existing tools not appropriate for this problem, focus on:**
  - Data sets that fit in single computer's memory
  - Answering queries approximately correctly

Example

What We Have:

- **Missing and corrupt data**

What We Want:

- **Definitions and metrics for data quality**

Example U.S. Mortgage Data

id	balance	FICO	LTV	ZIP	Owner Occupied
1	#!@a	750	87	75987	TRUE
2	250,000	580	75	H2AOB8	FALSE
3	1,000,000		400	91538	TRUE

Example

What We Have:

- Missing and corrupt data
- Valid data that doesn't belong in this data set

What We Want:

- Definitions and metrics for data quality
- Efficient quality control for many data sets

Example U.S. Mortgage Data

id	balance	FICO	LTV	ZIP	Owner Occupied
1	#!@a	750	87	75987	TRUE
2	250,000	580	75	H2AOB8	FALSE
3	1,000,000		400	91538	TRUE

Example

What We Have:

- Missing and corrupt data
- Valid data that doesn't belong in this data set
- Data violating domain expertise

What We Want:

- Definitions and metrics for data quality
- Efficient quality control for many data sets
- Institutionalized knowledge

Example U.S. Mortgage Data

id	balance	FICO	LTV	ZIP	Owner Occupied
1	#!@a	750	87	75987	TRUE
2	250,000	580	75	H2AOB8	FALSE
3	1,000,000		400	91538	TRUE


https://pixabay.com/en/lightning-thunder-lightning-storm-1056419/


https://pixabay.com/en/sky-cloud-sunshine-summer-solar-383823/

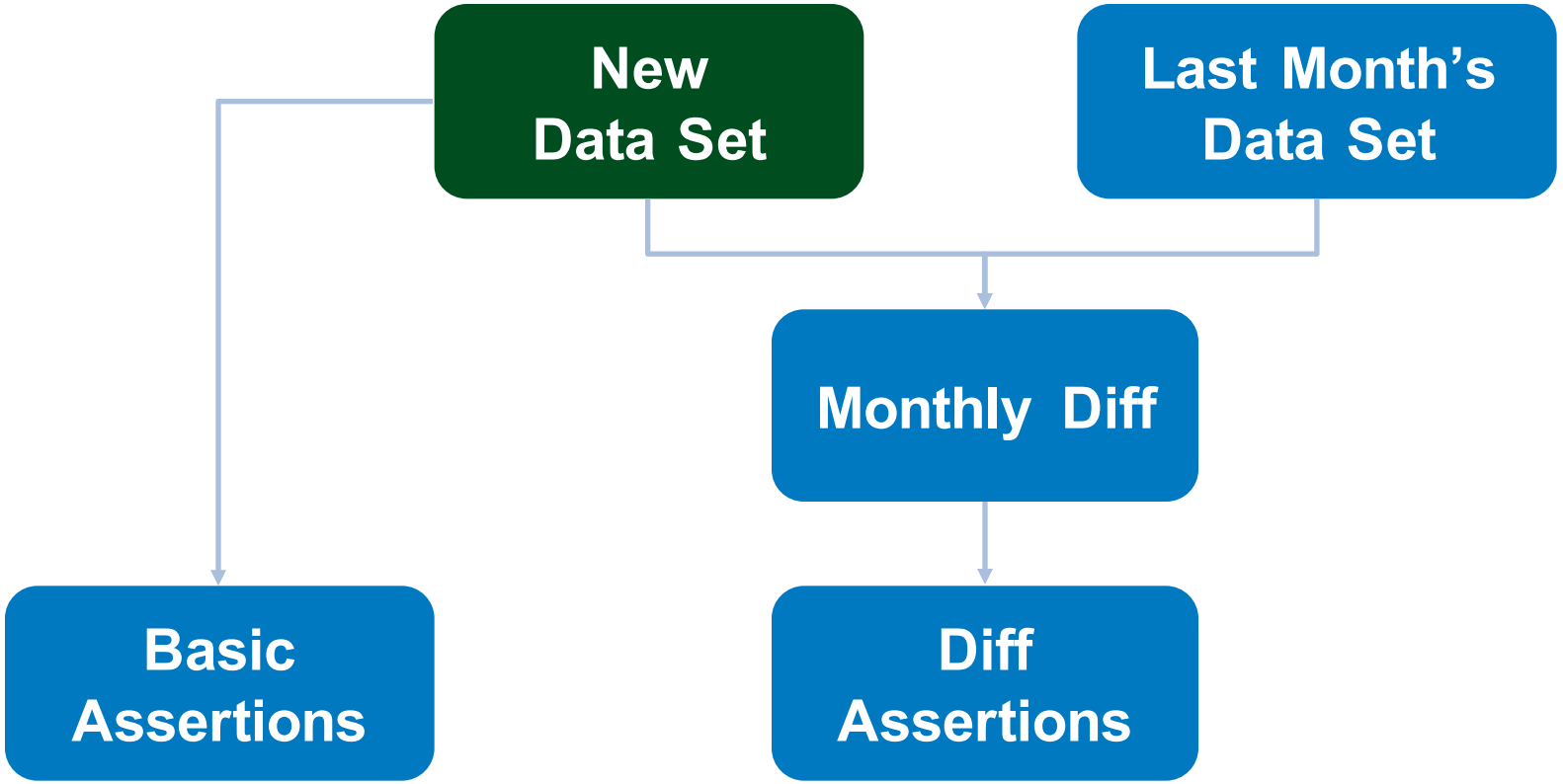
The Solution: TopNotch

- A unit testing framework for data
- Assertion – verify a fact about a data set
- Diff – compare two data sets with similar schemas
- View – transform data for diff and assertion
- Plan – an ordered series of assertions, diffs, and views, known as commands, which handles all file io

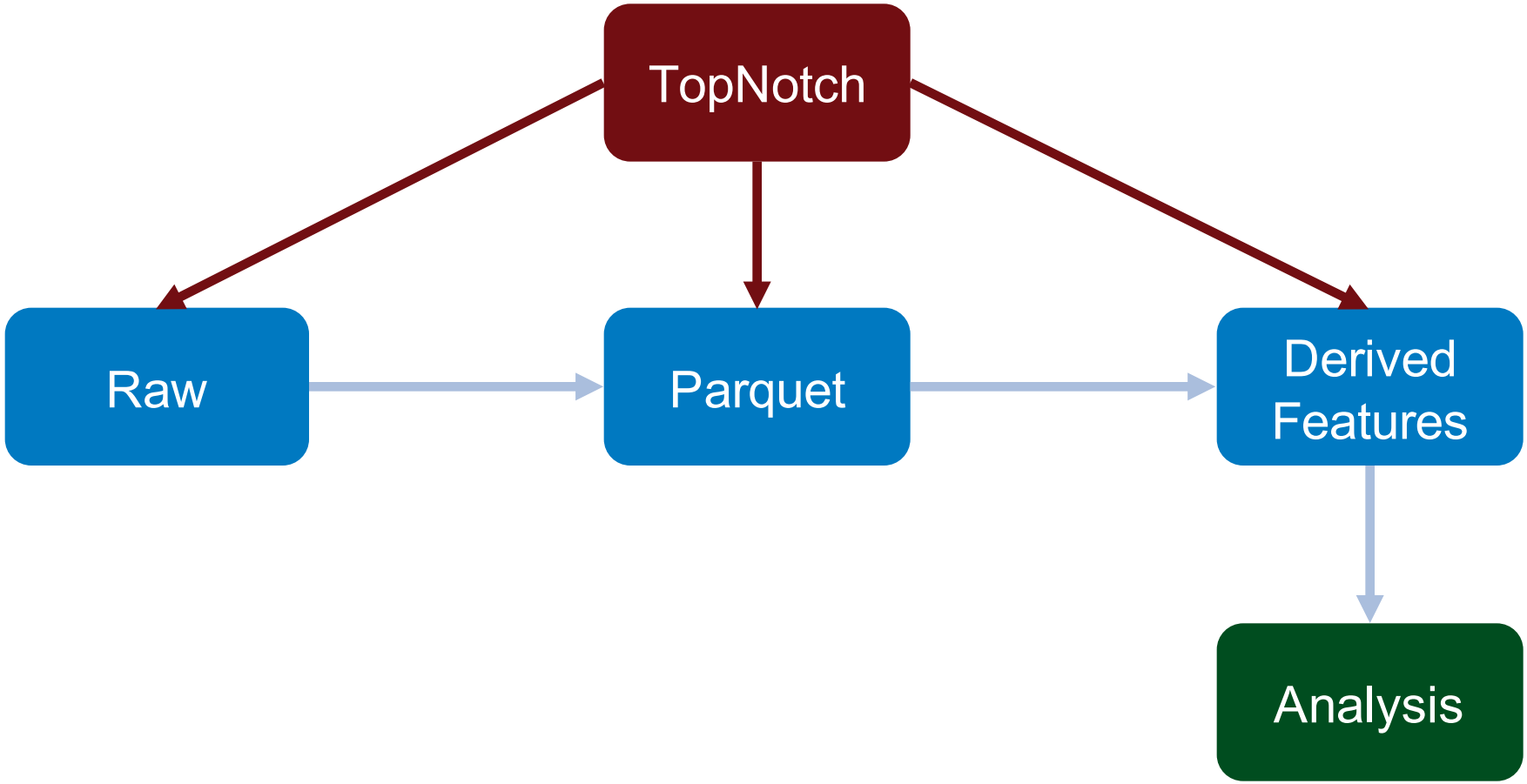
How TopNotch Solves The Three Problems

1. Assertions rigorously define and measure data quality
2. Reuse of commands across data sets maximizes efficiency
3. Plans and commands institutionalize knowledge of data sets

Example Monthly Data Set



TopNotch For One Model



TopNotch For All Models


https://pixabay.com/en/screws-
hardware-ironware-metal-600491/

https://pixabay.com/en/metal-type-lead-vintage-mechanical-110989/

Proposed Standard For Organizing Commands

- Each command is stored as a JSON file
- Keep organized, central library of commands in GitHub
- Each community maintains a central repository and learns to quality control data together

How Can I Get Involved?

- TopNotch Backend: https://github.com/blackrock/TopNotch
- TopNotch Frontend: Coming Soon
- Talk With Me After Presentation!
- Email Me: david.durst (at) blackrock (dot) com

Appendix

Basic Assertion

```
{
 "topnotch" : {
  "assertions" : [ {
     "query": "CurBal < OrigBal",
     "description": "balance has gone down over time",
     "threshold": 0.20
   } ]
 }
}
```

Back

Description	Query	Fraction Invalid	Threshold	Invalid Sample
balance has gone down over time	CurBal < OrigBal	0.3962641761174116	0.2	curbal / origbal 1018560 / 888510 10230346 / 4815197 11543251 / 9404500 11700947 / 1831884 14090452 / 7987762

Back

Monthly Diff

```
{
  "topnotch": {
    "input1Columns": {
      "joinColumns": [ "LoanID" ],
      "diffColumns": [ "CurBal" ]
    },
    "input2Columns": {
      "joinColumns": [ "LoanID" ],
      "diffColumns": [ "CurBal" ]
    }
  }
}
```

Back

Diff Demo

New_LoanID	Old_LoanID	New_CurBal	Old_CurBal
31	31	281937	324066

New_CurBal__minus__Old_CurBal	New_CurBal__equals__Old_CurBal
-42129	both not null, same type, not equal

Back

Diff Assertions

```
{
  "topnotch" : {
    "assertions" : [ {
      "query": "New_CurBal < Old_CurBal",
      "description": "Loans' balances are decreasing each month",
      "threshold": 0.20
    } ]
  }
}
```

Back

Description	Query	Fraction Invalid	Threshold	Invalid Sample
Loans' balances are decreasing each month	New_CurBal < Old_CurBal	0.09673115410273515	0.2	old_curbal: 147065, 171363, 171719, 1772516, 181288 new_curbal: 185301, 293030, 240406, 2782850, 362576

Back

SQL And The Where Clause

Pros

- **Solves the three major problems of data quality control**
- **Maximizes number of possible TopNotch users**
- **Enables catching most important errors**

Con

- **Limits ability to express more complicated queries**
  - Time relationships
  - Complicated joins

- **Potential Solutions**
  - Self joins in views
  - Hive UDFs
  - Integration with Thunder or Spark-TS

Making Rules Schema Independent

- Dependence on column names can limit reusability
- Solution 1: use Spark's HiveQL Parser to identify column names, document them, and allow for easy modification to fit schema
- Solution 2: use pattern matching for column names
- Better Solution: a proper combination of 1 and 2
- Under Consideration

Time Series Data

- See discussion of SQL and where clauses

Disclosures