Algorithmic Digital Attribution Using Spark

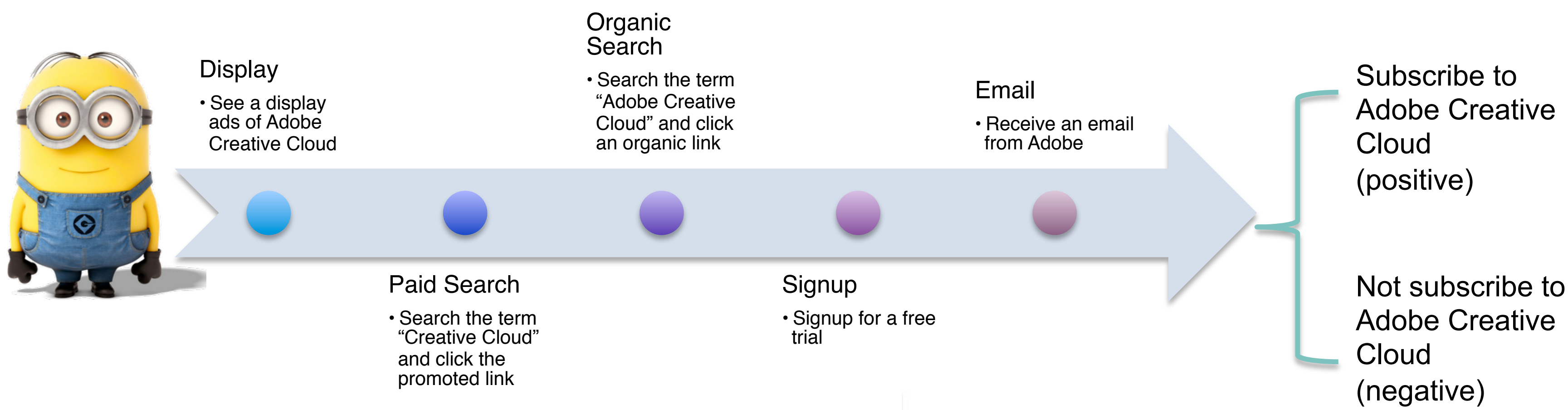
Anny (Yunzhu) Chen and William (Zhenyu) Yan

Adobe

Agenda

- Digital attribution
- Algorithmic digital attribution
- Spark implementation
- Lessons learned

Path of media touches to conversion



A customer may receive various kinds of media touch points before deciding whether to subscribe to a product (conversion) or not

What's digital attribution?

- A digital attribution model determines how credits for conversions are assigned to media touch points
- It is quite important in performance monitoring and budget planning

Display
- Saw a display ads of Adobe Creative Cloud
- 0.1

Paid Search
- Search the term "Creative Cloud" and click the promoted link
- 0.2

Organic Search
- Search the term "Adobe Creative Cloud" and click an organic link
- 0.2

Signup
- Signup for a free trial
- 0.2

Email
- Received an email from Adobe
- 0.3

Convert!

Digital attribution model at a glance

Models	Consider all media touch points	Time decay	Data driven (vs. rule based)
Last interaction, first interaction, last paid search click	no	no	no

Algorithmic Attribution

- Rule based: predetermined weights based on rules
- Algorithmic: machine learning or statistical models are used to determine the weights

Algorithmic Attribution Modeling

The attribution model is based on a combination of

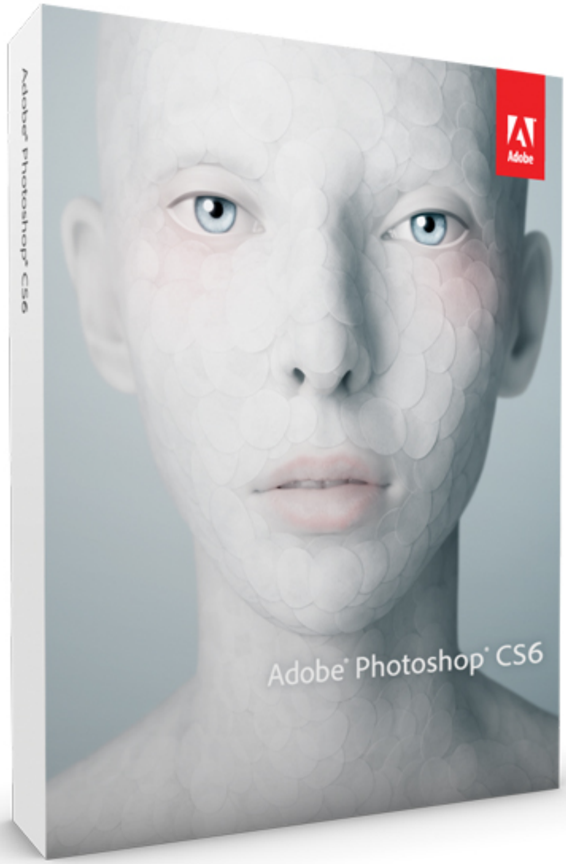
- Distributed-lag econometric model
- Discrete-time survival model

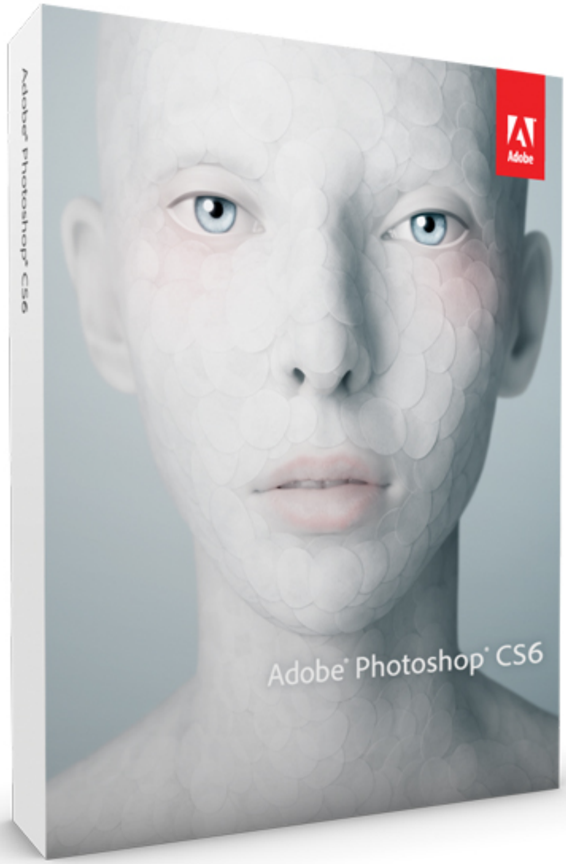
Some highlights:

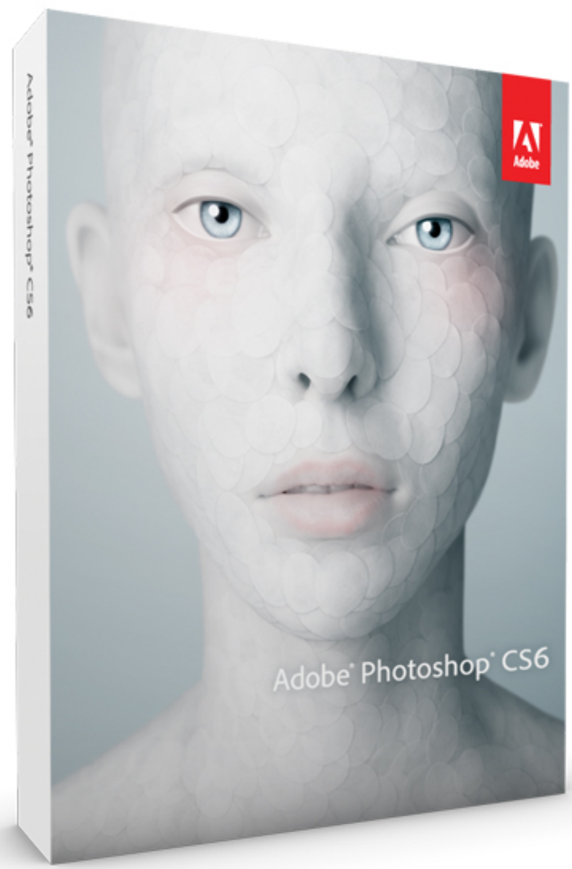
- The basic idea is to compare the media touches in positive paths vs. those in negative paths and thus infer their effects
- Logistic link function
- Time decay parameters to address time-decaying of media effects
- Fit media touch effects and decay parameters simultaneously
- Constraints on coefficients (combining with rules)
- Stratified sampling
- Bias reduction:
  - Using control variables, such as duration of exposure
  - Causal modeling
- Maximum likelihood estimation

Tokenization

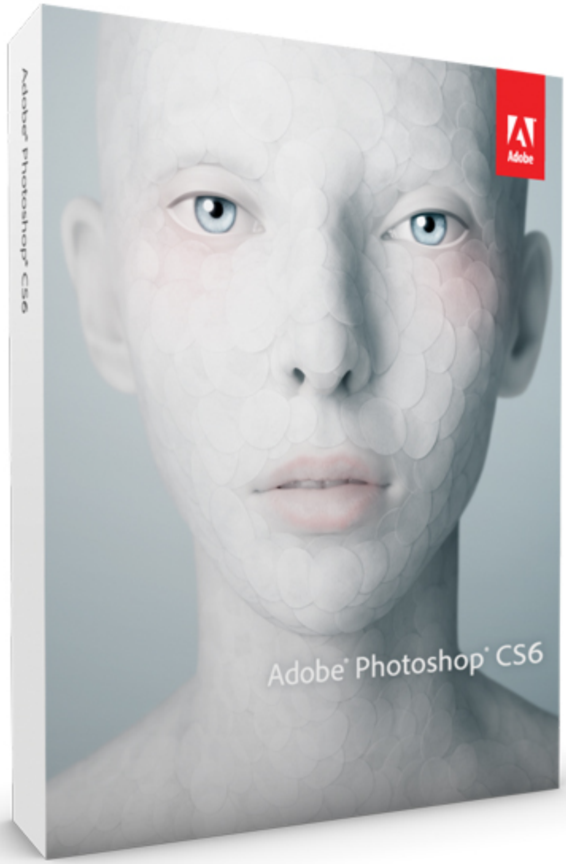
- Tokenization is used to group neighboring events together
- [why is tokenization needed?]()


Adobe
Adobe Photoshop CS6
Adobe Photoshop CS6


Adobe
Adobe Photoshop CS6
Adobe Photoshop CS6


Adobe
Adobe Photoshop CS6
Adobe® Photoshop® CS6

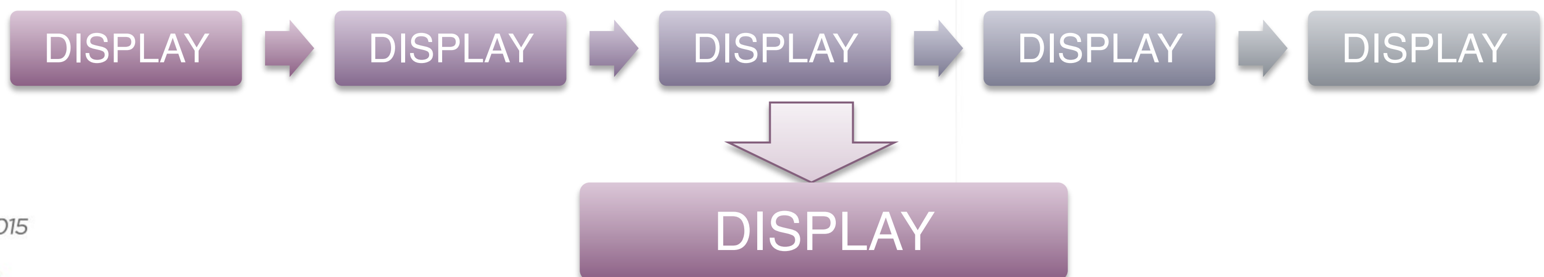

Adobe
Adobe Photoshop CS6
Adobe Photoshop CS6


Adobe
Adobe® Photoshop® CS6

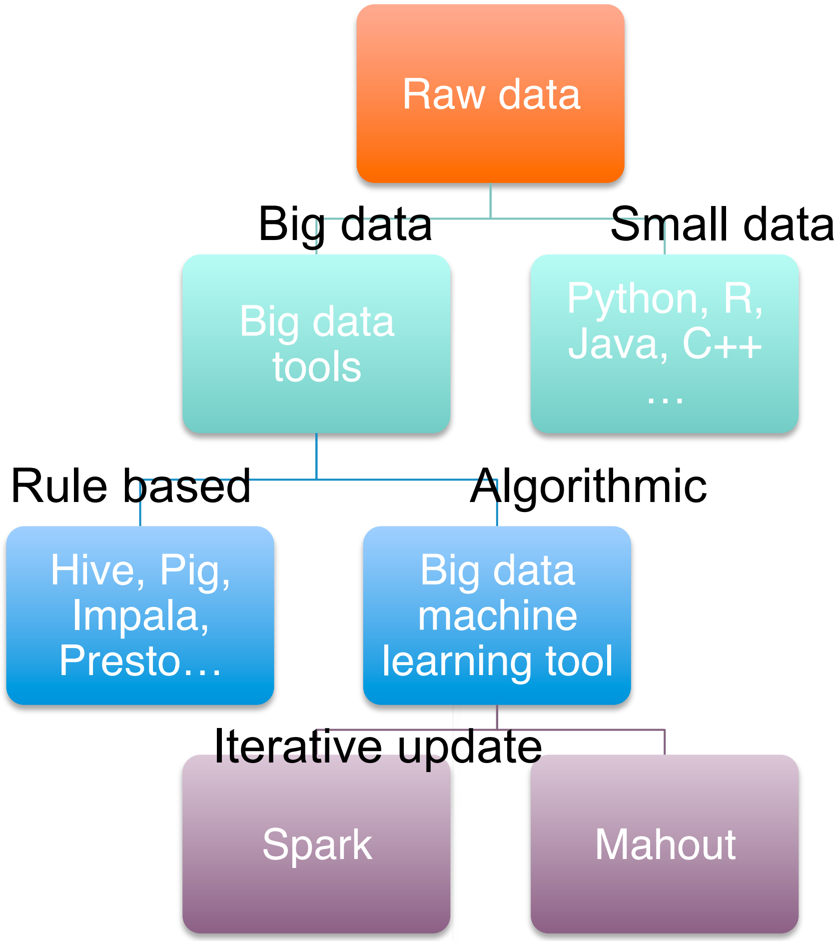
Tokenization

- The effect of a media touch is hardly doubled or tripled if it is sent repeatedly to a customer twice or three times in a short period
- Tokenization is used to group neighboring events together

- If you see the same advertisement 5 times in one hour, will you be impressed 5 times or just have the impression “I saw that ad”?



Why Spark?


Raw data
Big data
Small data
Big data tools
Python, R, Java, C++ …
Rule based
Algorithmic
Hive, Pig, Impala, Presto…
Big data machine learning tool
Iterative update
Spark
Mahout

Data process and stitching

Event

- Media touch events; conversion events; other user behavior events
- Data fields: customer id , time stamp, event type, campaign id and etc.

Path

- Stitch the events of the same customer by “id”
- Generate both positive paths and negative paths

Model

- Each path is a record
- Positive/negative label is the outcome

Algorithm evolution

First model

- Fix time decay
- Logistic regression using Mllib logistic regression with SGD

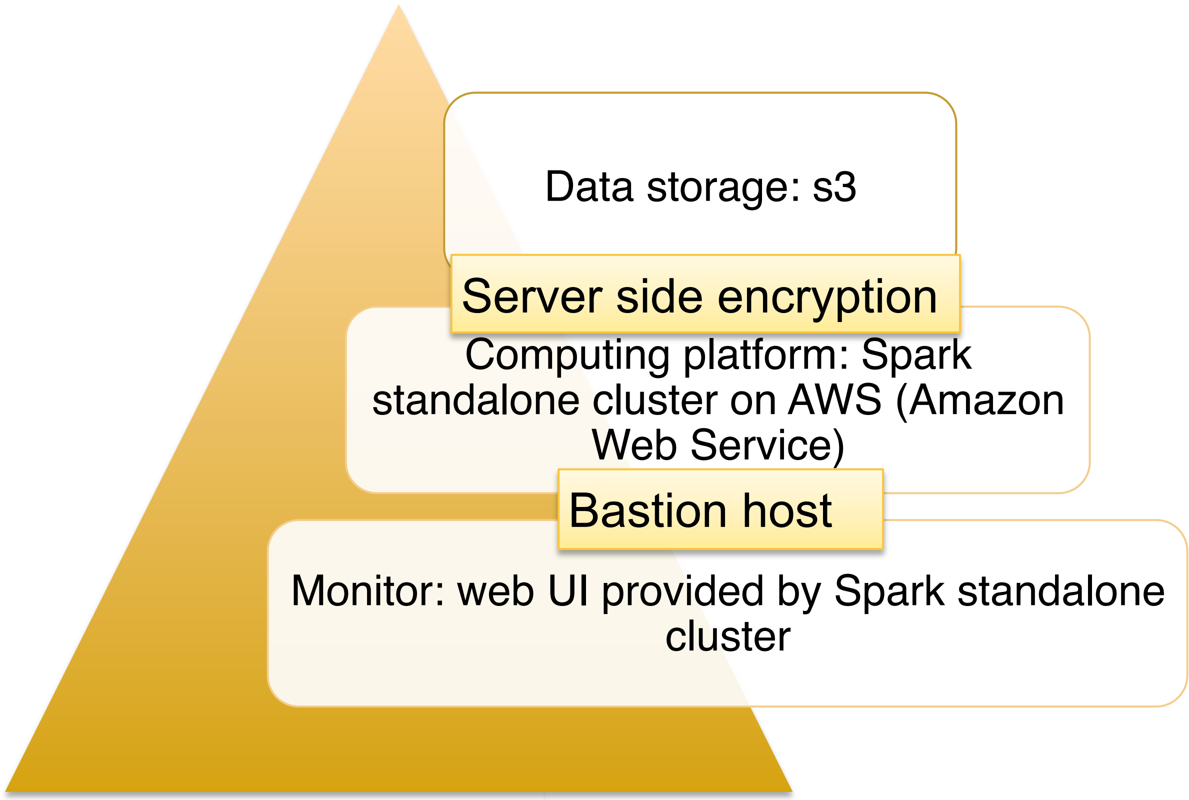
Second model

- Include both regression coefficients and decay parameters in the model
- Optimize alternatively in each iteration
- Customize and extend the original Mllib logistic regression

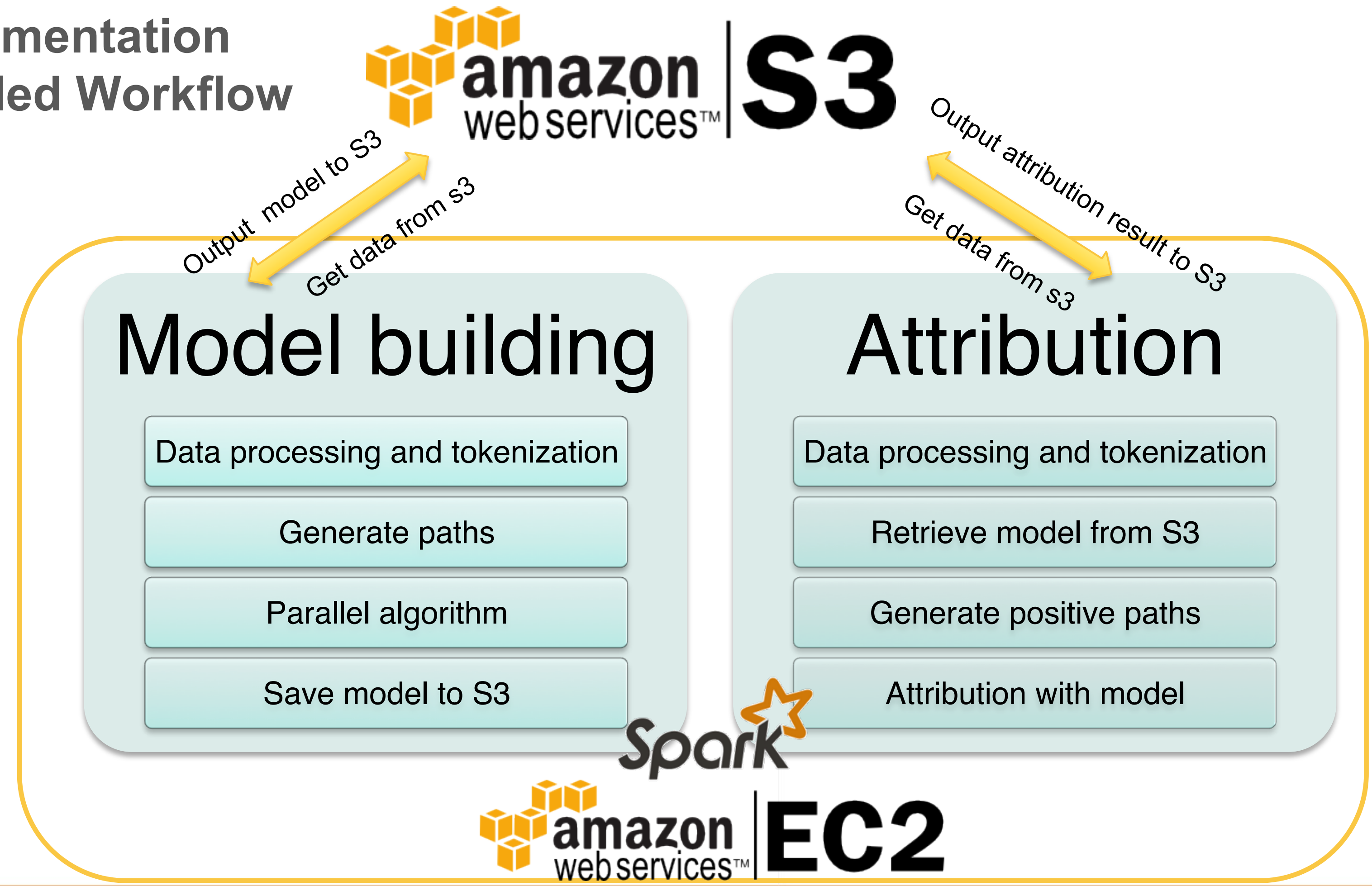
Third model

- Second model + Causal modeling (matched sample)

Implementation architecture



Implementation Detailed Workflow



Lessons learned

- Memory management
  - Each record was transformed from String-based to Byte-based using a hash function
    - Tremendously increased the speed
    - Reduced shuffle size in the next groupBy step

Completed Stages (2)

Stage Id	Pool Name	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Shuffle Read	Shuffle Write
2	default	map at StrByteMapping.scala:129 +details	2015/06/10 05:32:18	16 min	206/206	225.0 GB		58.1 GB
0	default	toArray at AttributeResults.scala:82 +details	2015/06/10 05:32:13	5 s	2/2	1735.0 B		

Lessons learned

- Memory management
  - Write separate classes for model training and attribution computation
    - Model training needs complex transformation and intensive iteration
    - Attribution computation needs more information from each objects

Lessons Learned (cont'd)

- Cache before iterative computation
  - Only cache the RDDs right before entering the model
  - Unpersist unused RDDs to save space
- Clear jobs of stopped apps in workers from time to time
  - Check and clean ~/spark/work folder in workers
  - Check and clean /mnt/spark folder in workers if necessary
- Be careful when dealing with unserialized Java objects

Lessons Learned (cont'd)

- Errors of one line of code doesn't necessarily come from that line
  - Spark is lazy evaluation
  - Errors arise at action step, but may come from the previous transformation steps
- Adjustment of step size
  - Time decay must be between 0 and 1
  - After the matching, sample size decreased dramatically, the step size has to be adjusted accordingly

Thank you!