ADATAO

DATA INTELLIGENCE FOR ALL

Distributed DataFrame (DDF)

Simplifying Big Data
For The Rest Of Us

Christopher Nguyen, PhD
Co-Founder & CEO

Agenda

1. Motivation: Problem & Solution
2. DDF Features & Benefits
3. Demo

Christopher Nguyen, PhD
Adatao Inc.
Co-Founder & CEO

- Former **Engineering Director of Google Apps** (Google Founders' Award)
- Former Professor and Co-Founder of the **Computer Engineering program at HKUST**
- **PhD Stanford, BS U.C. Berkeley** *Summa cum Laude*


Small Data
World


BIG
BAD
DATA
Pig
HDFS
HBASE
Hive
Cascading
MapReduce
SummingBird
Crunch
Storm
Scalding
RHadoop


BIG
BAD
DATA
SHARK
Spark
Spark

```
class RandomSplitRDD( prev: RDD[Array[Double]],  seed: Long
upper: Double,   isTraining: Boolean)
  extends RDD[Array[Double]](prev) {
  override def getPartitions: Array[Partition] = {
    val rg = new Random(seed)
    firstParent[Array[Double]].partitions.map(x => new
rg.nextInt))
  }
  override def getPreferredLocations(split: Partition):

firstParent[Array[Double]].preferredLocations(split.asI
n].prev)
  override def compute(splitIn: Partition, context: TaskC
Iterator[Array[Double]] = {
    val split = splitIn.asInstanceOf[SeededPartition]
    val rand = new Random(split.seed)
    if (isTraining) {
      firstParent[Array[Double]].iterator(split.prev, context).fi
        val z = rand.nextDouble;
        z < lower || z >= upper
      })
    } else {
      firstParent[Array[Double]].iterator(split.prev, context).filter(x => {
        val z = rand.nextDouble;
        lower <= z && z < upper
      })
    }
  }
}
def randomSplit[Array[Double]](rdd: RDD[Array[Double]], numSplits: Int,
trainingSize: Double, seed: Long): Iterator[(RDD[Array[Double]],
RDD[Array[Double]])] = {
    require(0 < trainingSize && trainingSize < 1)
    val rg = new Random(seed)
    (1 to numSplits).map(_ => rg.nextInt).map(z =>
      (new RandomSplitRDD(rdd, z, 0, 1.0 - trainingSize, true),
        new RandomSplitRDD(rdd, z, 0, 1.0 - trainingSize, false))).toIterator
  }
```

WHY

Can’t It Be Even Simpler?

```
class RandomSplitRDD( prev: RDD[Array[Double]],  seed: Long
upper: Double,   isTraining: Boolean)
  extends RDD[Array[Double]](prev) {
  override def getPartitions: Array[Partition] = {
    val rg = new Random(seed)
    firstParent[Array[Double]].partitions.map(x => new
rg.nextInt))
  }
  override def getPreferredLocations(split: Partition):

firstParent[Array[Double]].preferredLocations(split.asI
n].prev)
  override def compute(splitIn: Partition, context: TaskC
Iterator[Array[Double]] = {
    val split = splitIn.asInstanceOf[SeededPartition]
    val rand = new Random(split.seed)
    if (isTraining) {
      firstParent[Array[Double]].iterator(split.prev, context).fi
        val z = rand.nextDouble;
        z < lower || z >= upper
      })
    } else {
      firstParent[Array[Double]].iterator(split.prev, context).filter(x => {
        val z = rand.nextDouble;
        lower <= z && z < upper
      })
    }
  }
}
def randomSplit[Array[Double]](rdd: RDD[Array[Double]], numSplits: Int,
trainingSize: Double, seed: Long): Iterator[(RDD[Array[Double]],
RDD[Array[Double]])] = {
    require(0 < trainingSize && trainingSize < 1)
    val rg = new Random(seed)
    (1 to numSplits).map(_ => rg.nextInt).map(z =>
      (new RandomSplitRDD(rdd, z, 0, 1.0 - trainingSize, true),
        new RandomSplitRDD(rdd, z, 0, 1.0 - trainingSize, false))).toIterator
  }
```

What Happiness Looks Like

table = load(“housePrices”)

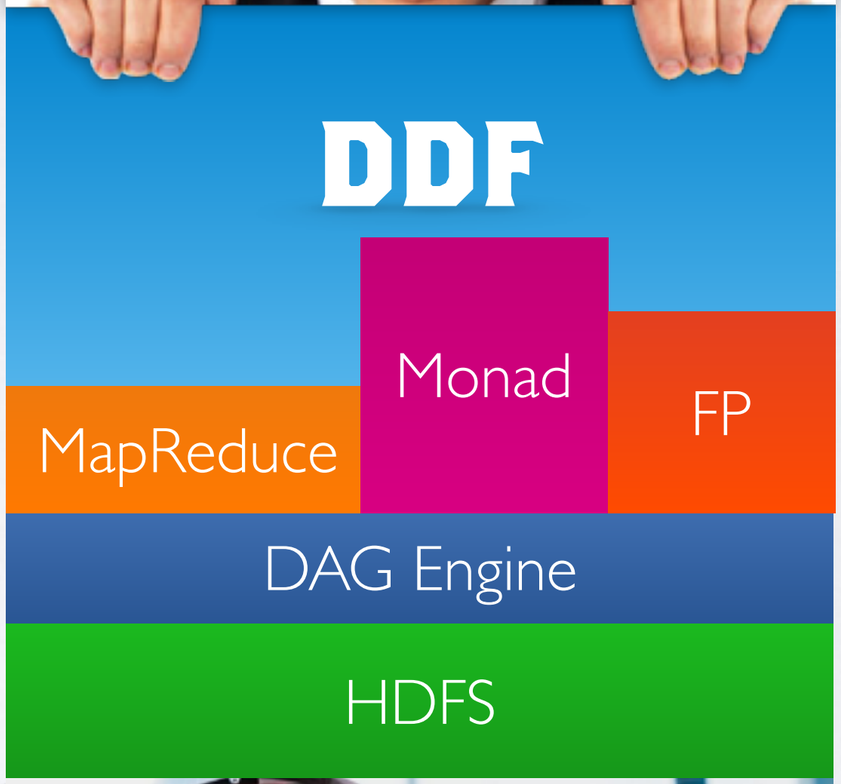
table.dropNA

table.train.glm(“price”, “sf,zip,beds,baths”)

Top-Down

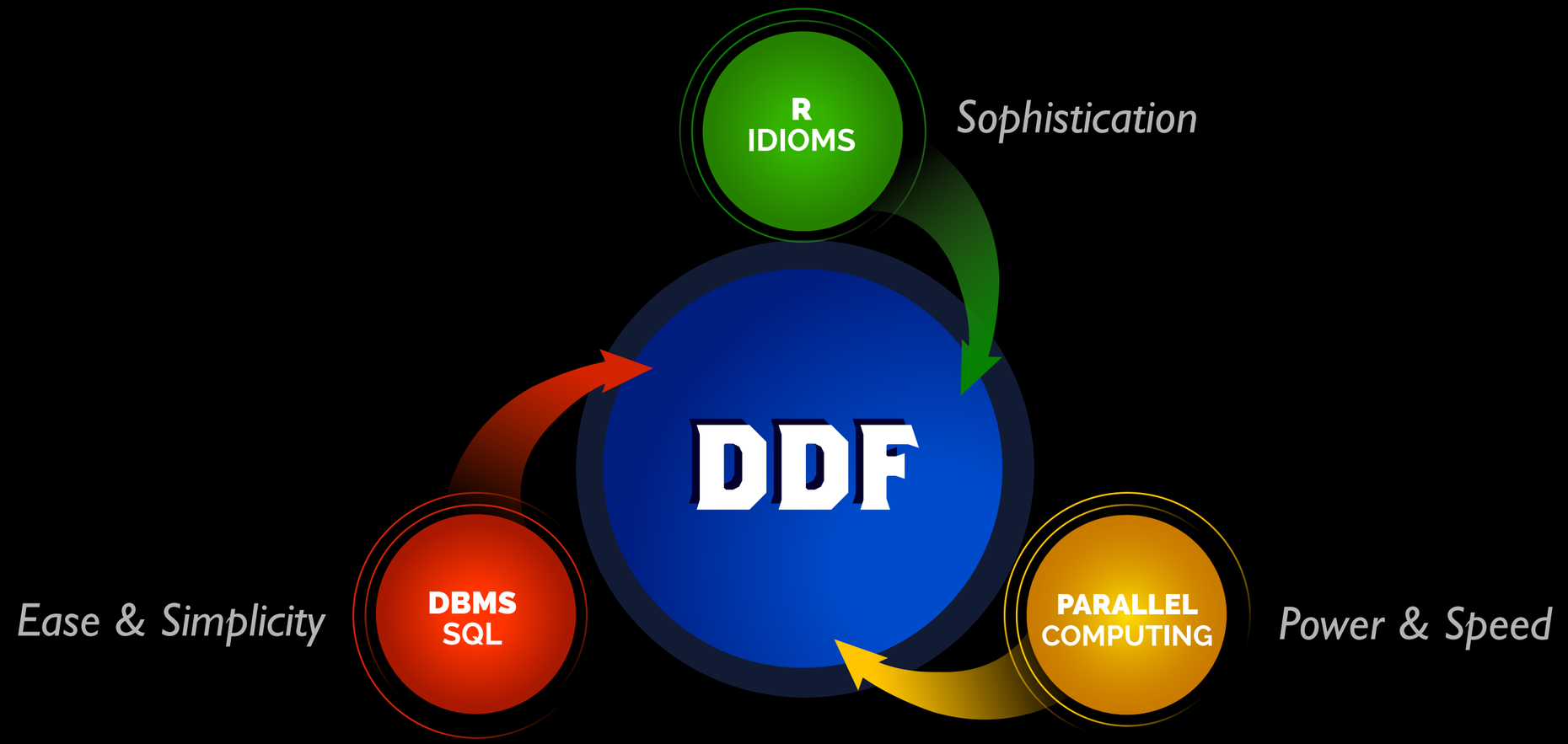
vs.

Bottom-Up

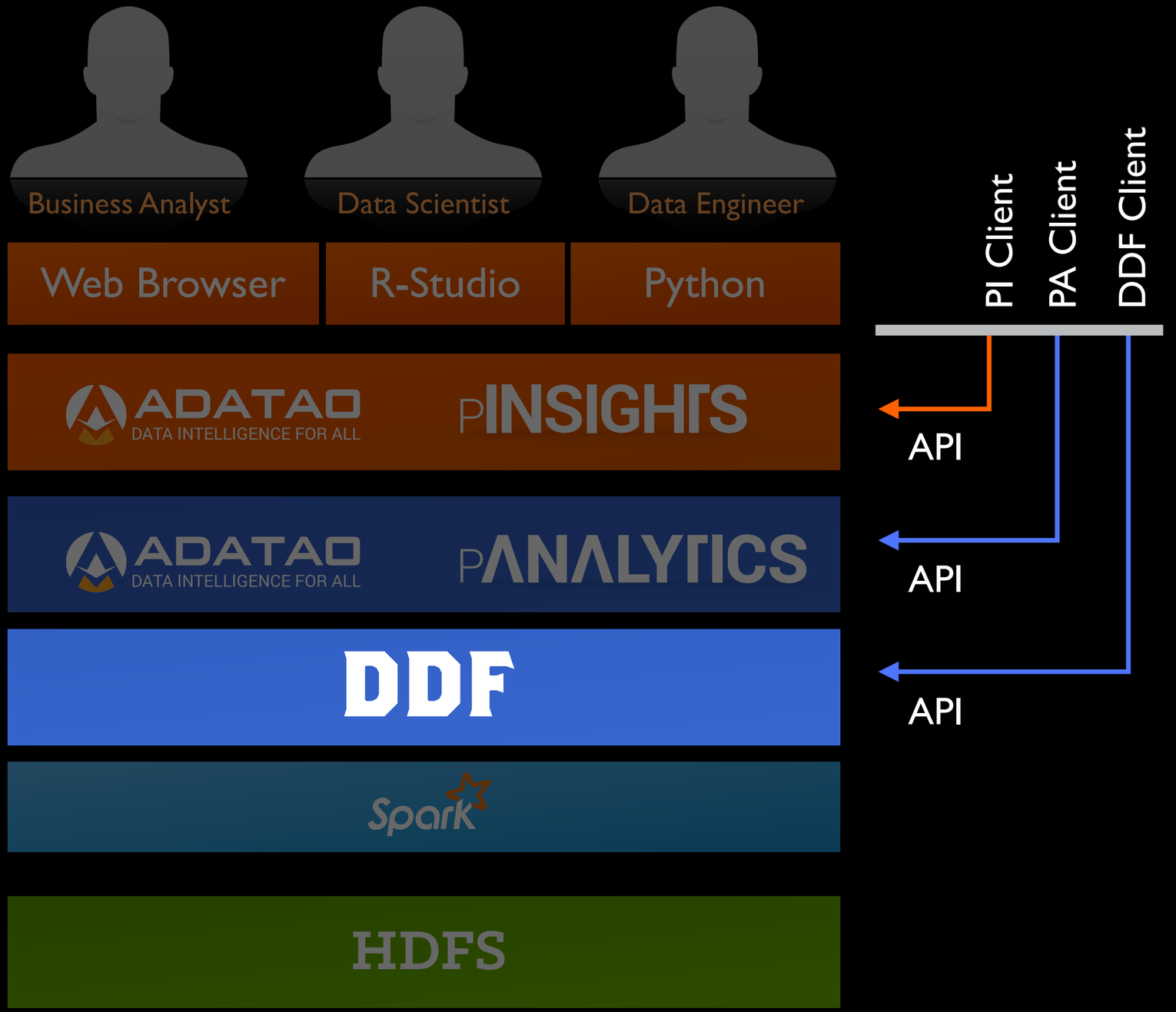



DDF

Design Principles



Example:


Business Analyst
Data Scientist
Data Engineer
Web Browser
R-Studio
Python
ADATAO
DATA INTELLIGENCE FOR ALL
pINSIGHTS
ADATAO
DATA INTELLIGENCE FOR ALL
pANALYTICS
DDF
Spark
HDFS
PI Client
PA Client
DDF Client
API
API
API

Demo Deployment

Diagram

DDF Offers

Table-Like Abstraction
on Top of Big Data

1

6

DDF Offers

```
ddf.getSummary()

ddf.getFiveNum()

ddf.binning("distance", 10)

ddf.scale(SCALE.MINMAX)
```

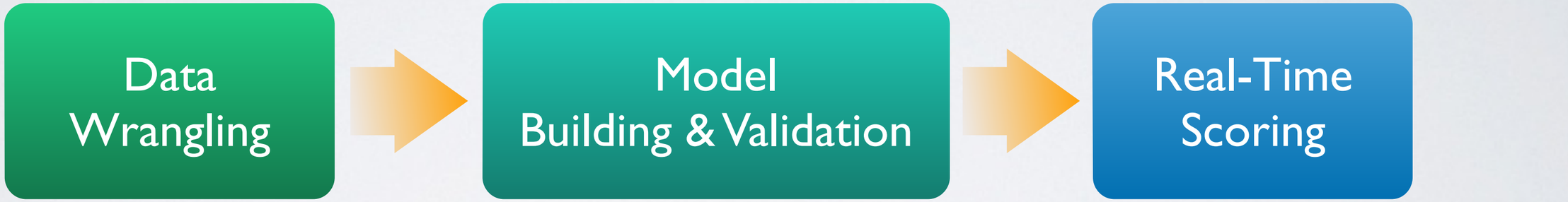
2

Native R Data.frame Experience

DDF Offers

3

Focus on Analytics, Not MapReduce




```
ddf.dropNA()
ddf.transform("speed
=distance/duration")
```

```
ddf.lm(0.1, 10)
ddf.roc(testddf)
```

```
manager.loadModel(uri)
lm.predict(point)
```

DDF Offers

4

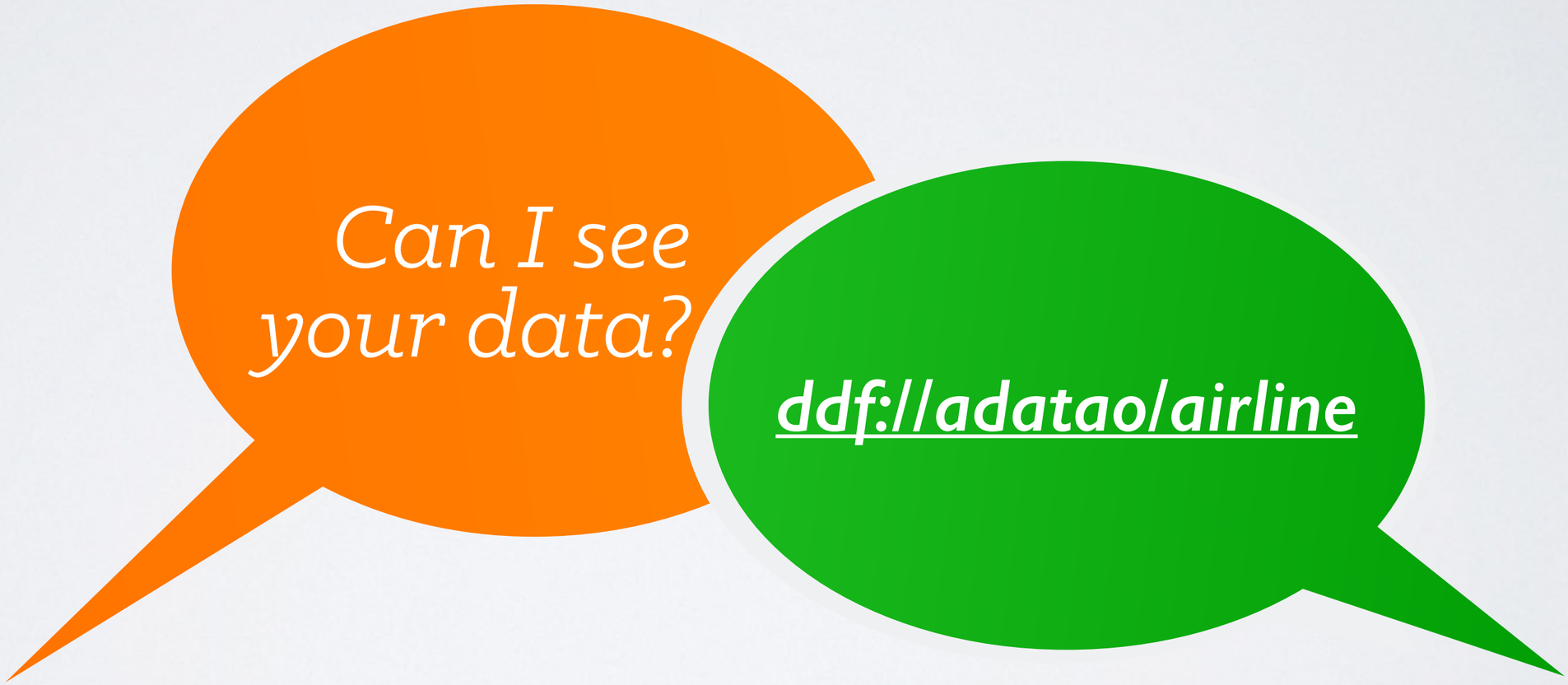
Seamlessly Integrate
with External ML Libraries

Data *Algorithm* *Intelligence*

DDF Offers

5

Share DDFs
Seamlessly & Efficiently

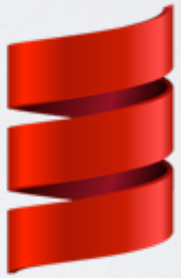


DDF Offers

Work in Your Preferred Language

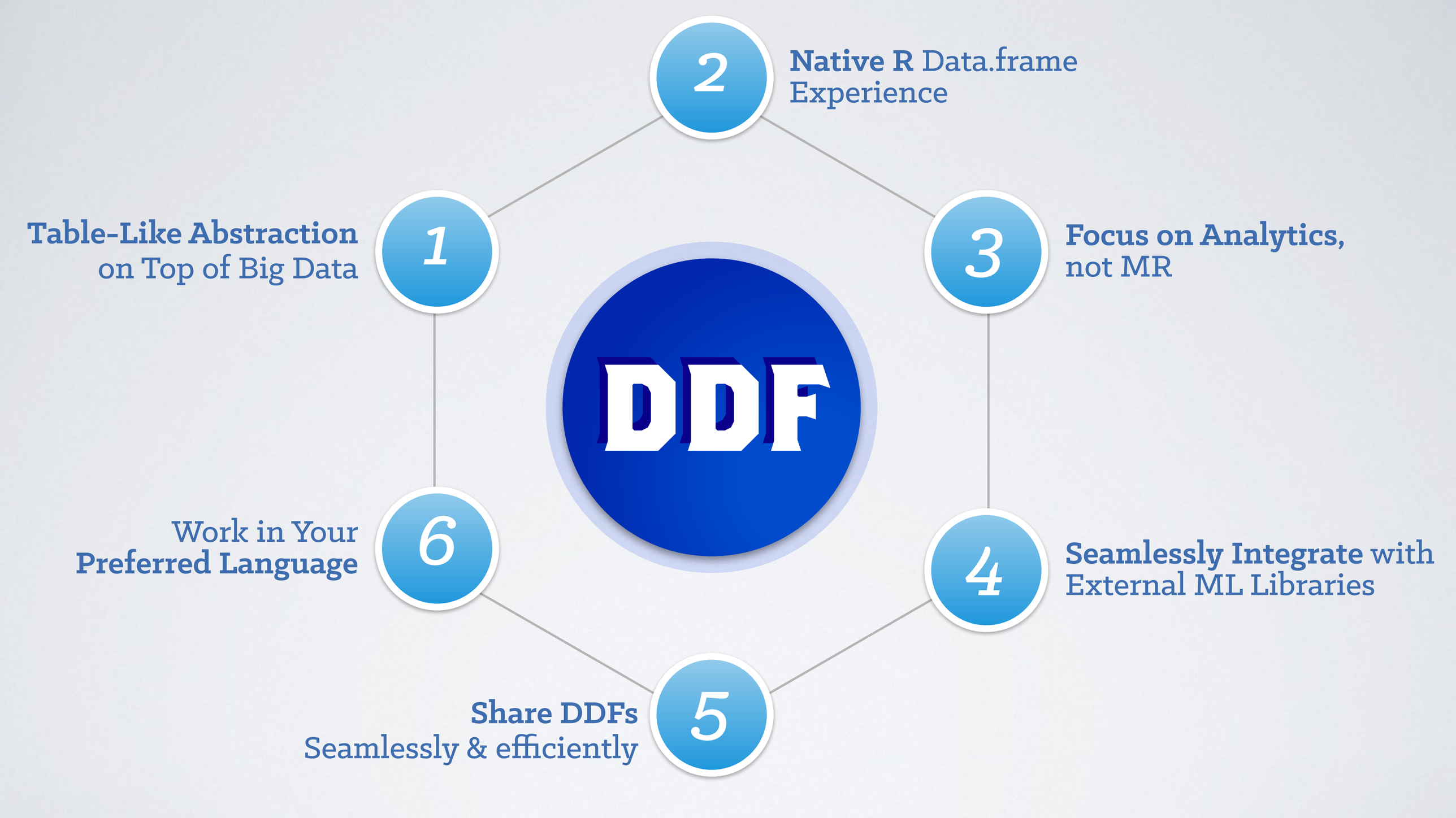
6

python



Scala

…even Plain English!


2
Native R Data.frame Experience
1
Table-Like Abstraction on Top of Big Data
3
Focus on Analytics, not MR
DDF
6
Work in Your Preferred Language
4
Seamlessly Integrate with External ML Libraries
5
Share DDFs Seamlessly & efficiently

DDF *is to* Big Data *as* SQL *is to* Small Data

ADATAO

DATA INTELLIGENCE FOR ALL

We're Open Sourcing DDF!!!

1. Accept Core Committers
2. Clean up & Prep
3. Open up for public access

www.adatao.com/ddf