Lecture 8: DBSCAN Clustering and Analytics Dataflow

COSC 526: Introduction to Data Mining Spring 2021

THE UNIVERSITY OF
TENNESSEE
KNOXVILLE

KNOXVILLE

BIG ORANGE. BIG IDEAS.®

Lecture Outline

- Assignment 8
  - Three problems with missing data
- Project Discussion
  - Define key questions and tentative title
- Dataflow and DBSCAN
  - More in the next slides
- If time left
  - Live chat and video - https://www.youtube.com/watch?v=_2u_eHHzRto

Lecture Outline

- Use work in paper *"Development of a Scalable Method for Creating Food Groups Using the NHANES Dataset and MapReduce"* to learn about:
  - Using a dataflow for tackling a data problem
  - Structuring a research project into a set of slides
    - Motivation, goals, background, methodology and results
  - Using a different clustering method than k-mean
    - How to cluster numerical data with the Density-based spatial clustering of applications with noise (DBSCAN)
    - How to set up the setting parameters of the DBSCAN
  - Using code from other scientists
    - Re-use rather than rewriting from scratch
    - Replicability of work as the first step for new research

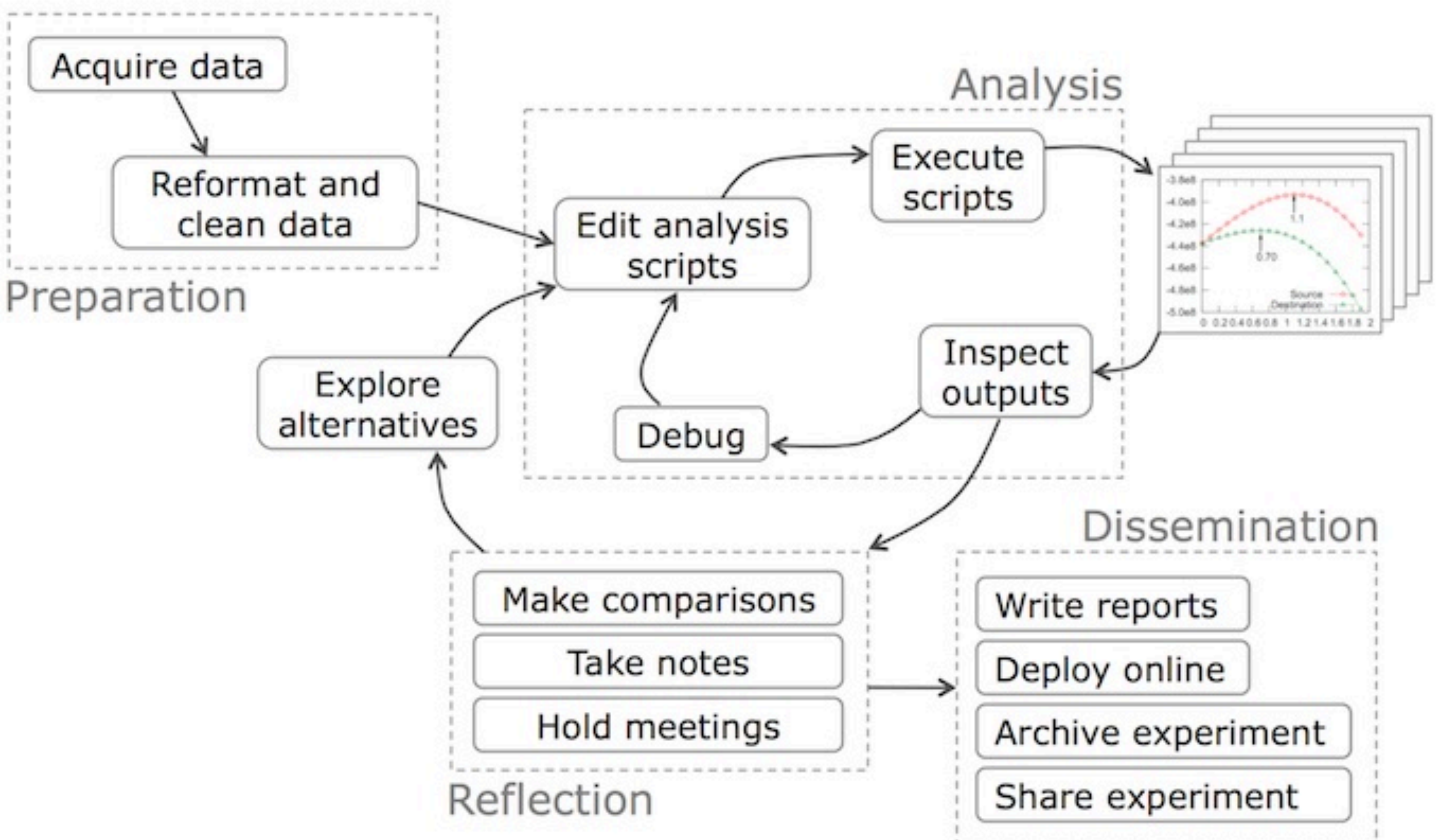
Reading

Reading

- Philip Guo. Data Science Workflow: Overview and Challenges. BLOG@CACM, October 30, 2013
- Philip Guo. Software Tools to Facilitate Research Programming. PhD Dissertation, 2012
- Michael R. Wyatt II, Travis Johnston, Mia Papas, and Michela Taufer. Development of a Scalable Method for Creating Food Groups Using the NHANES Dataset and MapReduce. In proceedings of IEEE eScience, 2019.

Data Workflow

Data Workflow


Preparation

Acquire data

Reformat and clean data

Analysis

Edit analysis scripts

Execute scripts

Inspect outputs

Debug

Explore alternatives

Reflection

Make comparisons

Take notes

Hold meetings

Dissemination

Write reports

Deploy online

Archive experiment

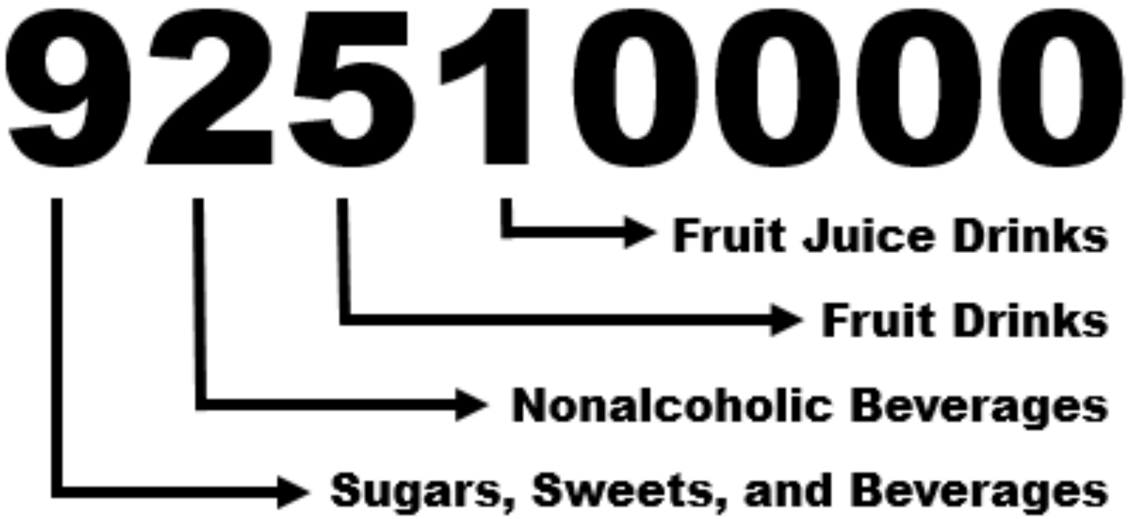
Share experiment

Courtesy of Philip Guo: goo.gl/y42rp1

Clustering Food Items

USDA Food Classification

- Used in dietary datasets to assign food items to groups
- Subjective and general
- Categorical, not nutrient-driven



92510000

Fruit Juice Drinks

Fruit Drinks

Nonalcoholic Beverages

Sugars, Sweets, and Beverages

What’s Wrong with USDA Food Groups?

- Food Groups are **subjective**
    - Based on human expertise
    - Based on dietary trends
- Using **subjective** food groups for dietary studies give **bad results**
- Lack of standard food groups for use in dietary studies

How Do We Fix Food Groups?

- Food Groups should be **objective**
  - Based on micro- and macro-nutrient content
- We need a standard set of food groups for dietary studies
- We need **scalable** methods for identifying food groups

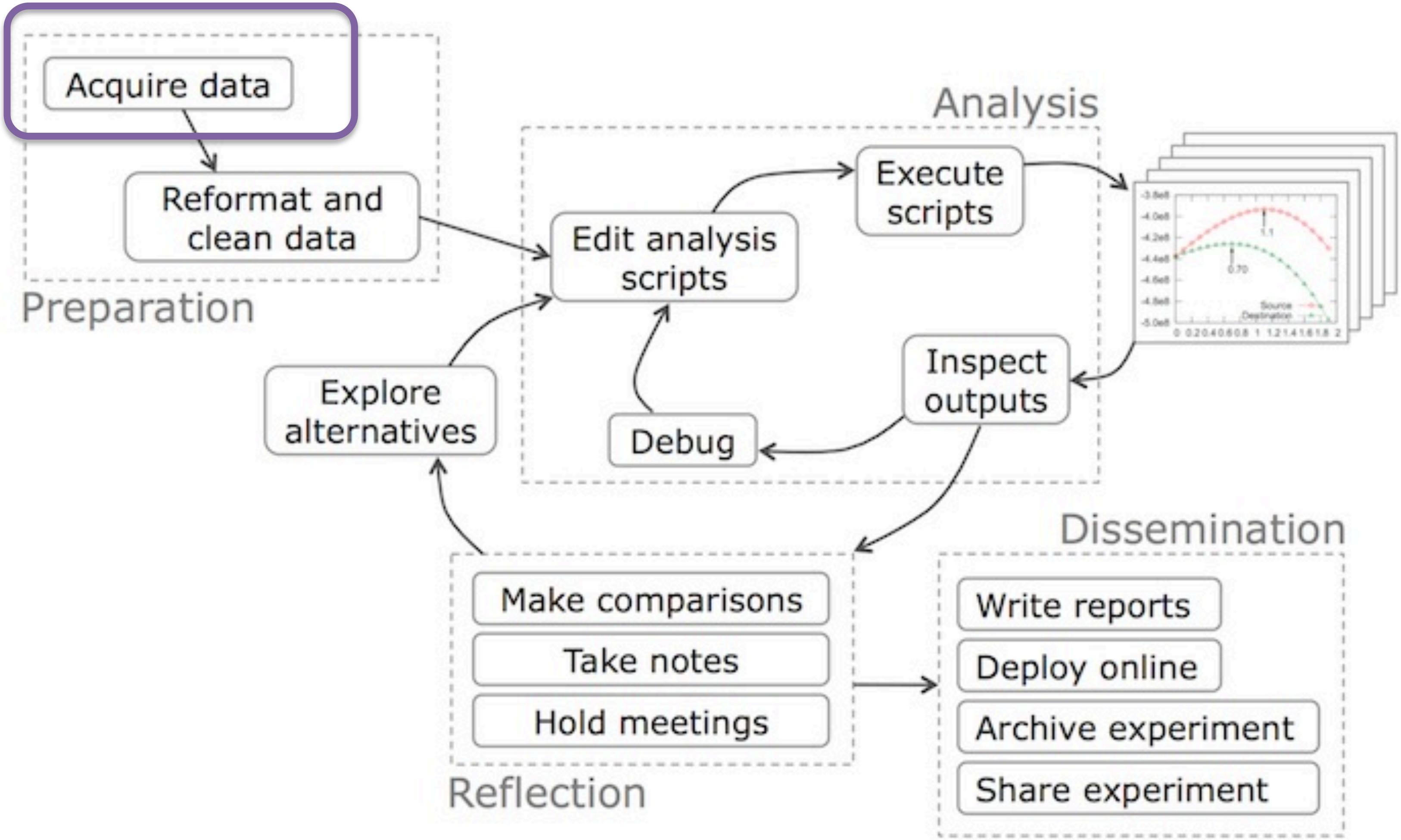
"Define an objective methods to group food items in a dietary datasets based on the item micro- and macro-nutrient content"

Paper Contributions

Paper's Contributions

- Identify an **relevant, open-source dataset** with subjective classification of food item
- Define an objective method to classify food items into **nutrient-driven food groups**
- Parallelize our methods using a **scalable framework** such as Apache Spark's **based on MapReduce**
- Critically **compare and contrast** the subjective and our nutrient-driven food classification

Data Workflow



Courtesy of Philip Guo: goo.gl/y42rp1

Relevant Open-source Dataset

- Use a dataset with well-known and broadly used data format

NHANES: National Health and Nutrition Examination Survey

- Medical, demographic, and dietary records
- Available to the public for free
- *Contains subjective food groups provided by USDA*

Data available at: http://www.cdc.gov/nchs/nhanes/nhanes_questionnaires.htm

NHANES Dietary Data

- Dietary intake of 64,653 Americans
- 7,494 unique food items
- 1,587,750 food entries
- 46 nutrient features for each food item
  - Macronutrients (e.g., fats, carbohydrates)
  - Micronutrients (e.g., vitamins, minerals)

NHANES Dietary Data

- Dietary intake of 64,653 Americans
- 7,494 unique food items
- 1,587,750 food entries
- 46 nutrient features for each food item
    - Macronutrients (e.g., fats, carbohydrates)
    - Micronutrients (e.g., vitamins, minerals)

Structure of Dietary Data Item

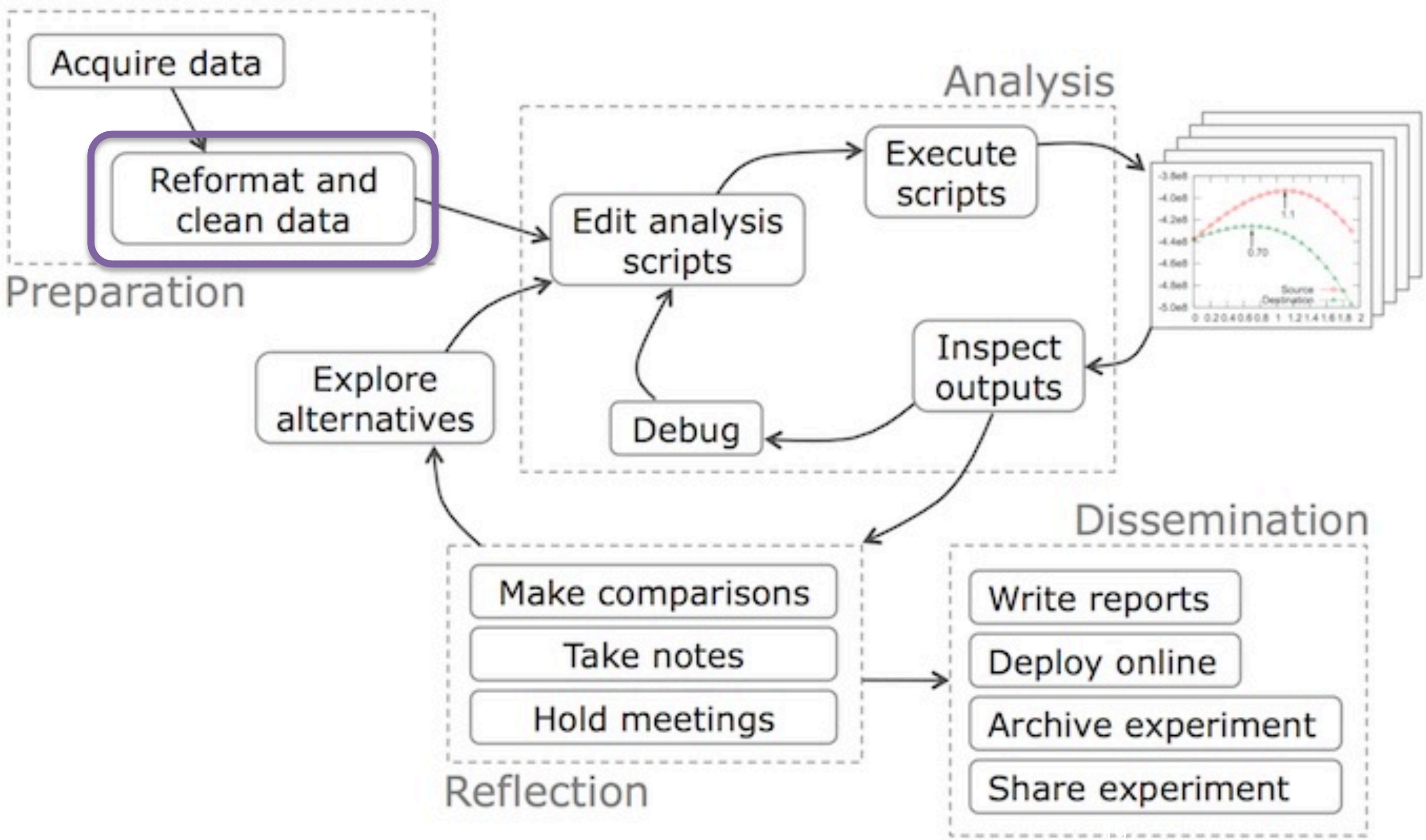
<143672, 92510000, 3, 0, 8:15am, 7, 3, 10.1, 4, 3.45, 10, 178, …>

- Participant ID
- **USDA Food Code**

- Meta Data

- Macronutrients
- Micronutrients

Data Workflow



Preparation

Acquire data

Reformat and clean data

Analysis

Edit analysis scripts

Execute scripts

Inspect outputs

Debug

Explore alternatives

Reflection

Make comparisons

Take notes

Hold meetings

Dissemination

Write reports

Deploy online

Archive experiment

Share experiment

Courtesy of Philip Guo: goo.gl/y42rp1

NHANES Snapshot

Food	Mod Code	Weight (g)	Protein (g)	Sodium (mg)
Milk	*0*	244	8	107
Milk	*0*	122	4	54
Milk	*0*	100	3	44
Milk	*0*	300	10	132
Milk	*0*	10	0	4
Milk	*0*	93	3	41
Cereal	*0*	30	2	231
Cereal	*10*	60	6	462
Cereal	*0*	0	3	300
Cereal	*0*	10		77
Steak	*0*	256	62	146

Missing Values

Food	Mod Code	Weight (g)	Protein (g)	Sodium (mg)
Milk	*0*	244	8	107
Milk	*0*	122	4	54
Milk	*0*	100	3	44
Milk	*0*	300	10	132
Milk	*0*	10	0	4
Milk	*0*	93	3	41
Cereal	*0*	30	2	231
Cereal	*10*	60	6	462
Cereal	*0*	0	3	300
Cereal	*0*	10		77
Steak	*0*	256	62	146

Weight-Based Nutrient Values

Food	Mod Code	Weight (g)	Protein (g)	Sodium (mg)
Milk	*0*	244	8	107
Milk	*0*	122	4	54
Milk	*0*	100	3	44
Milk	*0*	300	10	132
Milk	*0*	10	0	4
Milk	*0*	93	3	41
Cereal	*0*	30	2	231
Cereal	*10*	60	6	462
Cereal	*0*	0	3	300
Cereal	*0*	10		77
Steak	*0*	256	62	146

Redundant Data

Food	Mod Code	Weight (g)	Protein (g)	Sodium (mg)
Milk	*0*	244	8	107
Milk	*0*	122	4	54
Milk	*0*	100	3	44
Milk	*0*	300	10	132
Milk	*0*	10	0	4
Milk	*0*	93	3	41
Cereal	*0*	30	2	231
Cereal	*10*	60	6	462
Cereal	*0*	0	3	300
Cereal	*0*	10		77
Steak	*0*	256	62	146

Features on Different Scales

Food	Mod Code	Weight (g)	Protein (g)	Sodium (mg)
Milk	*0*	244	8	107
Milk	*0*	122	4	54
Milk	*0*	100	3	44
Milk	*0*	300	10	132
Milk	*0*	10	0	4
Milk	*0*	93	3	41
Cereal	*0*	30	2	231
Cereal	*10*	60	6	462
Cereal	*0*	0	3	300
Cereal	*0*	10		77
Steak	*0*	256	62	146

Preprocessing: Missing Values

Food	Mod Code	Weight (g)	Protein (g)	Sodium (mg)
Cereal	*0*	10		77

- Standard Approach:
  - Fill in missing values e.g., median or zero
- Drawback:
  - Introduces artificial data which creates bias in the data
- Our Solution:
  - Redundant food entries allows us to discard entries with missing values
- Observations:
  - 9,586 of 1,587,750 entries removed
  - 0 unique foods lost

Preprocessing: Missing Values

Food	Mod Code	Weight (g)	Protein (g)	Sodium (mg)
Milk	*0*	244	8	107
Milk	*0*	122	4	54
Milk	*0*	100	3	44
Milk	*0*	300	10	132
Milk	*0*	10	0	4
Milk	*0*	93	3	41
Cereal	*0*	30	2	231
Cereal	*10*	60	6	462
Cereal	*0*	0	3	300
Cereal	*0*	10		77
Steak	*0*	256	62	146

Preprocessing: Weight-Based Values

Food	Mod Code	Weight (g)	Protein (g)	Sodium (mg)
Milk	*0*	244	8	107
Milk	*0*	122	4	54
Milk	*0*	100	3	44
Milk	*0*	300	10	132
Milk	*0*	10	0	4
Milk	*0*	93	3	41

- Nutrient values are based on the weight of the food entry.
  - Unfair comparison between foods
  - Food entries of the same type do not match

Preprocessing: Weight-Based Values

Food	Mod Code	Weight (g)	Protein (g)	Sodium (mg)
Cereal	*0*	0	3	300

- Standard Approach:
  - Normalize with respect to weight – divide entry by weight
- Drawback:
  - Some entries have a weight of “0”
- Our Solution:
  - Redundant food entries allows us to discard entries with a weight of “0”
- Observations:
  - 10,983 of 1,587,750 entries removed
  - 0 unique foods lost

Preprocessing: Weight-Based Values

Food	Mod Code	Weight (g)	Protein (g)	Sodium (mg)
Milk	*0*	1 (244)	0.033	0.439
Milk	*0*	1 (122)	0.033	0.443
Milk	*0*	1 (100)	0.030	0.440
Milk	*0*	1 (300)	0.033	0.440
Milk	*0*	1 (10)	0	0.400
Milk	*0*	1 (93)	0.032	0.441
Cereal	*0*	1 (30)	0.067	7.70
Cereal	*10*	1 (60)	0.100	7.70
Cereal	*0*	0	3	300
Cereal	*0*	10		77
Steak	*0*	1 (256)	0.242	0.570

Preprocessing: Redundant Data

Food	Mod Code	Weight (g)	Protein (g)	Sodium (mg)
Milk	*0*	1 (244)	0.033	0.439
Milk	*0*	1 (10)	0	0.400
Cereal	*0*	1 (30)	0.067	7.70
Cereal	*10*	1 (60)	0.100	7.70

- Standard Approach:
  - Select one entry for each unique food item
- Drawback:
  - Nutrient densities don’t always match for the same food item – How do we pick a representative entry?
    - Modification codes
    - Rounding error

Preprocessing: Redundant Data

Food	Mod Code	Weight (g)	Protein (g)	Sodium (mg)
Milk	*0*	1 (244)	0.033	0.439
Milk	*0*	1 (10)	0	0.400
Cereal	*0*	1 (30)	0.067	7.70
Cereal	*10*	1 (60)	0.100	7.70

- Our Solution:
  - Remove entries with a non-zero modification code
  - Average top 5 entries for each food when sorted by weight
- Observations:
  - 32 unique foods lost

Preprocessing: Redundant Data

Food	Mod Code	Weight (g)	Protein (g)	Sodium (mg)
Milk	*0*	1 (244)	0.033	0.439
Milk	*0*	1 (122)	0.033	0.443
Milk	*0*	1 (100)	0.030	0.440
Milk	*0*	1 (300)	0.033	0.440
Milk	*0*	1 (10)	0	0.400
Milk	*0*	1 (93)	0.032	0.441
Cereal	*0*	1 (30)	0.067	7.70
Cereal	*10*	1 (60)	0.100	7.70
Cereal	*0*	0	3	300
Cereal	*0*	10		77
Steak	*0*	1 (256)	0.242	0.570

BIG ORANGE BIG IDEAS

Preprocessing: Redundant Data

Food	Mod Code	Weight (g)	Protein (g)	Sodium (mg)
Milk	*0*	1 (244)	0.032	0.441
Milk	*0*	1 (122)	0.033	0.443
Milk	*0*	1 (100)	0.030	0.440
Milk	*0*	1 (300)	0.033	0.440
Milk	*0*	1 (10)	0	0.400
Milk	*0*	1 (93)	0.032	0.441
Cereal	*0*	1 (30)	0.067	7.70
Cereal	*10*	1 (60)	0.100	7.70
Cereal	*0*	0	3	300
Cereal	*0*	10		77
Steak	*0*	1 (256)	0.242	0.570

Preprocessing: Different Scales

Food	Mod Code	Weight (g)	Protein (g)	Sodium (mg)
Milk	*0*	1	0.032	0.441
Cereal	*0*	1	0.067	7.70
Steak	*0*	1	0.242	0.570

- Standard Approach:
  - Standardize – divide by largest value
- Drawback:
  - Removes effects of highly skewed feature distributions

Preprocessing: Different Scales

Food	Mod Code	Weight (g)	Protein (sd)	Sodium (sd)
Milk	*0*	1	-0.726	-0.593
Cereal	*0*	1	-0.416	1.16
Steak	*0*	1	1.13	-0.561

- Our Solution:
  - **Z-score standardization**: convert original values into values indicating how many standard deviations above or below a value is from the mean of the original distribution
- Observations:
  - Feature scales are similar, but not the same
  - Distributions are not dramatically altered

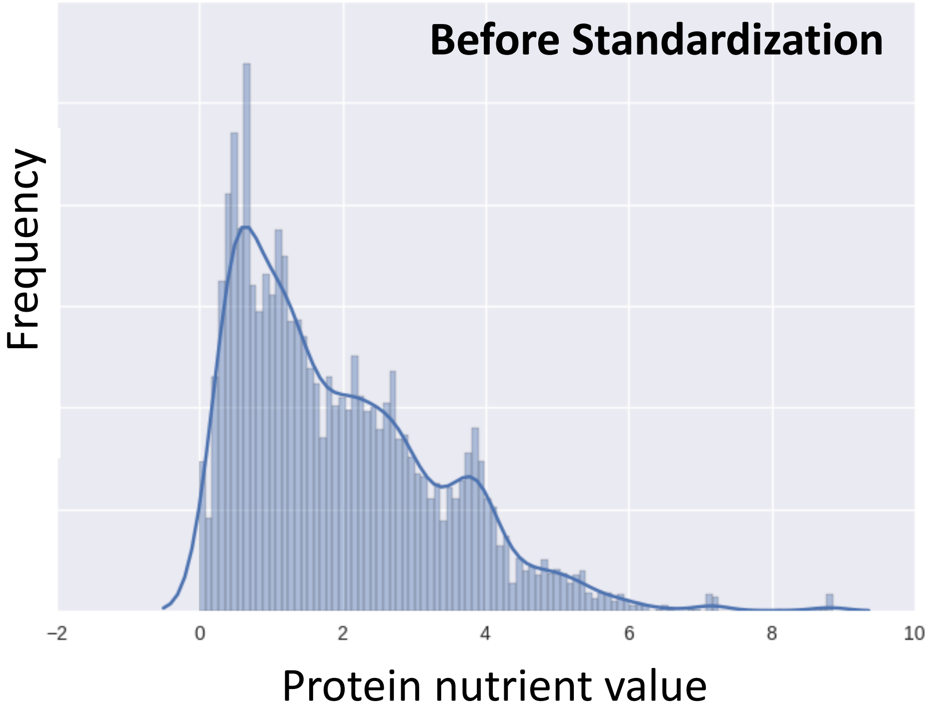
Preprocessing: Different Scales

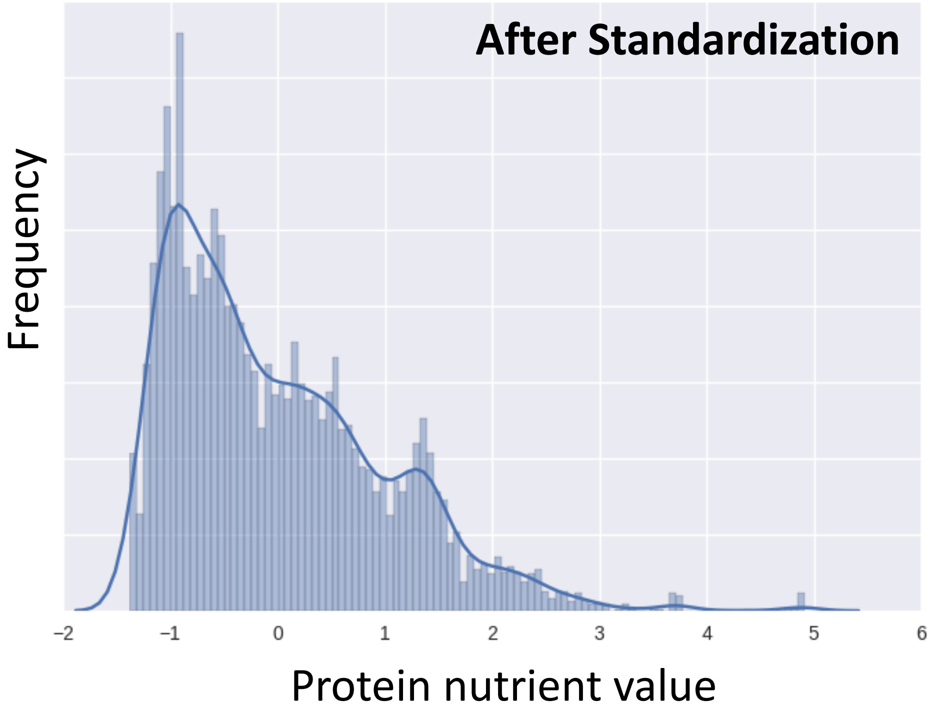
Food	Mod Code	Weight (g)	Protein (g)	Sodium (mg)
Milk	*0*	1	0.032	0.441
Cereal	*0*	1	0.067	7.70
Steak	*0*	1	0.242	0.570

Food	Mod Code	Weight (g)	Protein (sd)	Sodium (sd)
Milk	*0*	1	-0.726	-0.593
Cereal	*0*	1	-0.416	1.16
Steak	*0*	1	1.13	-0.561

Example of Standardization

- Macro-nutrient, protein, with its content distribution before and after standardization





NHANSE Snapshot After Preparation

Food	Mod Code	Weight (g)	Protein (g)	Sodium (mg)
Milk	*0*	1	-0.726	-0.593
Milk	*0*	1	0.033	0.443
Milk	*0*	1	0.030	0.440
Milk	*0*	1	0.033	0.440
Milk	*0*	1	0	0.400
Milk	*0*	1	0.032	0.441
Cereal	*0*	1	-0.416	1.16
Cereal	*10*	1	0.100	7.70
Cereal	*0*	0	3	300
Cereal	*0*	10		77
Steak	*0*	1	1.13	-0.561

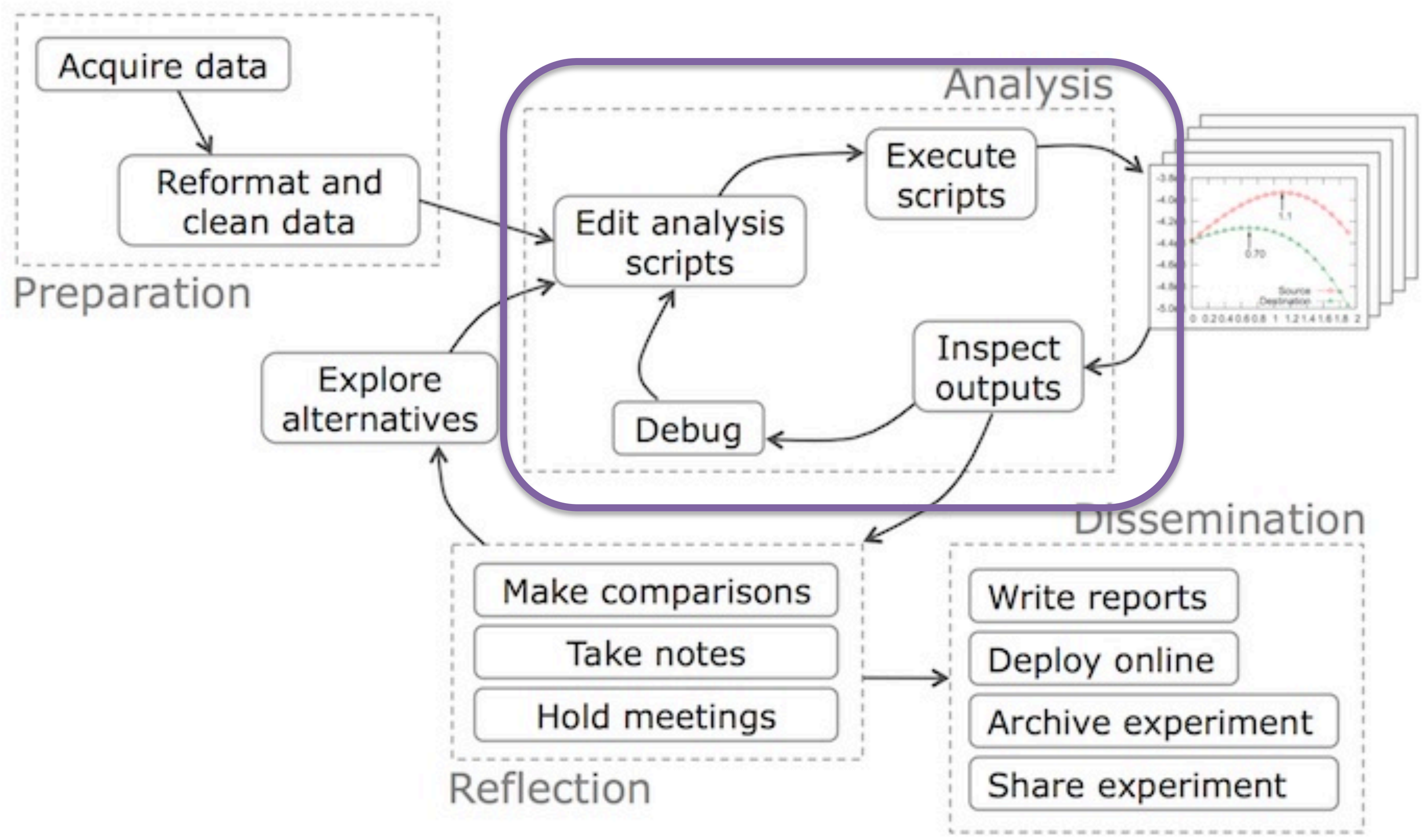
NHANSE Snapshot After Preparation

Food	Mod Code	Weight (g)	Protein (g)	Sodium (mg)
Milk	*0*	1	-0.726	-0.593
Milk	*0*	1	0.033	0.443
Milk	*0*	1	0.030	0.440
Milk	*0*	1	0.033	0.440
Milk	*0*	1	[illegible]	0.400
Milk	*0*	1	0.032	0.441
Cereal	*0*	1	-0.416	1.16
Cereal	*10*	1	0.100	7.70
Cereal	*0*	0	3	300
Cereal	*0*	10		77
Steak	*0*	1	1.13	-0.561

BEFORE: 7,494 unique food items

AFTER: 7,462 unique food items

Data Workflow



Preparation

Acquire data

Reformat and clean data

Analysis

Edit analysis scripts

Execute scripts

Inspect outputs

Debug

Explore alternatives

Reflection

Make comparisons

Take notes

Hold meetings

Dissemination

Write reports

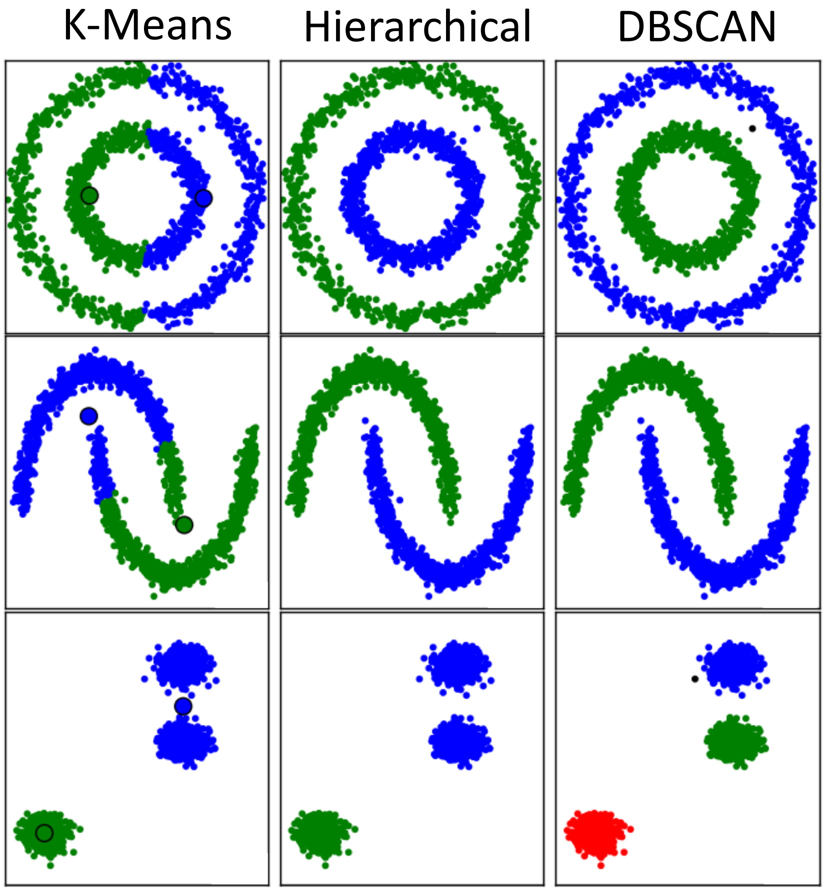
Deploy online

Archive experiment

Share experiment

Courtesy of Philip Guo: goo.gl/y42rp1

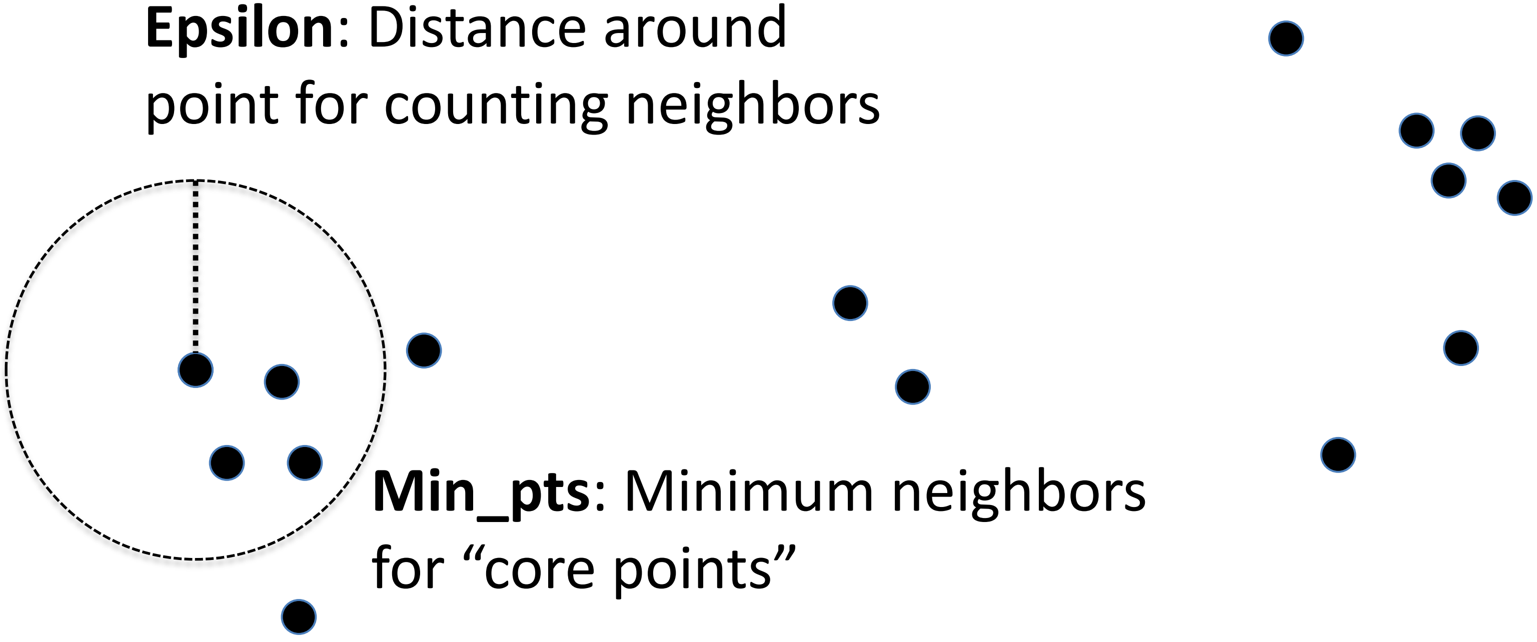
Selecting a Good Clustering Algorithm



Feature	K-Means	Hierarchical	DBSCAN
Resource Efficient	Yes	No	Yes
Noise Insensitive	No	No	Yes
Outlier Detection	No	No	Yes
Spheroid Clusters	Yes	Yes	Yes
Non-Spheroid Clusters	No	Yes	Yes
Undefined Cluster Count	No	Yes	Yes

Image: http://scikit-learn.org/stable/modules/clustering.html

DBSCAN Algorithm

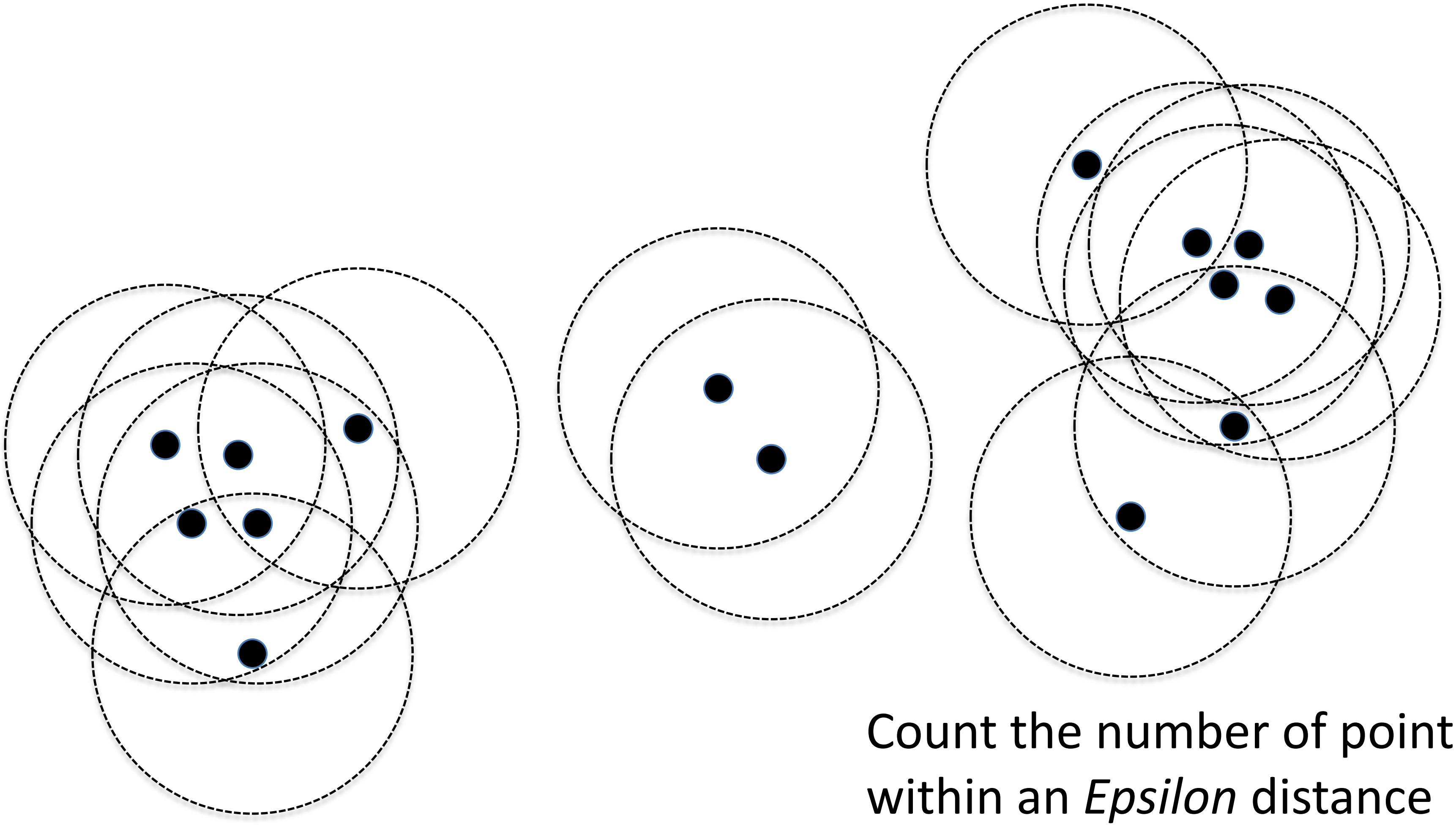


Epsilon: Distance around point for counting neighbors

Min_pts: Minimum neighbors for “core points”

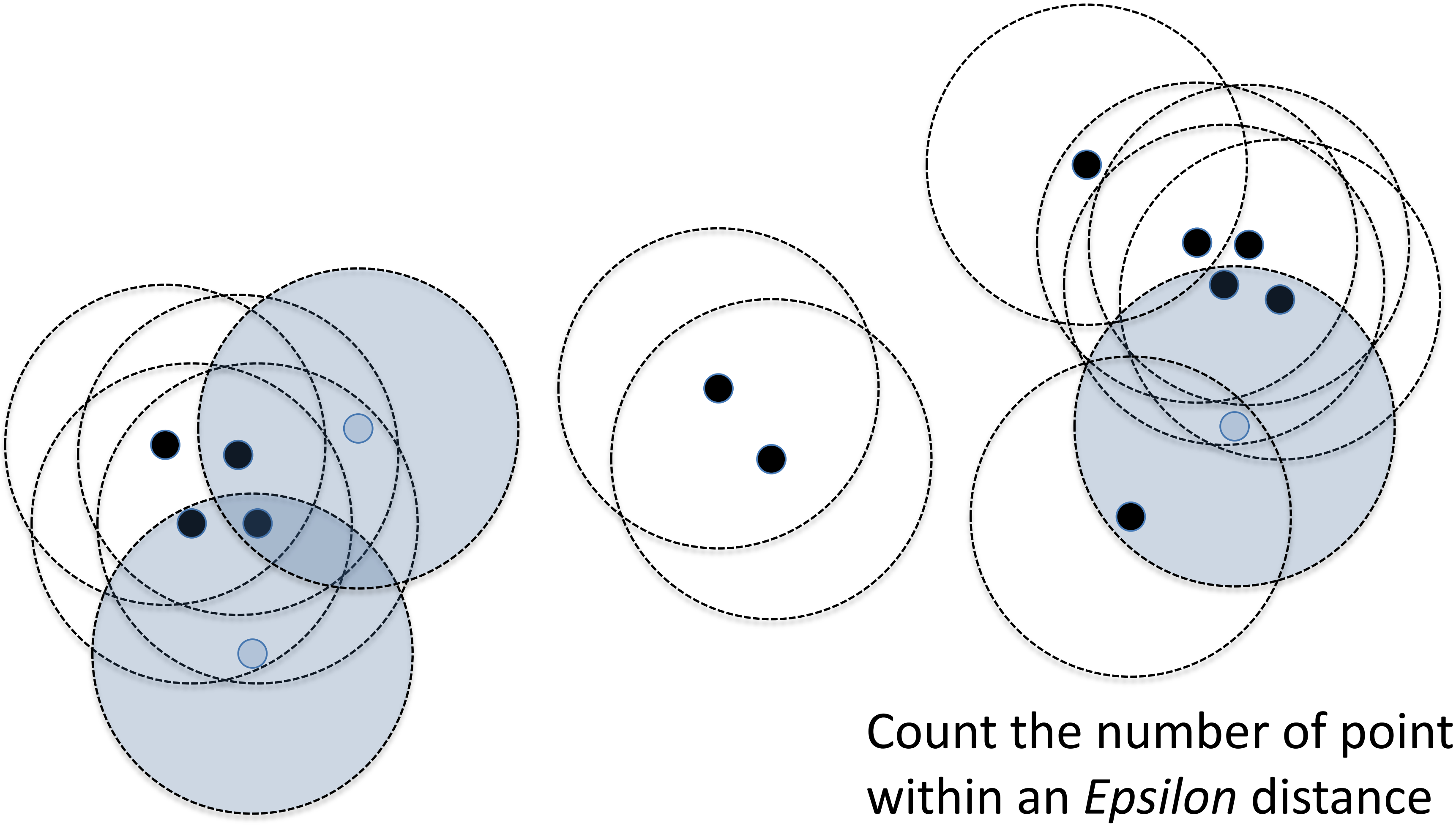
Together, these parameters define the minimum density for a cluster

DBSCAN Algorithm



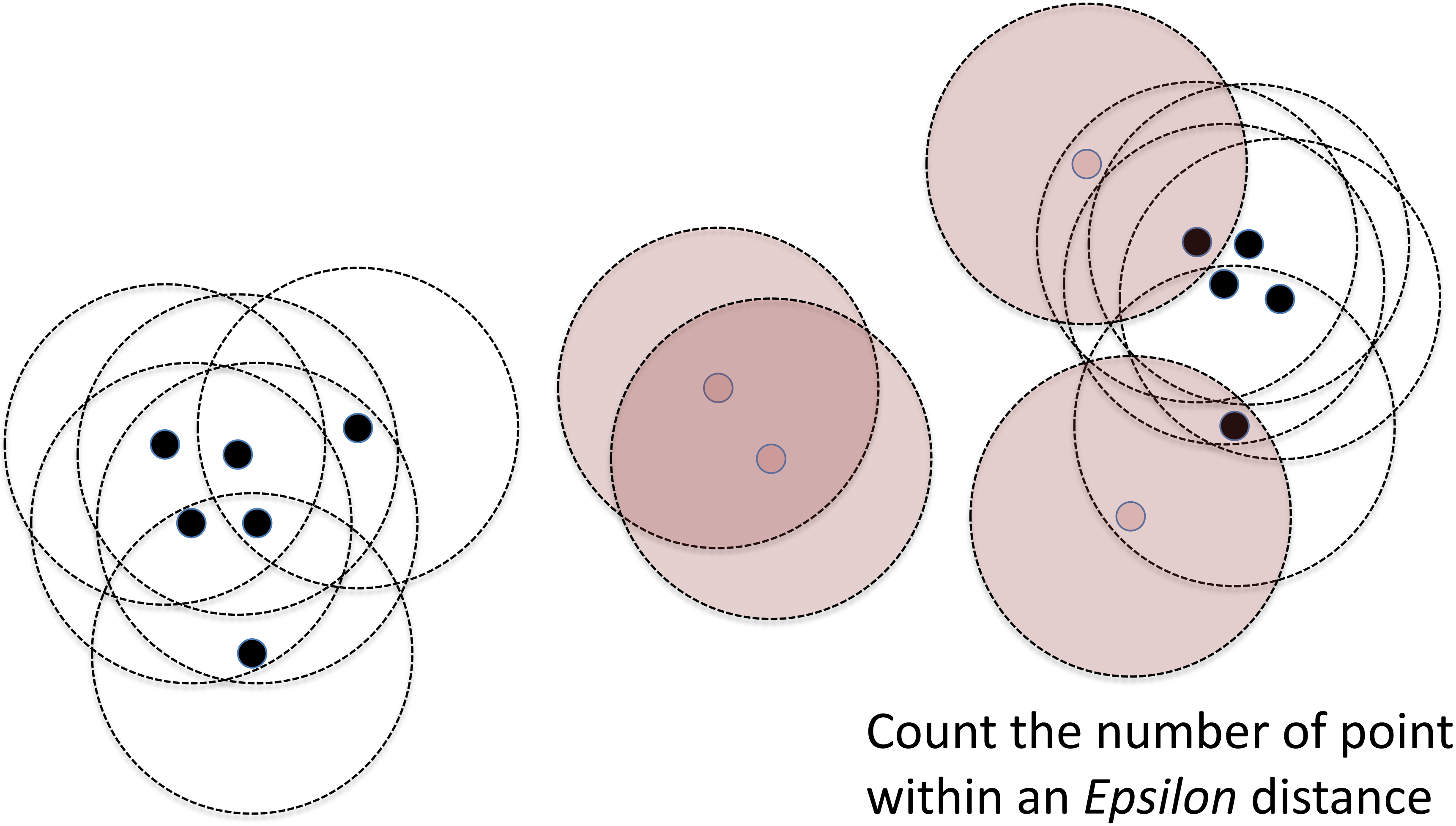
Count the number of points within an *Epsilon* distance

DBSCAN Algorithm: Border Points



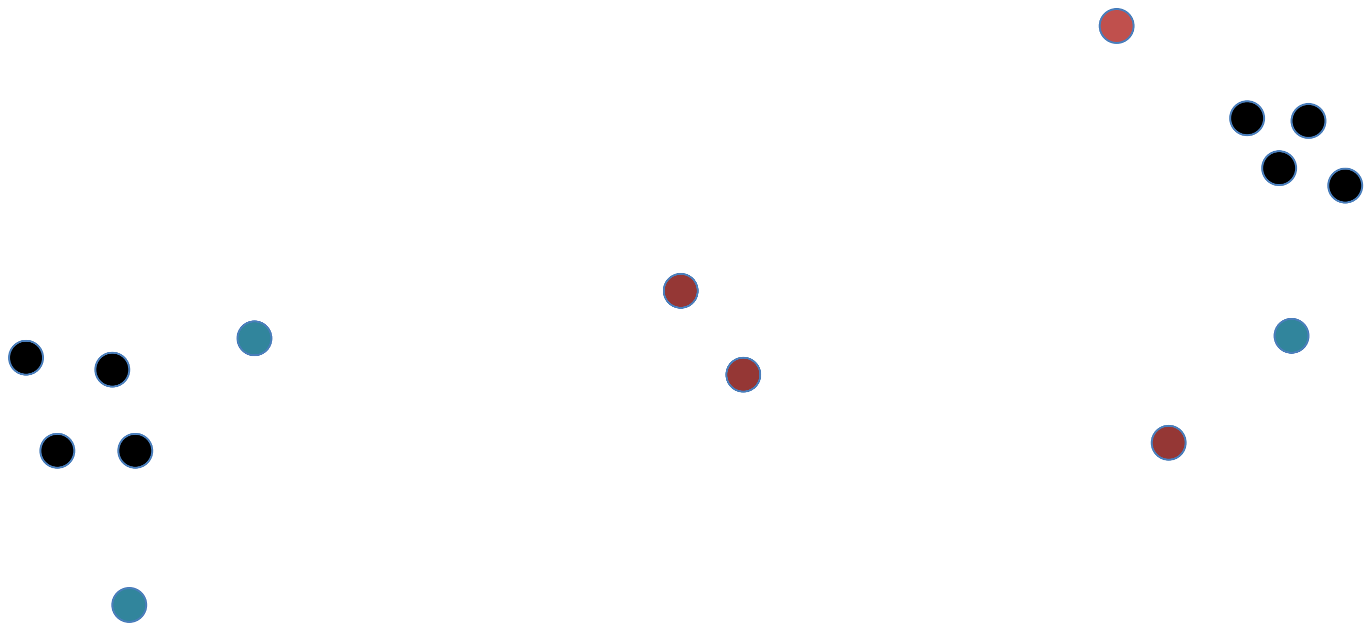
Count the number of points within an *Epsilon* distance

DBSCAN Algorithm: Noise Points



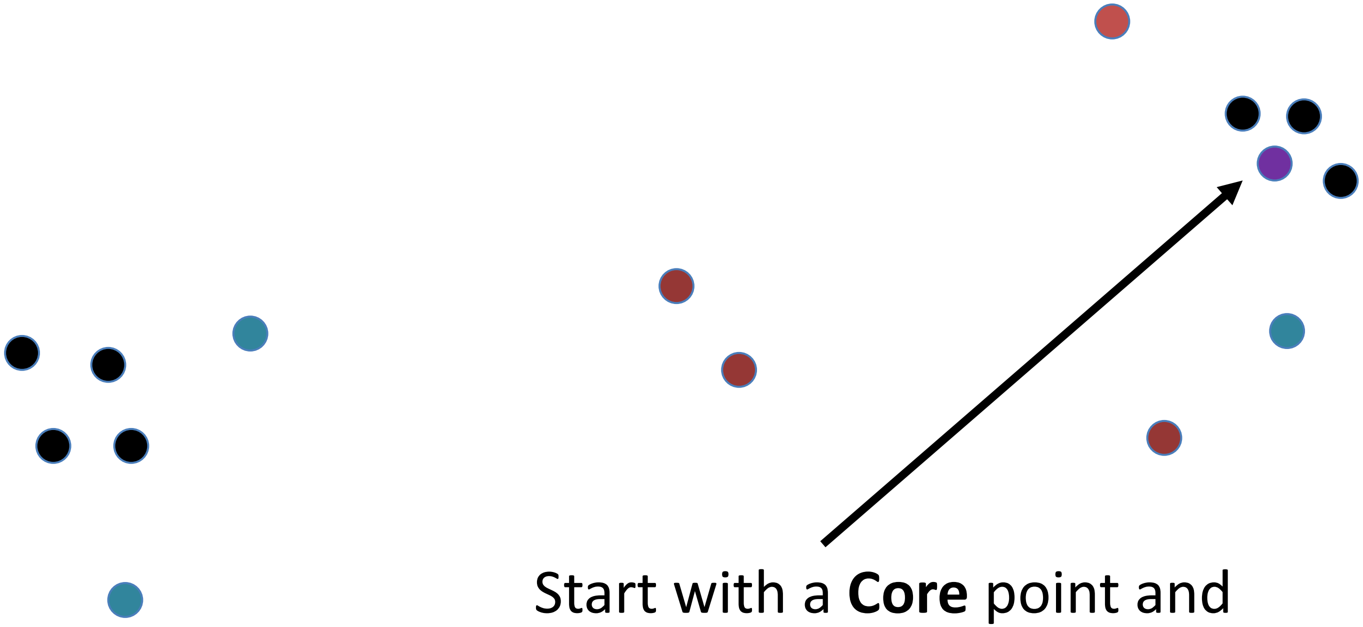
Count the number of points within an *Epsilon* distance

DBSCAN Algorithm



Identify **Core**, **Border**, and **Noise** points

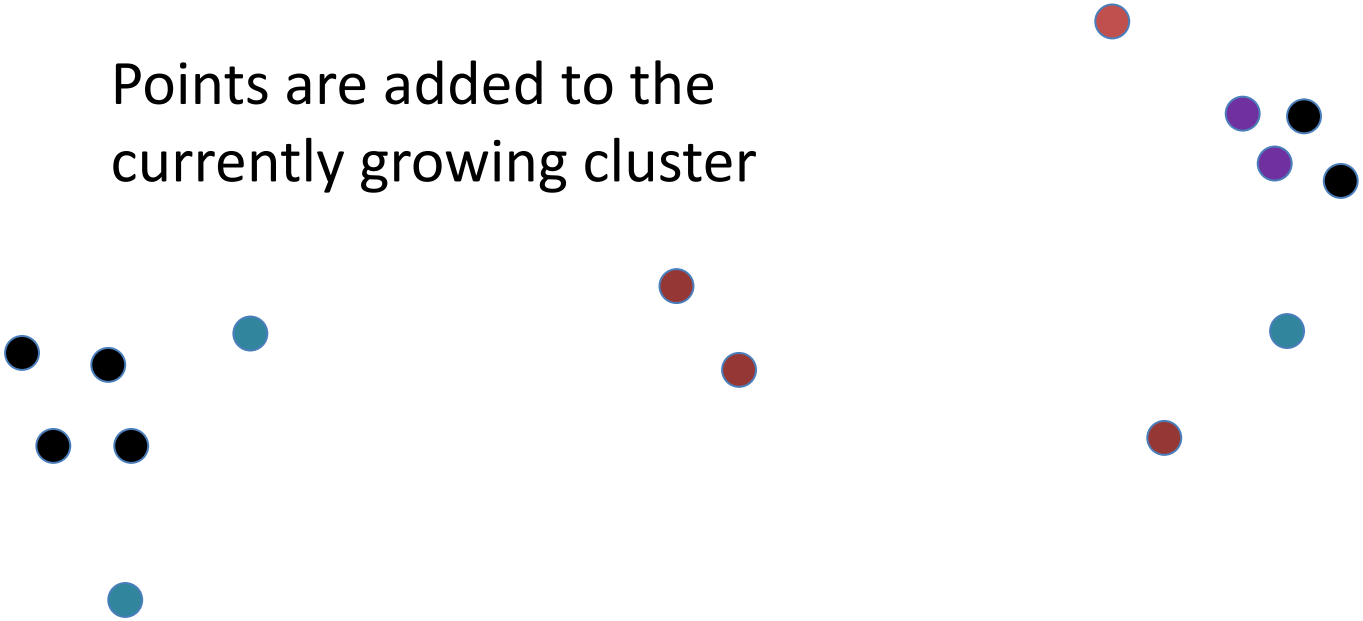
DBSCAN Algorithm



Start with a **Core** point and begin growing the clusters.

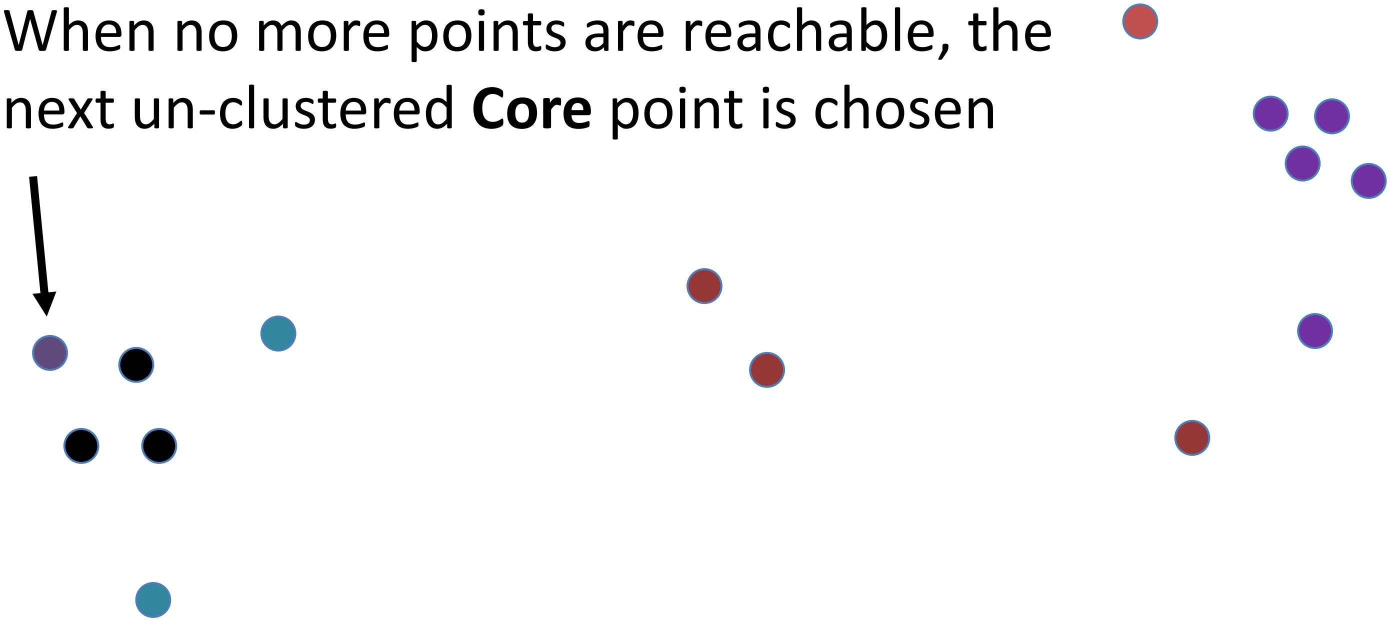
DBSCAN Algorithm

Points are added to the currently growing cluster

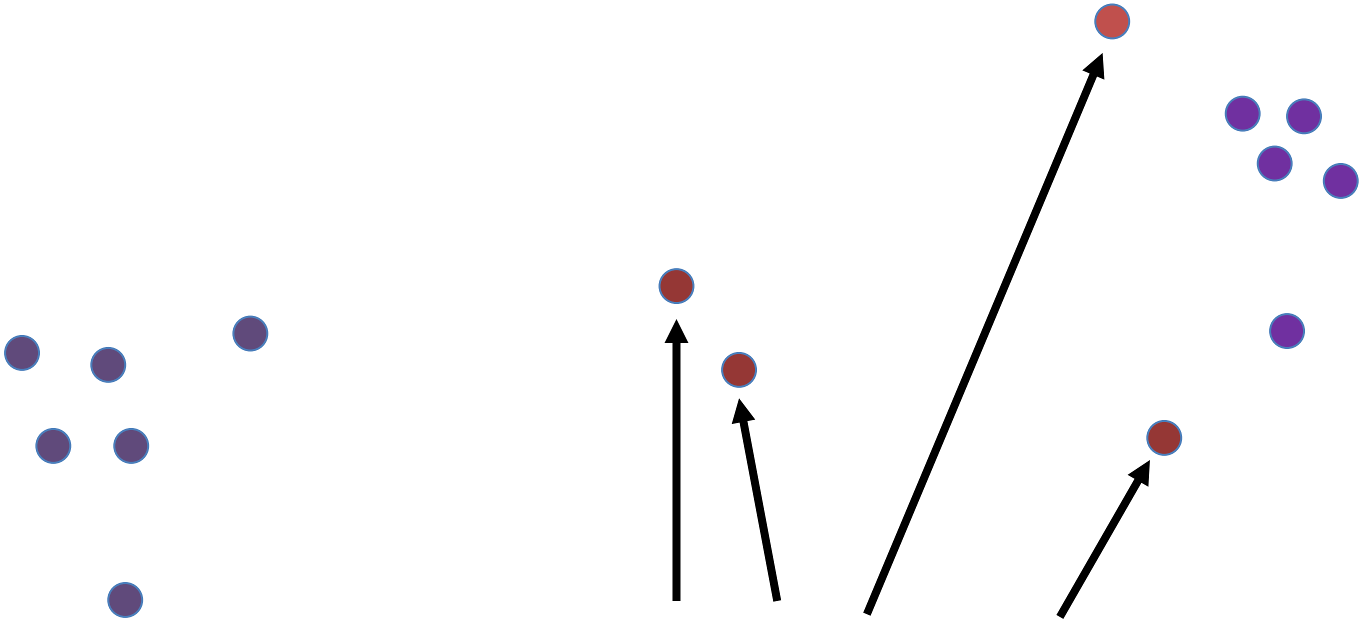


DBSCAN Algorithm

When no more points are reachable, the next un-clustered **Core** point is chosen



DBSCAN Algorithm



This pattern continues until only **Noise** points remain un-clustered.

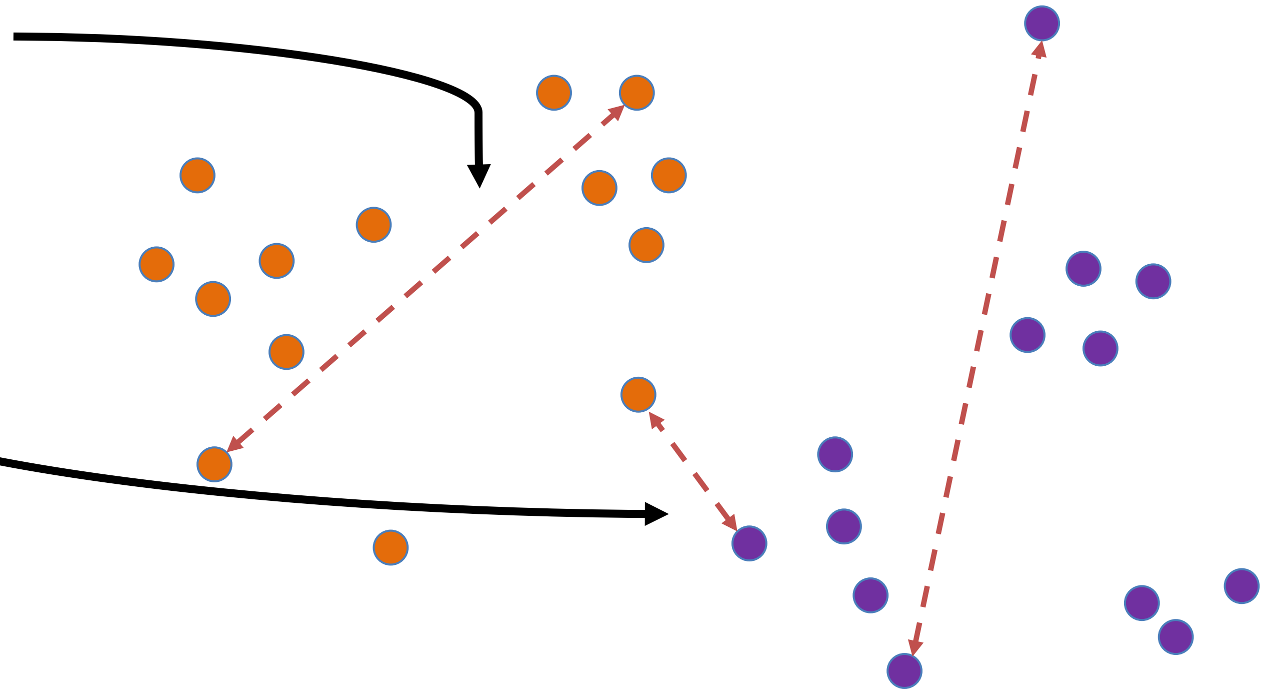
DBSCAN Algorithm

BDSCAN Demo

- Visualizing a DBSCAN clustering
  - Interactive interface at:
  - https://www.naftaliharris.com/blog/visualizing-dbscan-clustering/

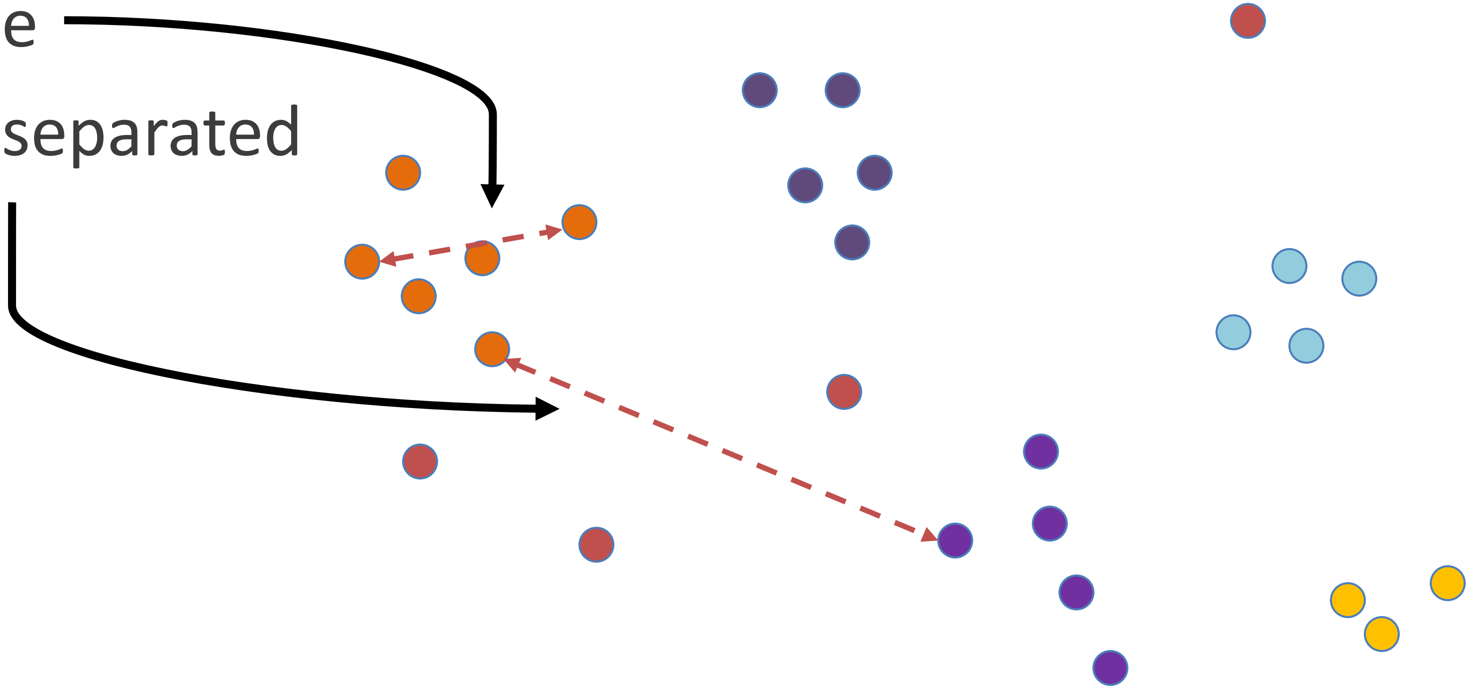
Bad Clustering

- Very Wide
- Not well separated



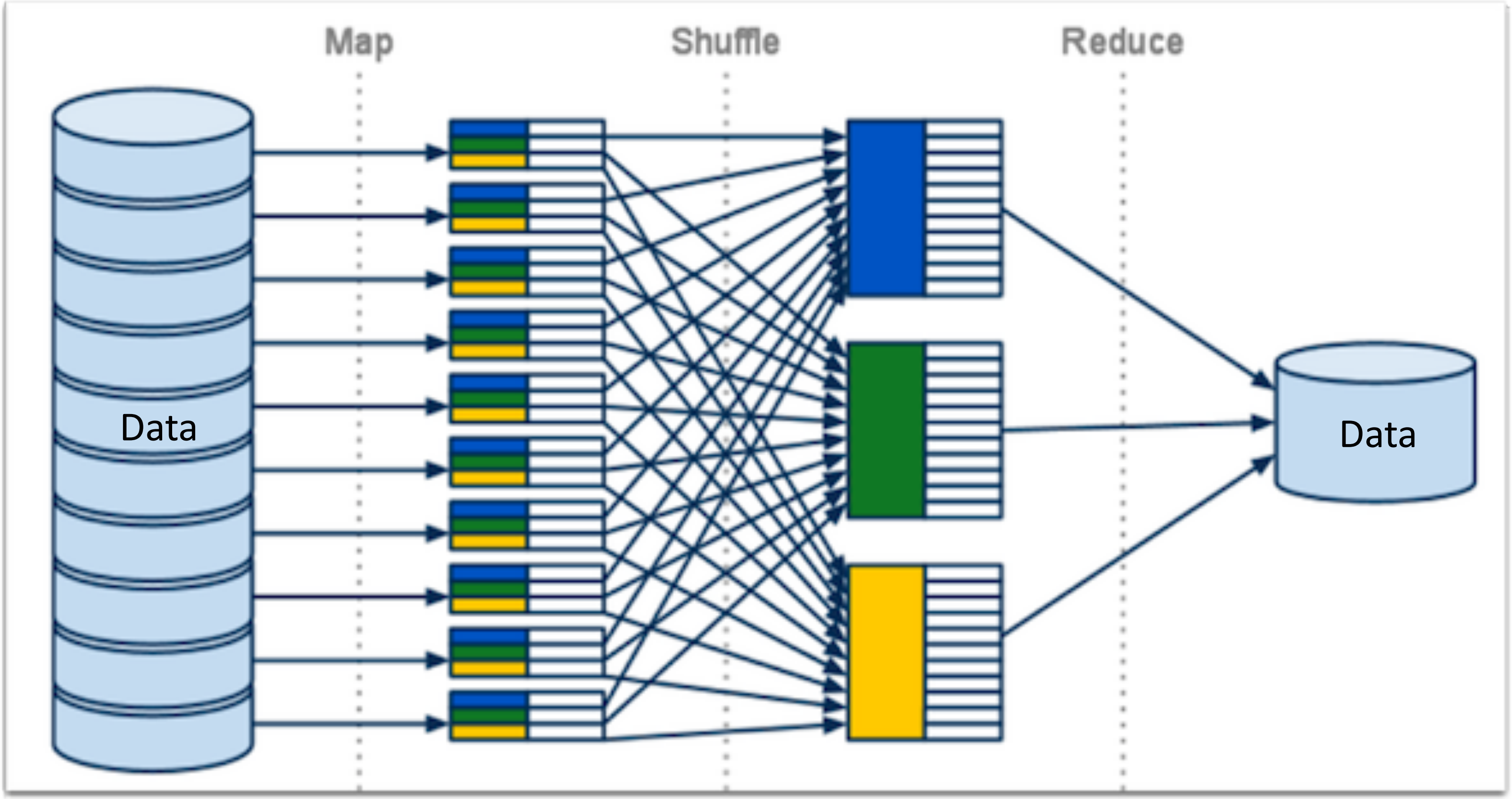
Good Clustering

- Dense
- Well separated

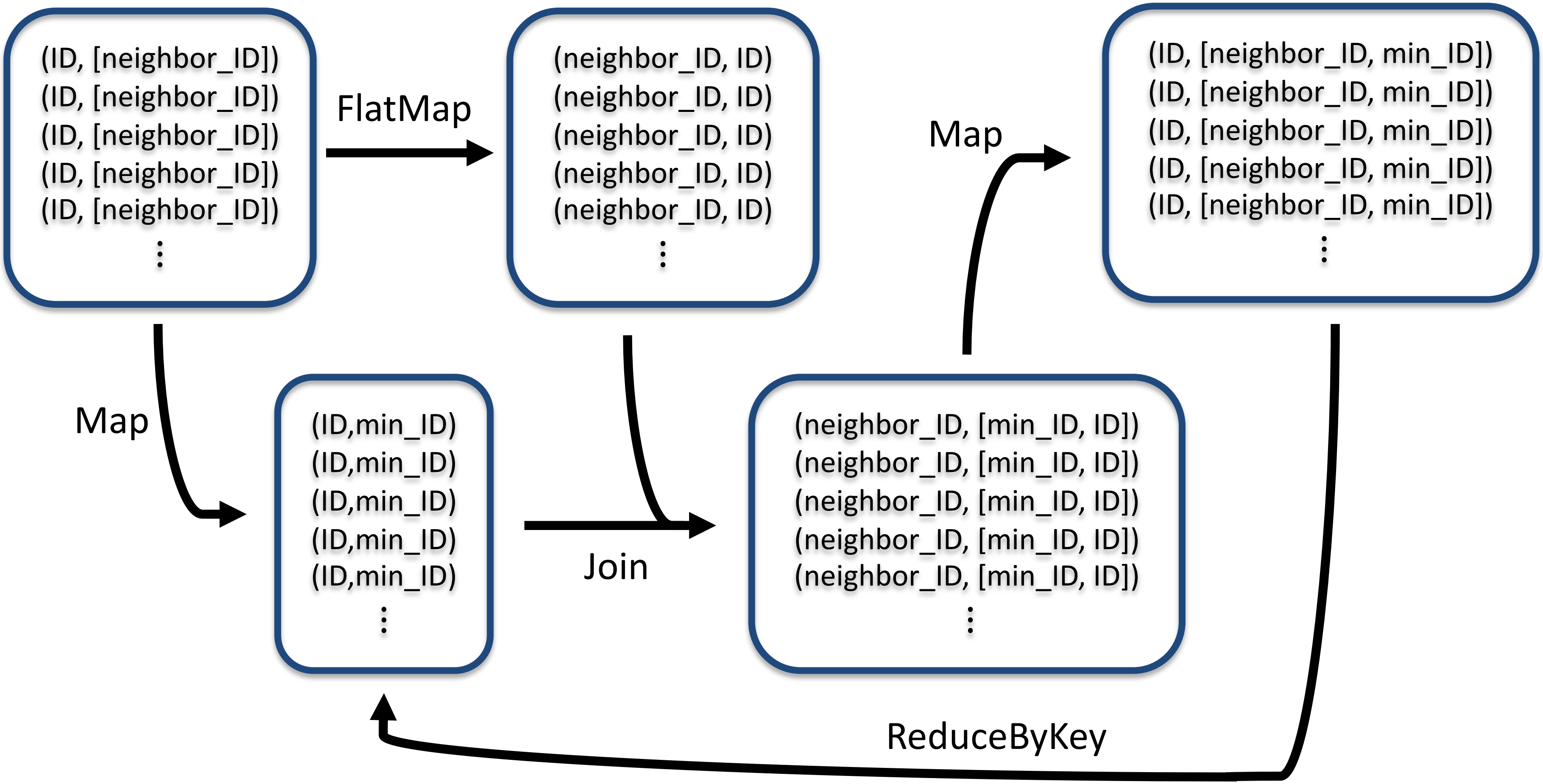


MapReduce Paradigm

Image: https://aimotion.blogspot.com/2012/08/introduction-to-recommendations-with.html



Parallel DBSCAN with Spark



(ID, [neighbor_ID])
(ID, [neighbor_ID])
(ID, [neighbor_ID])
(ID, [neighbor_ID])
(ID, [neighbor_ID])
⋮

FlatMap

(neighbor_ID, ID)
(neighbor_ID, ID)
(neighbor_ID, ID)
(neighbor_ID, ID)
(neighbor_ID, ID)
⋮

Map

(ID, [neighbor_ID, min_ID])
(ID, [neighbor_ID, min_ID])
(ID, [neighbor_ID, min_ID])
(ID, [neighbor_ID, min_ID])
(ID, [neighbor_ID, min_ID])
⋮

Map

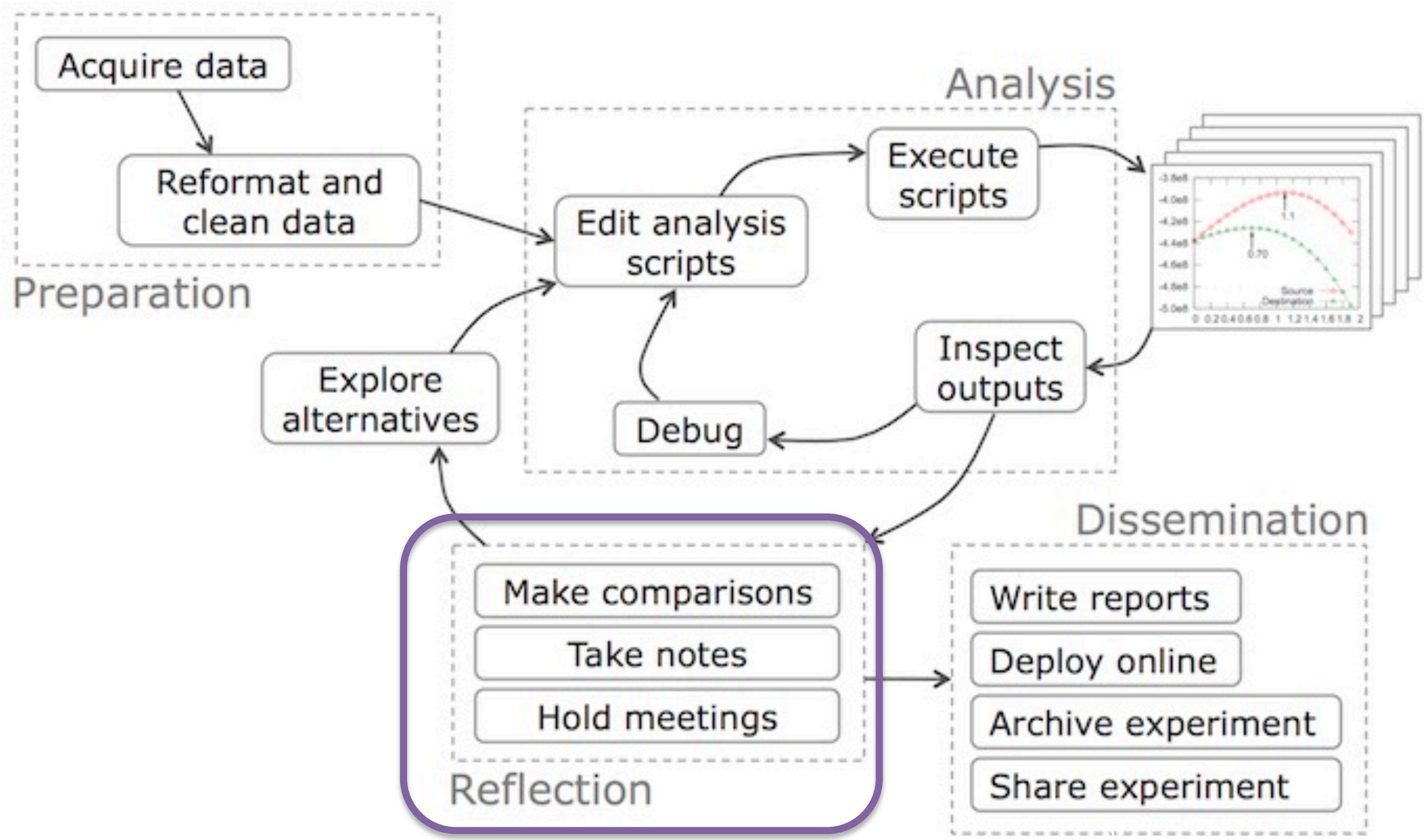
(ID,min_ID)
(ID,min_ID)
(ID,min_ID)
(ID,min_ID)
(ID,min_ID)
⋮

Join

(neighbor_ID, [min_ID, ID])
(neighbor_ID, [min_ID, ID])
(neighbor_ID, [min_ID, ID])
(neighbor_ID, [min_ID, ID])
(neighbor_ID, [min_ID, ID])
⋮

ReduceByKey

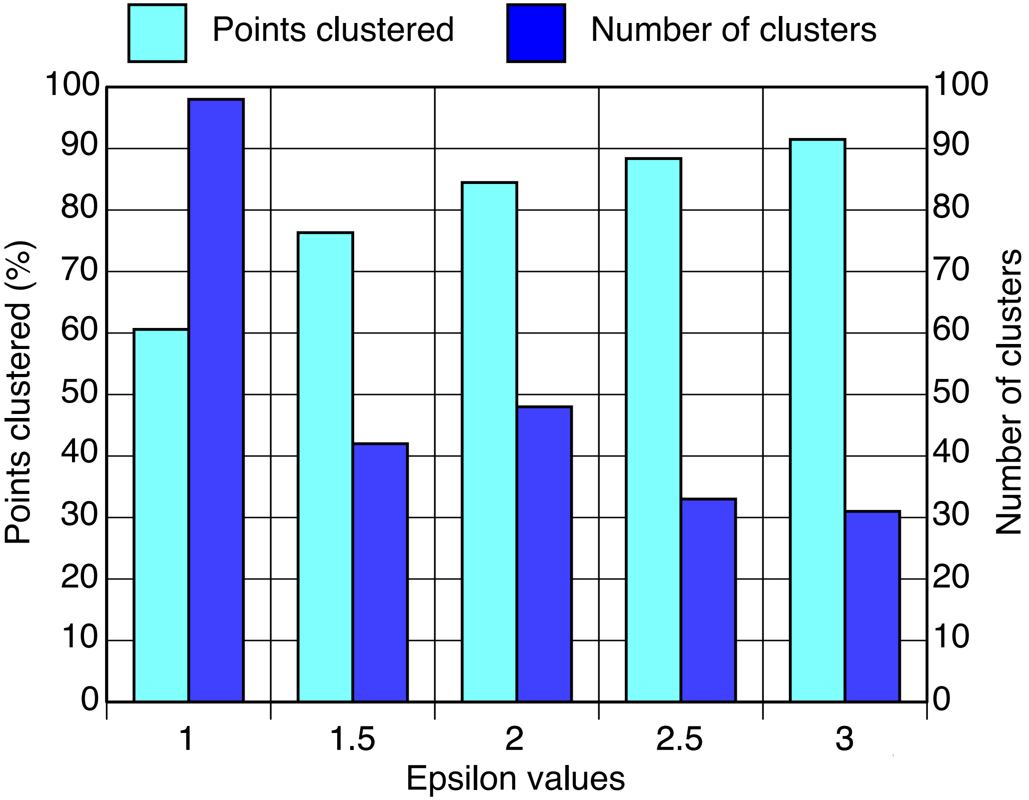
Data Workflow



Courtesy of Philip Guo: goo.gl/y42rp1

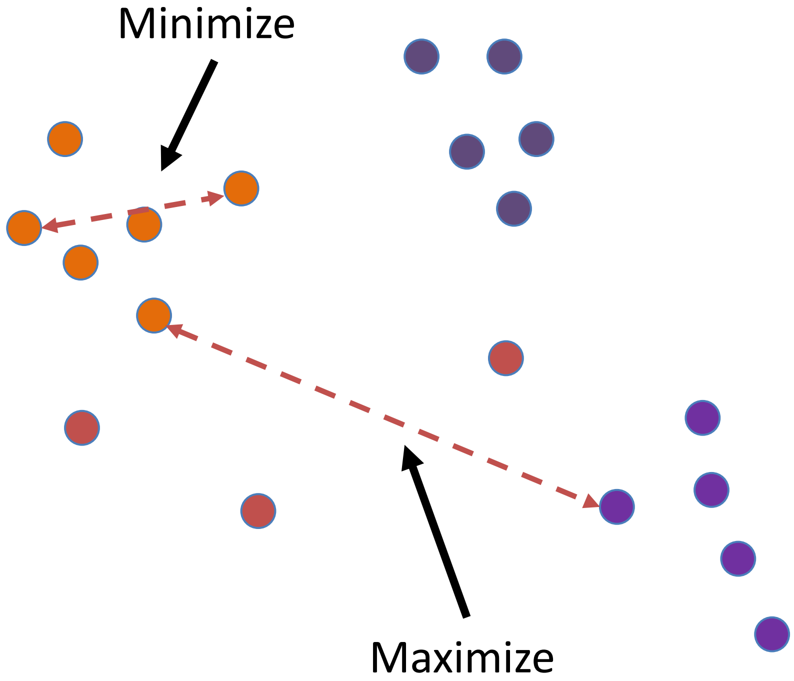
Experiment Setting and Metrics

- DBSCAN settings:
  - Epsilon = 1.0
  - Min_pts = 4
  - Euclidean distance



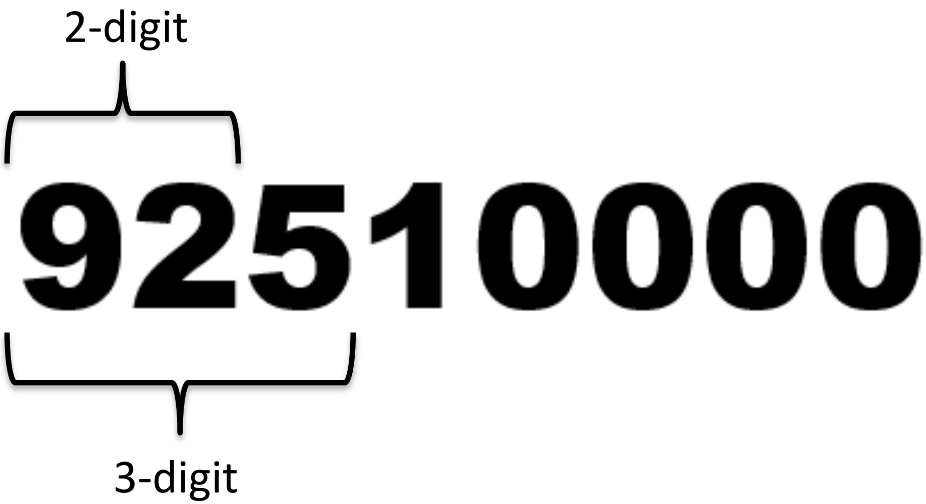
Experiment Setting and Metrics

- DBSCAN settings:
  - Epsilon = 1.0
  - Min_pts = 4
  - Euclidean distance
- Metrics of success:
  - Cluster diameter (min)
  - Cluster separation (max)



Experiment Setting and Metrics

- DBSCAN settings:
  - Epsilon = 1.0
  - Min_pts = 4
  - Euclidean distance
- Metrics of success:
  - Cluster diameter (min)
  - Cluster separation (max)
- Comparisons:
  - **Our clustering**
  - **USDA code clustering with 2-digit code**
  - **USDA code clustering with 3-digit code**

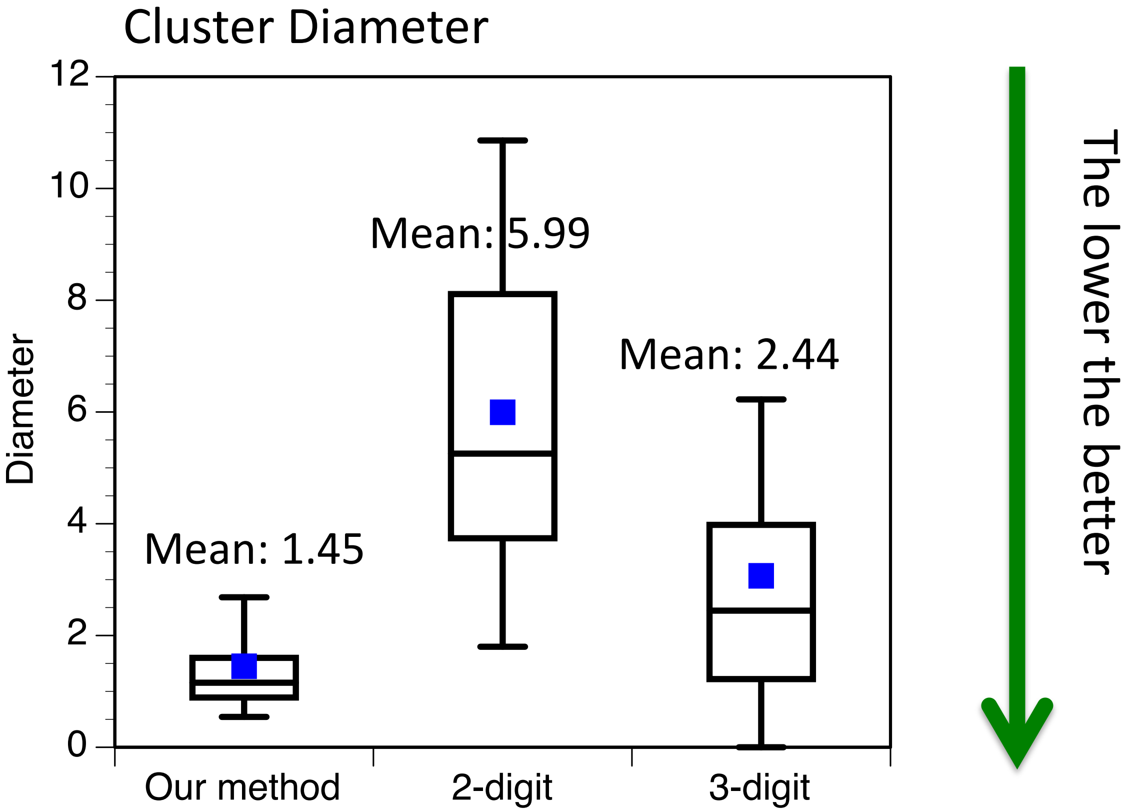


2-digit

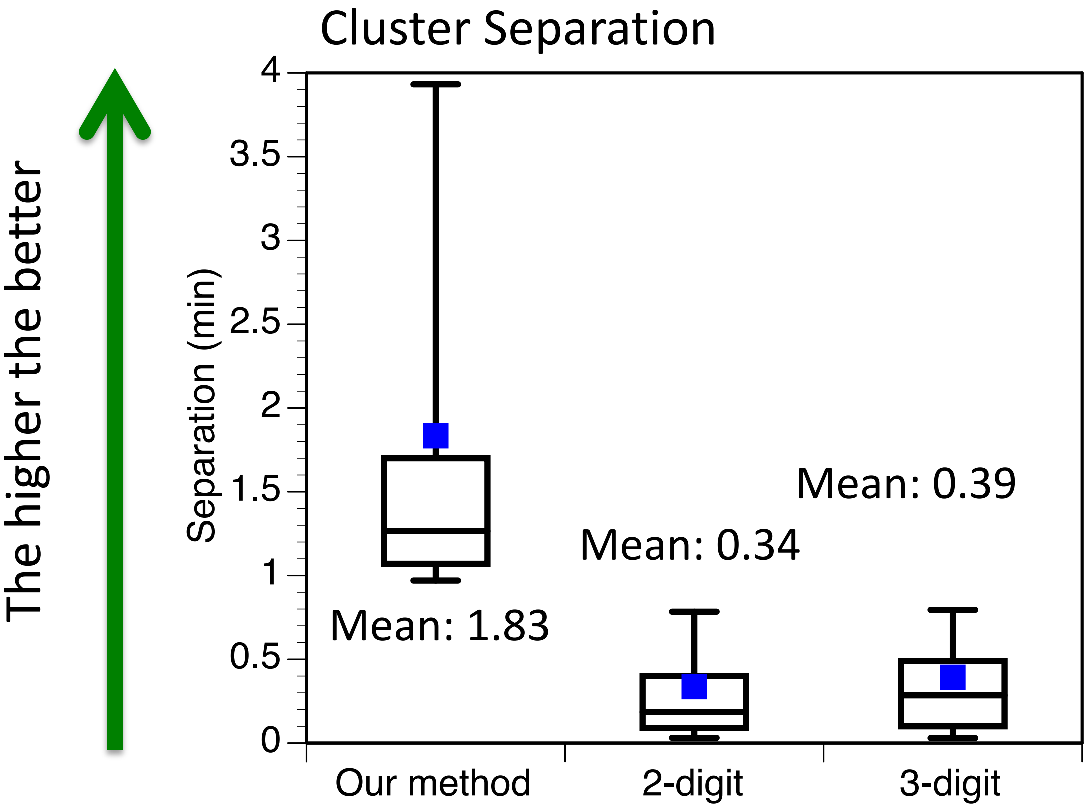
92510000

3-digit

Clustering Results

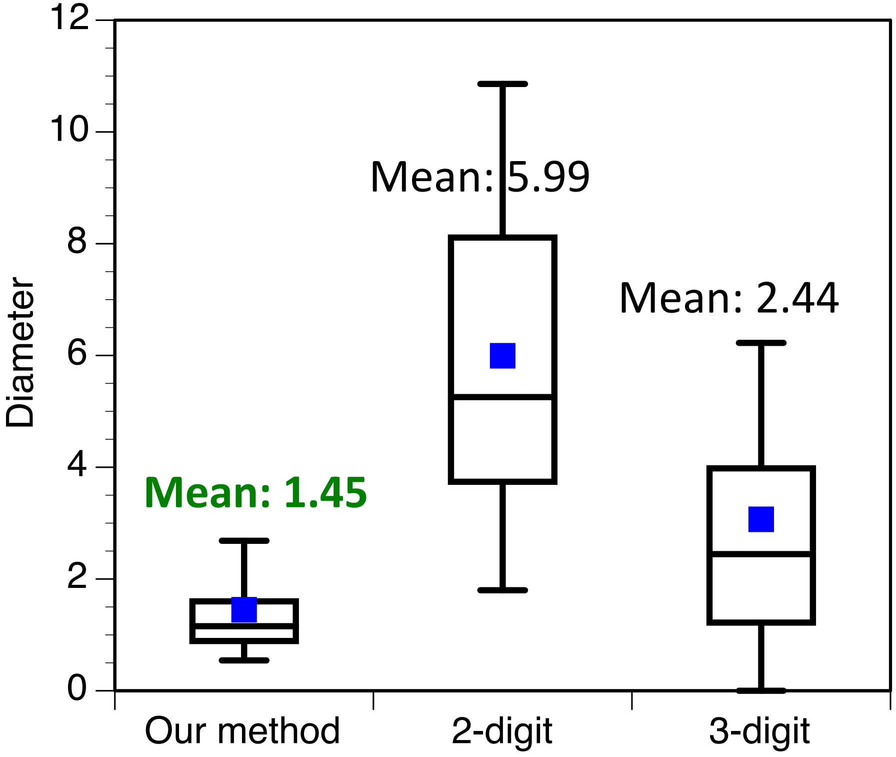


Clustering Results

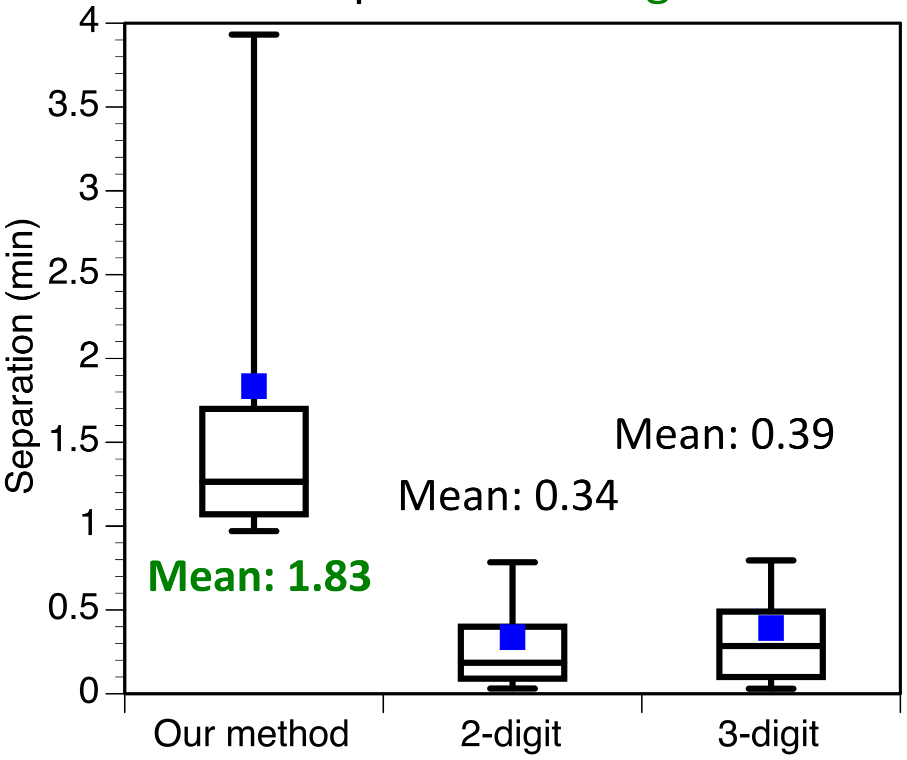


Clustering Results

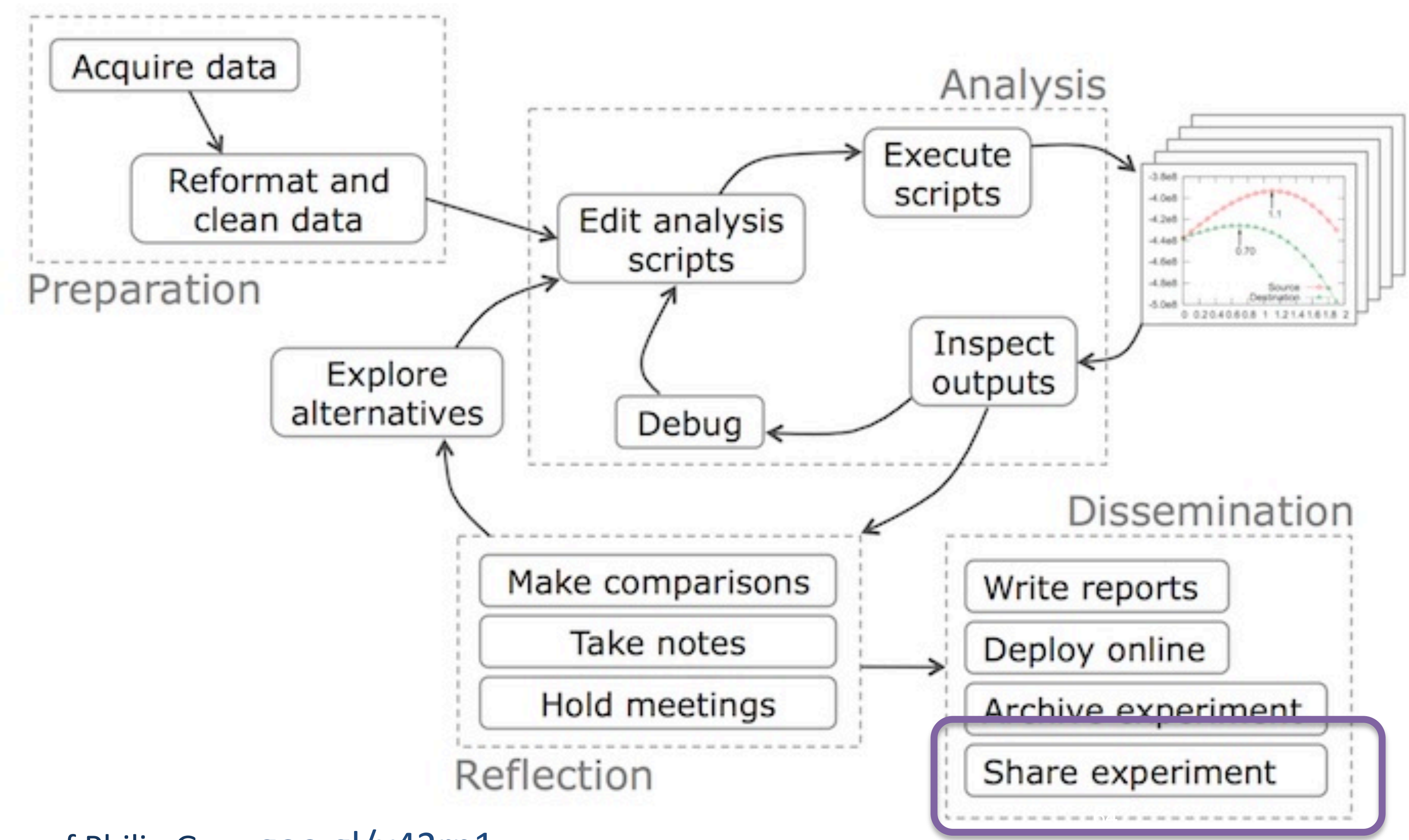
Cluster Diameter→ Lower=Better



Cluster Separation→ Higher=Better



Data Workflow



Courtesy of Philip Guo: goo.gl/y42rp1

Lessons Learned

- Using the traditional USDA classification of food items is misleading and can poorly advise patients with health issues

"Eat less fat!" "but USDA codes aren't based on nutrient content!"

- We propose a comprehensive data analysis workflow for NHANES dietary data
- Our approach clusters food items based **EXCLUSIVELY** on their nutritional content (i.e., micro- and macro-nutrients)
- Our methods is **scalable** (based on MapReduce) and produces **denser** and **better separated** food groups than USDA
  - Denser cluster diameter: 1.45 vs. 5.99 and 3.07
  - Better separated cluster separation: 1.83 vs. 0.34 and 0.39
- Our approach can provide a better indication of food group quality when advising patients with dietary restrictions