Purdue OATS Education Series – Data Analytics (1 of 2)

AgGateway Annual Meeting
November 2019

James V. Krogmeier
Professor and Associate Head of Electrical and Computer Engineering, Purdue University

AgGateway

OATS
Open Agriculture Technology and Systems

The Plan

- Day 1:
  - Background on OATS and OATS people
  - Purdue curricula @ AgGateway annual conference
  - Overview of data science as a field of study
  - Python basics with examples
- Day 2:
  - Continue python examples using OATS data sets
  - Survey of value

OATS Raison d’Être

The Open Agricultural Technologies and Systems Center works to enhance agricultural innovation in the areas of sensing, control, logistics, analytics, and data management via the catalyzing effects of open source technologies and educational outreach.

Observations / Issues

1. The **full value of data** is only realized when it can be **integrated** from multiple, context-rich sources in ways that adhere to the trust requirements of its owners.

Observations / Issues

2. The **data and algorithms** produced by publicly-funded food/ag research are **not easily obtainable** for verification, extension, or translation to practice.

Observations / Issues

3. The rapid rise of data-driven agriculture has left **many stakeholders short** of the proper data analytics, software development, and computational thinking **skills necessary to thrive**.

Observations / Issues

4. The information technologies used in the food/ ag industry and in university research **lag behind the state of the art** due in part to the **absence of** thriving **open source** communities that have propelled progress in the broader technology sector.

What Ag / Ag Research Need

Open Source

Volunteers/Industry writing freely available, public code

Open Standards

Standards grow out of implementation, not vice-versa

Market Forces

There doesn’t have to be only one way to do everything

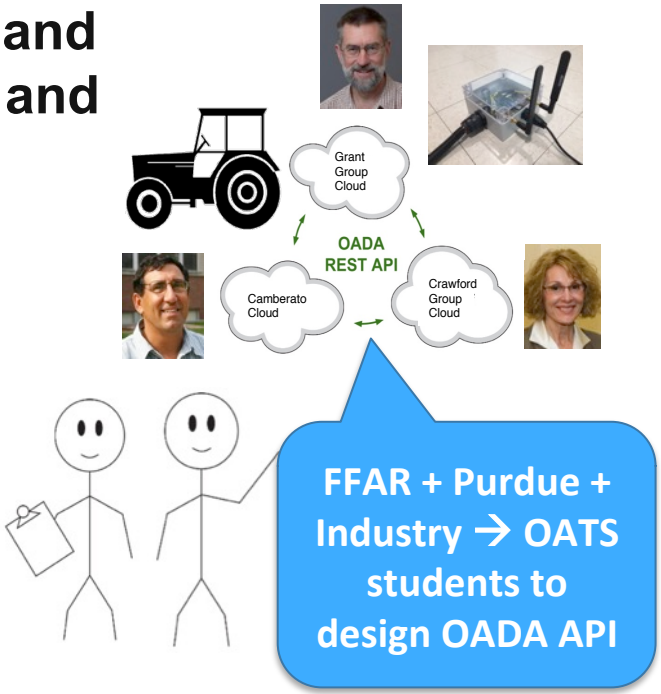
An Open Source Framework and Community for Sharing Data and Algorithms

Aims and Objectives

- Creation of Community Anchors
- Design of open educational experiences and materials
- Development and demonstration of useful open source hardware / software and toolkits

Real-time Data Exchange (RtX) Demonstrations

- Machinery/Crop/Soil Data – Automating Contexual Data Collection
- Food System Logistics – Tracking Activity and Safety Audits
- Remote Sensing – Multispec Open Source Release



Our Philosophy:

Data Exchange APIs are NOT designed in the abstract

- **Designed in response to specific need**
- **Publish Open Source**
- **Available to world for use and extension**

OATS Faculty and Staff

- **Aaron Ault** – Electrical & Computer Engr.
  - Software development, network security, crop / livestock operations
- **Mark Bell** – Electrical & Computer Engr.
  - Information theory, communication theory and systems, radar systems and signal processing, signal theory, free-space optical communications
- **Larry Biehl** – Information Technology at Purdue
  - Software development, IT systems administration, remote sensing

OATS Faculty and Staff

- **Dennis Buckmaster** – Ag & Bio Engr.
  - Ag Systems management, machine system design, forage/ field crop processes
- **James Camberato** – Agronomy
  - Plant production, soil nutrients, optimization of production potential
- **Melba Crawford** – Civil Engr. / Agronomy
  - Statistical pattern recognition for high dimensional data, multi-resolution image analysis, data mining

OATS Faculty and Staff

- **Amanda Deering** – Food Science
  - Human pathogenic bacteria in plants, routes of contamination, food safety.
- **Bruce Erickson** – Agronomy
  - Corn and soybean production, precision farming, instructional design, and competency-based education and assessment
- **John Evans** – Ag & Bio Engr.
  - Machine systems design, automation, precision agriculture.

OATS Faculty and Staff

- **Jim Garrison** – Aero / Astro Engr.
  - Earth remote sensing, Global Navigation Satellite Systems (GNSS), bistatic radar, signals of opportunity
- **Richard Grant** – Agronomy
  - Modeling of transport and deposition of gases and aerosols in the biosphere
- **Jian Jin** – Ag & Bio Engr.
  - Hyperspectral imaging systems, machine vision, big data modeling, automatic phenotyping systems

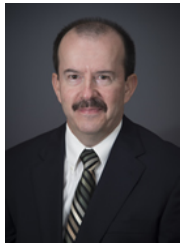
OATS Faculty and Staff

- **James Krogmeier** – Electrical & Computer Engr.
  - Statistical signal processing, sensor systems, wireless communications
- **David Love** – Electrical & Computer Engr.
  - Communication theory, multiple-input multiple-output (MIMO) wireless systems, millimeter wave communications
- **Haley F. Oliver** – Food Science
  - Persistence, prevalence, transmission of food pathogens; intervention strategies for food safety

OATS Faculty and Staff

- **Ankita Raturi** – Ag & Bio Engr.
  - agricultural informatics, human centered design, information modeling, software engineering
- **Amy R. Reibman** – Electrical & Computer Engr.
  - Image / video quality assessment, video transmission, video analytics
- **Dharmendra Saraswat** – Ag & Bio Engr.
  - Georeferenced information technologies, crowd sourcing applications, natural resource management

OATS Faculty and Staff



- **Mark A. Tucker** – Youth Development & Ag Education
  - Public engagement of science and technology, including risk communication, decision-making and development of strategies for engagement
- **Mark D. Ward** – Statistics
  - Data structures, algorithm analysis, large scale computation.

Additional faculty, staff, and students are welcome to join OATS!

Other Purdue Centers / Groups working with OATS

- Discovery Park
- CERIAS – Center for Education and Research in Information Assurance and Security
- Center for Food and Agricultural Business
- Joint Transportation Research Program
- Flexible and Efficient Spectrum Research Team
- Birck Nanotech Center: SMART Project

2019 Accomplishments

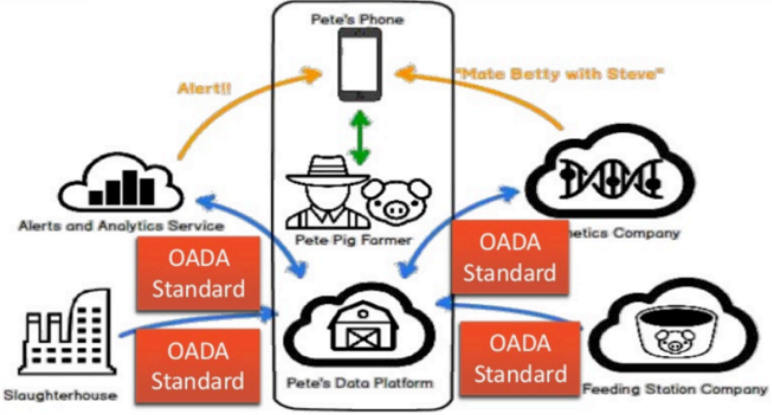
Goal 1: Creation of open source **community** anchors capable of providing the networking and support needed to encourage community development and democratized innovation in food production.

Goal 2: Design open **educational experiences and materials** that promote awareness while attracting and training an expanded pool of talent to participate in open source collaboration.

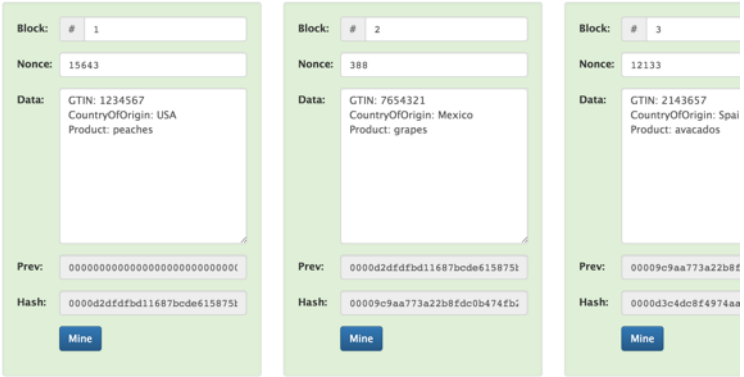
Goal 3: Development of useful **open source software/hardware** and toolkits through research applications that demonstrate how the open source paradigm can be used to innovate and generate building blocks for collaborative innovation and sharing between research and industrial practice.

Open Source Community Anchors

- **OATS Events**
  - Tech Meeting @ Stewart Center WL in March 2018
  - OATSCON @ Rosemont Feb 2019
- **AgGateway**
  - Mid-year meeting @ Altoona, IA in June 2018
  - Annual meeting @ Austin, TX in November 2018
  - Mid-year meeting @ Altoona, IA in June 2019
  - Annual meeting here right now!



Trellis
www.trellisframework.org



Open Source Community Anchors (cont'd.)

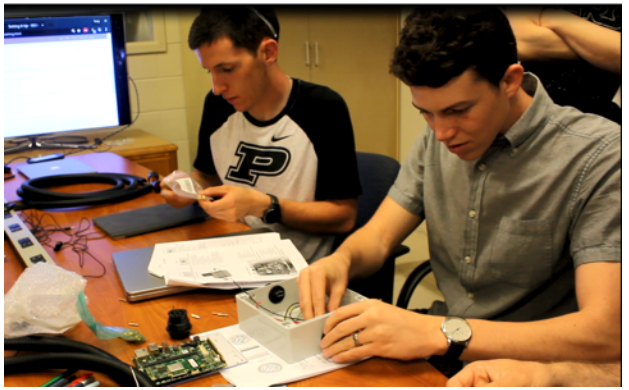
- **Trellis Framework**
  - OADA for produce safety audit data.
  - IBM Food Trust and Walmart Blockchain Pilot; Produce Supply Org., GlobalGAP, PMA, etc.
  - Trellis Governance Committee: 15 companies.

Trellis
www.trellisframework.org

MIDWEST BIG DATA HUB

- **And More …**
  - Midwest Big Data Hub meetings in Lincoln, NE and Cleveland
  - ASABE international meeting, International Conf. Prec. Ag in Montreal, Ag-Rural Broadband gathering in WL, AEM "Thinking Forward" event at Purdue.
  - Gathering for Open Agricultural Technology (GOAT)

Design Open Educational Experiences and Materials



- IsoBlue 2.0 Tutorial Videos and Build Party

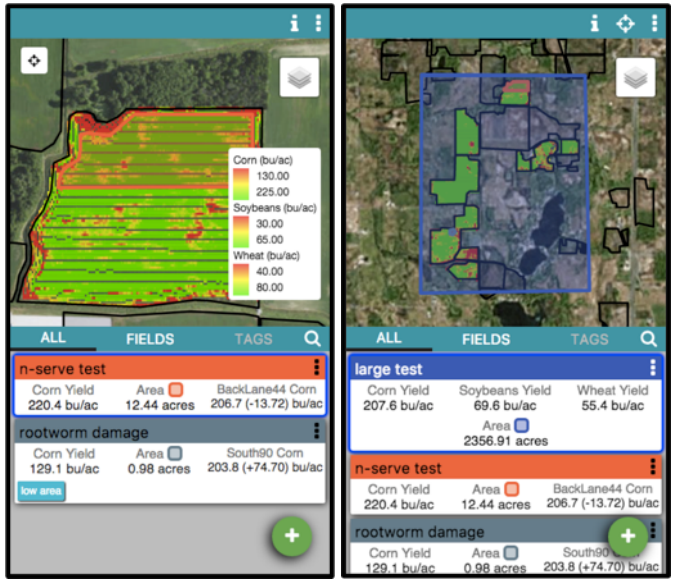
- USDA-NIFA grant: “Experiential Learning with Data Tools for Digital Agriculture and FACT” (PI: D. R. Buckmaster)

- Apps and Apps night event for Purdue students in an Ag Tech and Innovation Learning Community
- Educational sessions developed for AgGateway members

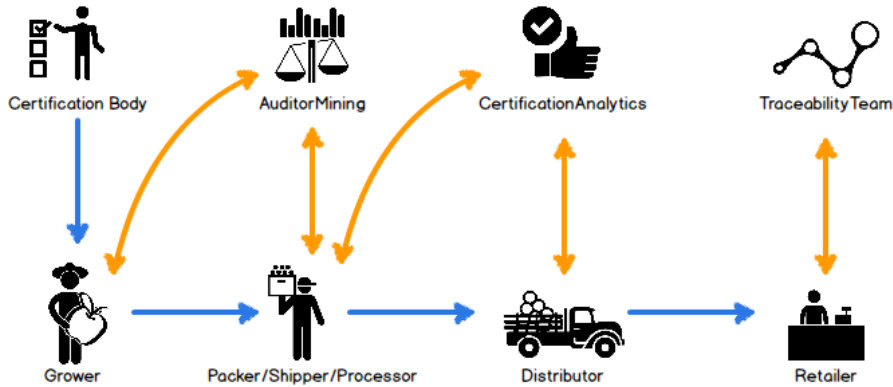
Development of Useful Open Source Software / Hardware and Toolkits

IsoBlue 2.0

Trials Tracker



Trellis



Other 2019 News

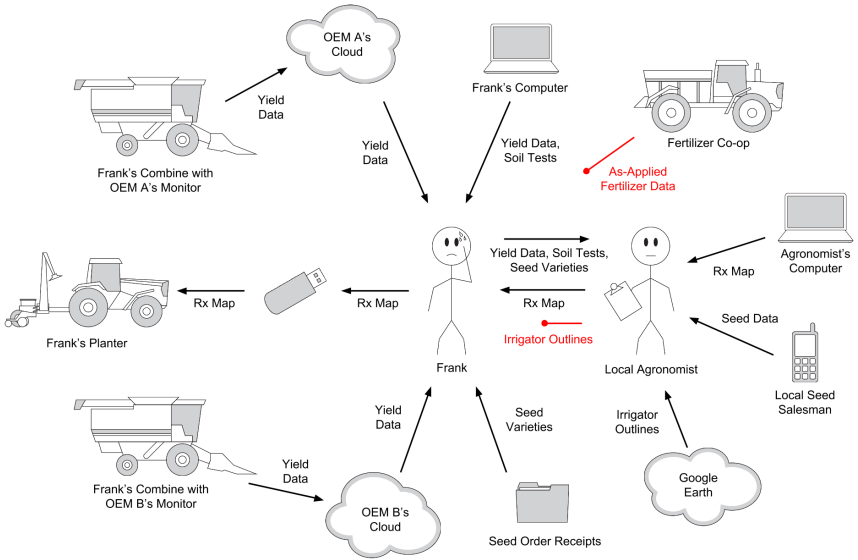
- **~New Members:** Arva Intelligence, Solinftec, The Climate Corporation
- **NSF PAWR:** Purdue/OATS partner in NC State led AERPAW (Aerial Experimentation and Research Platform for Advanced Wireless)
- **NSF ERC:** Purdue/OATS partner in U. Penn led Center for Internet of Things for Precision Agriculture (IoT4Ag)
- **IEDC:** Indiana 5G Wireless & Resilient Networks (WHIN + TOWER) Testbed

Sampling of OATS Research

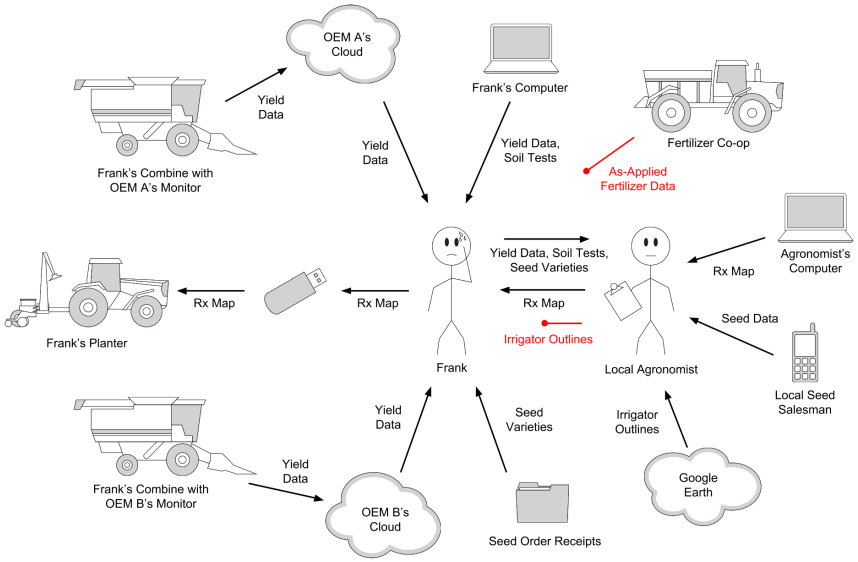
- **Cloud Computing Architectures**
- **Machine Learning**
- **Sensing Systems:** video & audio as sensors, mobile devices as sensors, hyperspectral imaging
- **Communications and Networking**
- **APIs for Sharing Data: Trellis / OADA**
- **Autonomous Labeling of Machine Sensor Data**

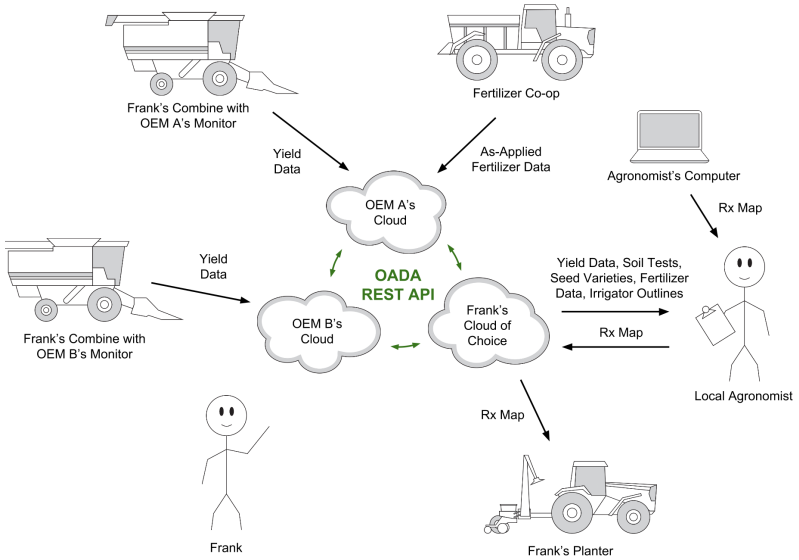
OATS Work is Animated By …

OATS Work is Animated By …

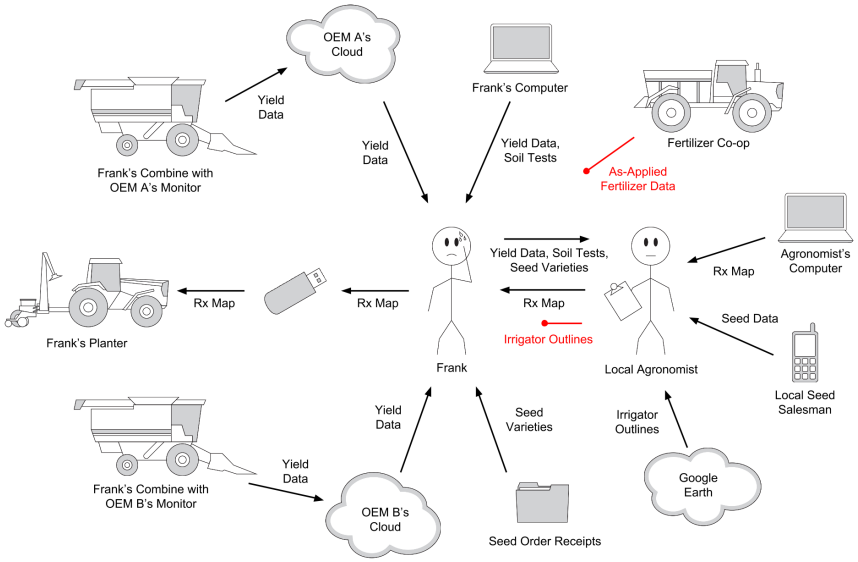


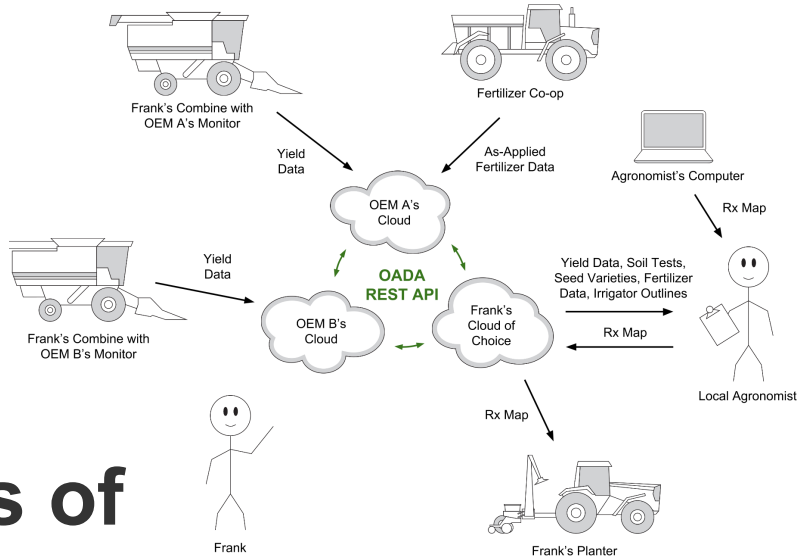
OATS Work is Animated By …





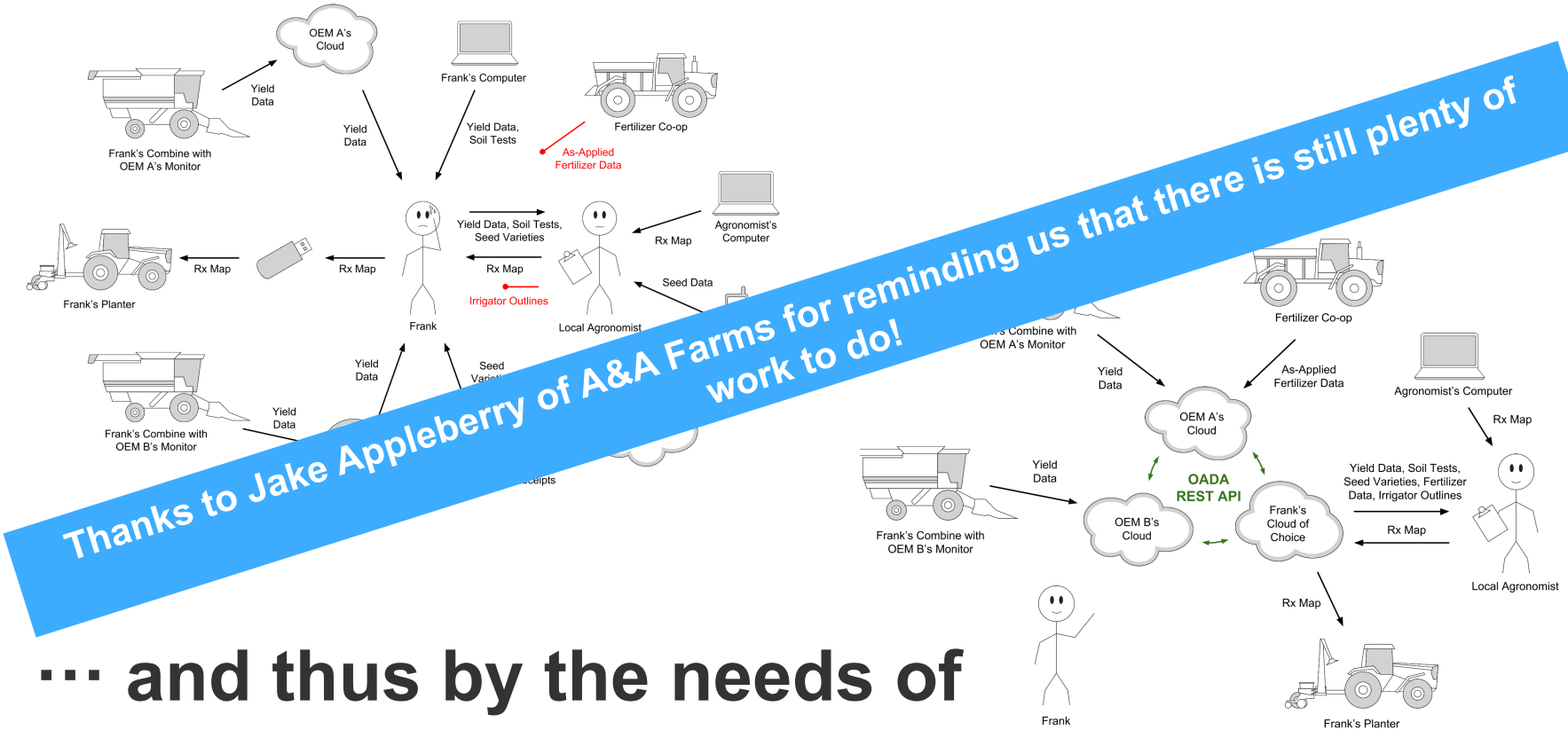
OATS Work is Animated By …





… and thus by the needs of farmers.

OATS Work is Animated By …

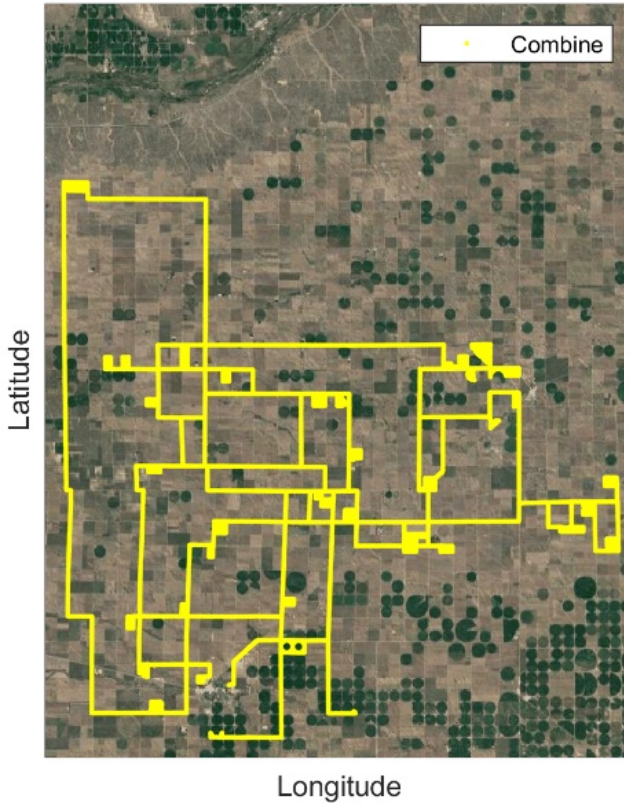


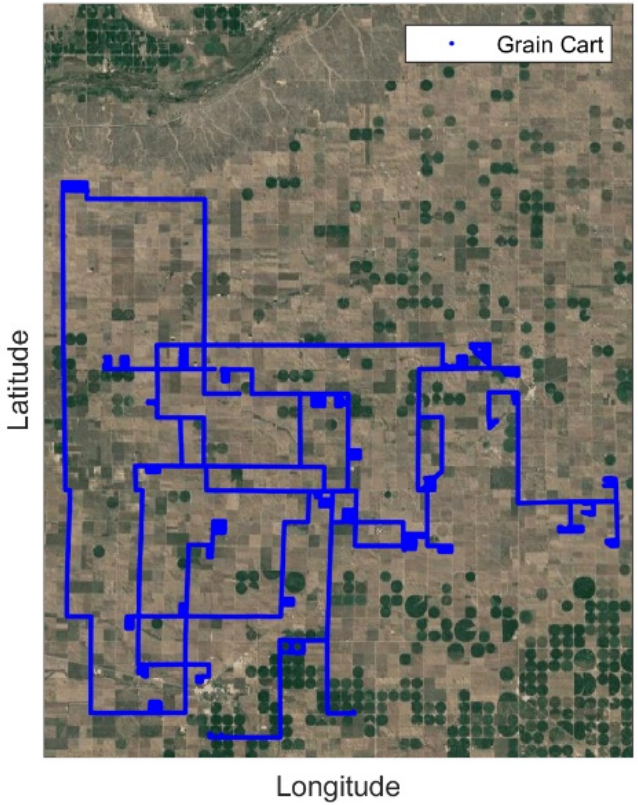
Thanks to Jake Appleberry of A&A Farms for reminding us that there is still plenty of work to do!

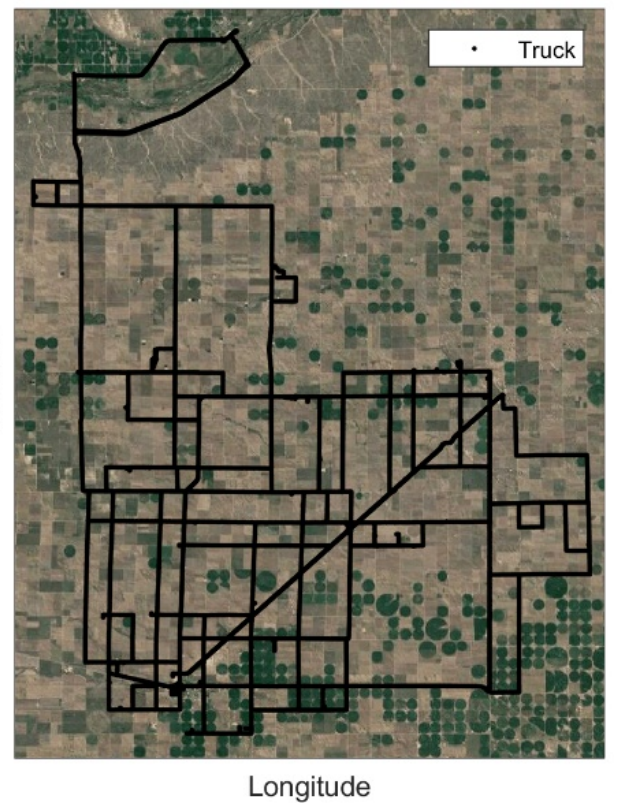
… and thus by the needs of farmers.

Therefore, in our work ⋯

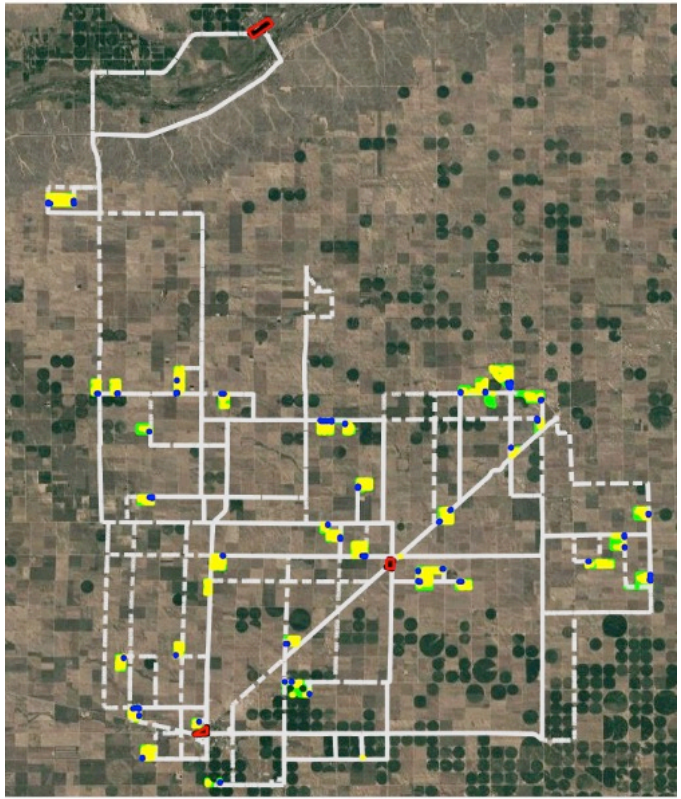
⋯ you will see actual farm data. Tracks.

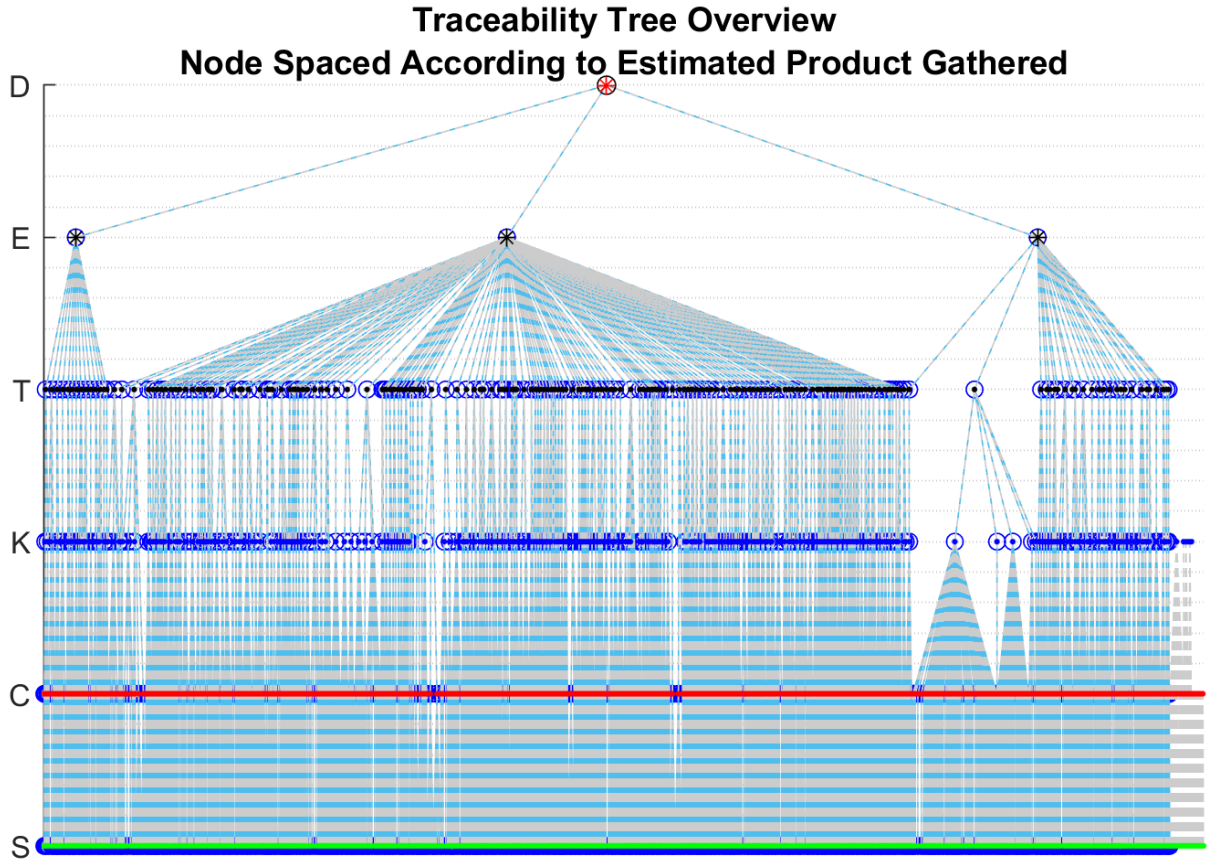




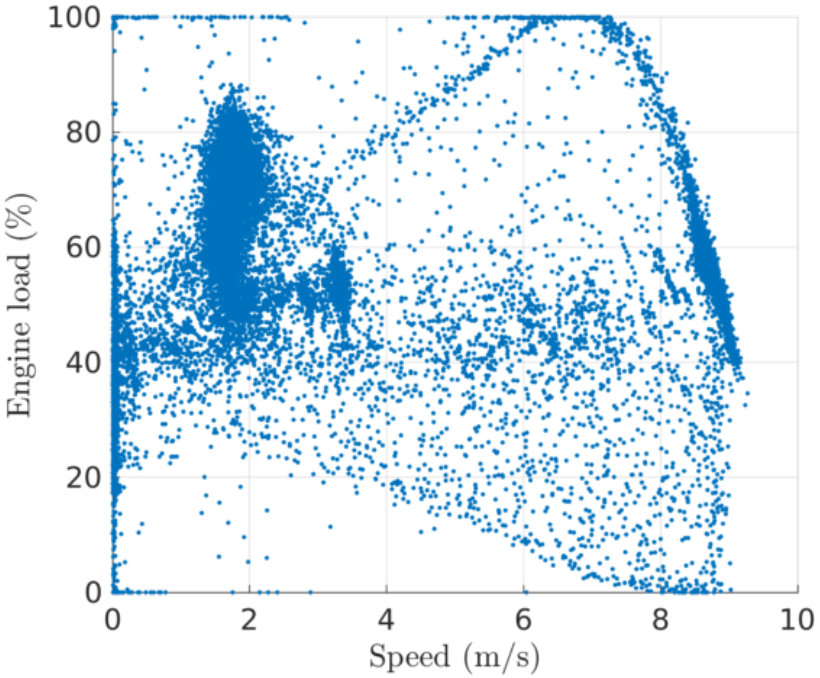


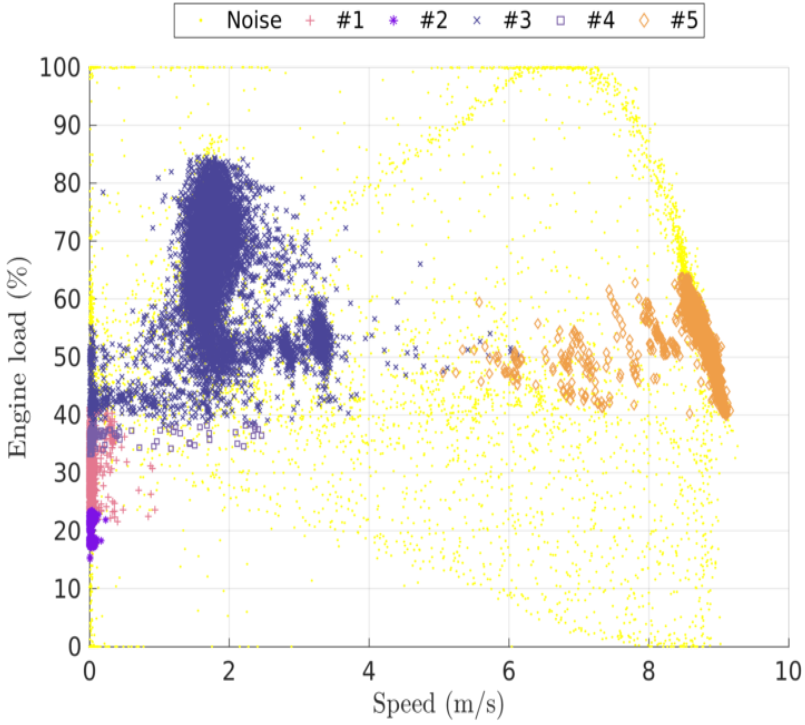
Grain Traceability.



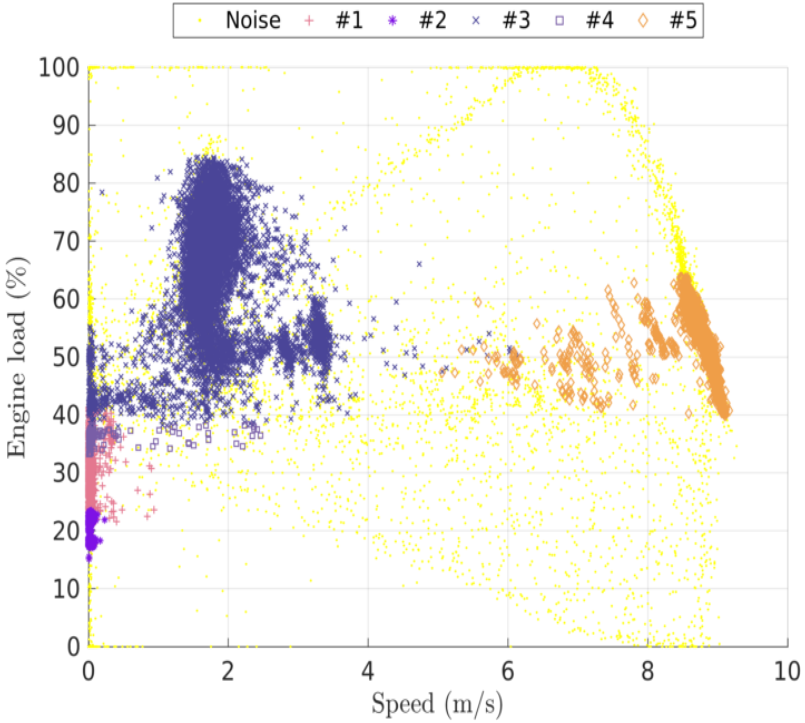


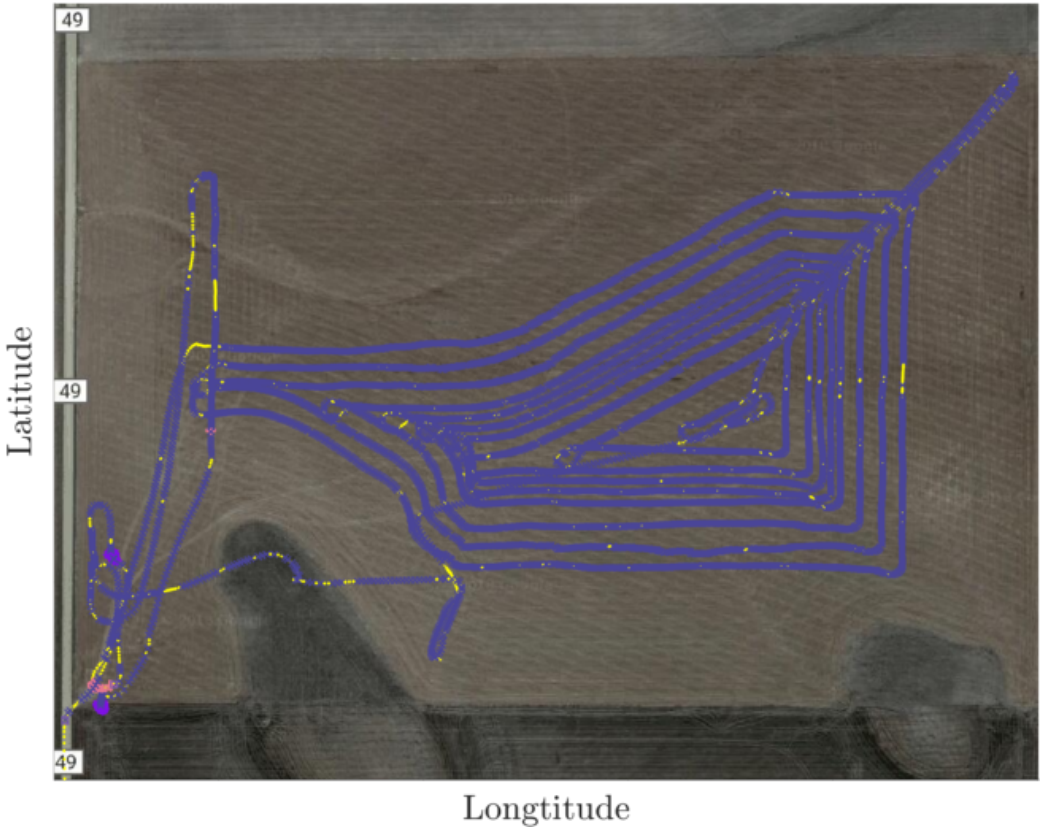
Machine Data. Analysis. Clustering.





Visualization. Anomaly Detection.





… and more …

OATS Supporters: Thank you!

Strategic Partners:

Dapicon, Inc.
Arva Intelligence
Solinftec

Supporters:

Other Support:

United States Department of Agriculture
National Institute of Food and Agriculture

OATS Supporters: Thank you!

Strategic Partners:

Dapicon, Inc.
Arva Intelligence
Solinftec

Supporters:

Other Support:

United States Department of Agriculture
National Institute of Food and Agriculture

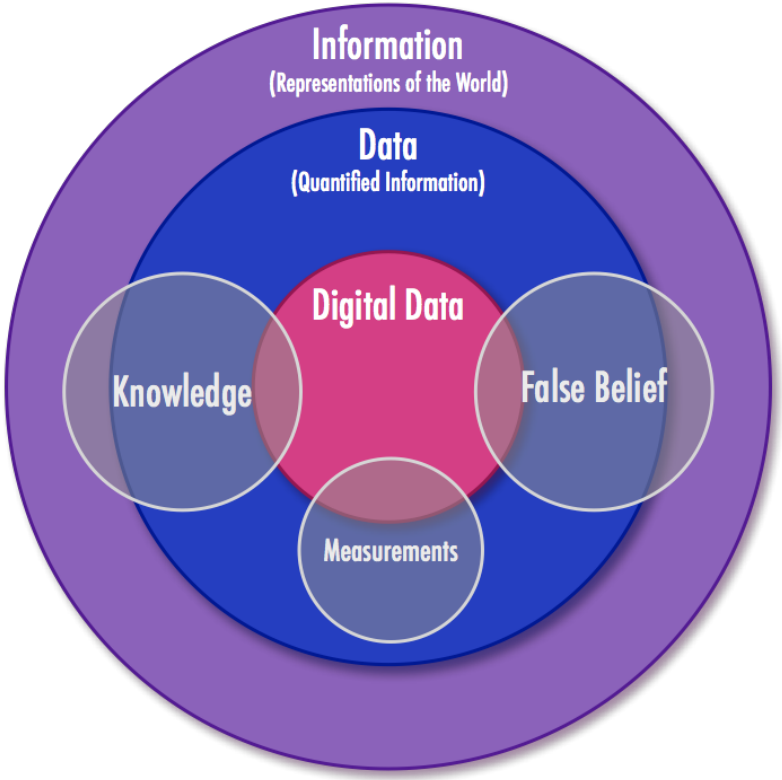
The Plan

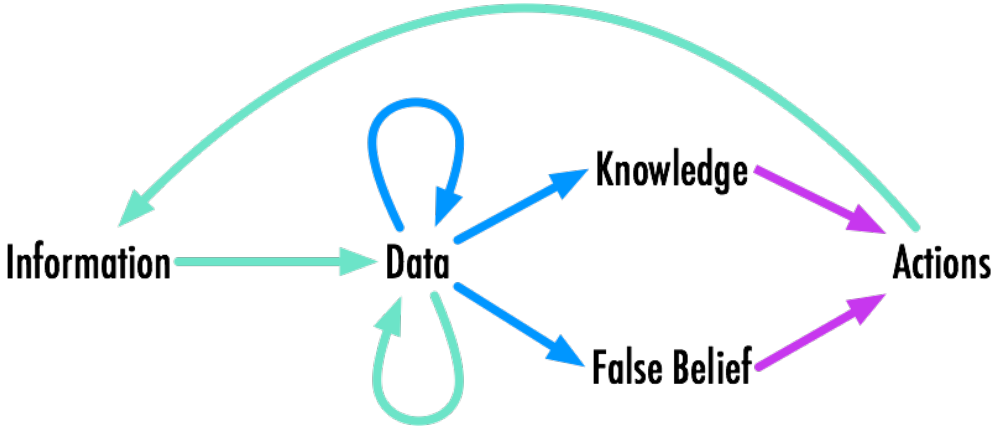
- Purdue curricula @ AgGateway annual conference
  - Two talks by JVK on data analytics via Python
  - Andrew Balmos on networking research for Ag IoT
  - Aaron Ault on blockchain and smart contracts
- JVK Day 1:
  - Background on OATS and OATS people
  - Overview of data science as a field of study
  - Python basics with examples
- JVK Day 2:
  - Continue python examples using OATS data sets
  - Survey of value

Data Science as a Field of Study (very fast and thanks to Profs. Milind Kulkarni and Chris Brinton)

what is data?

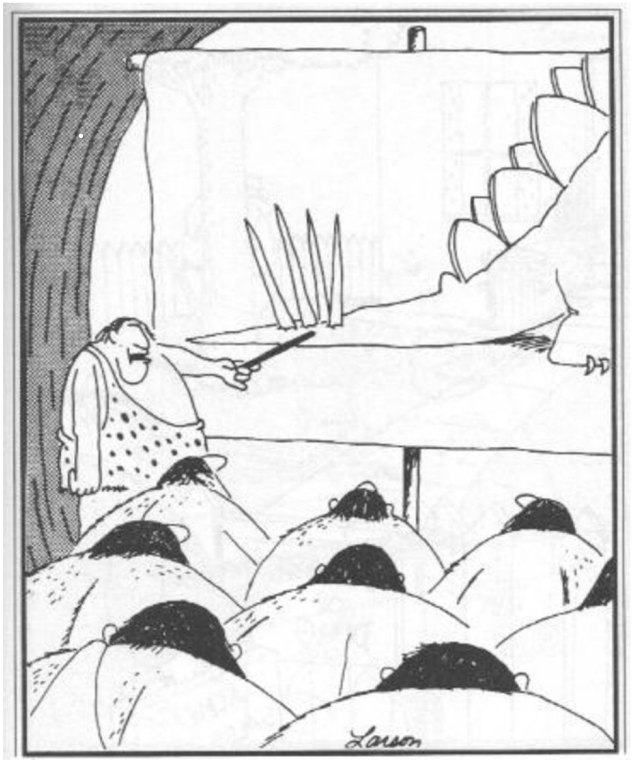
lots of different definitions





humans have used data forever

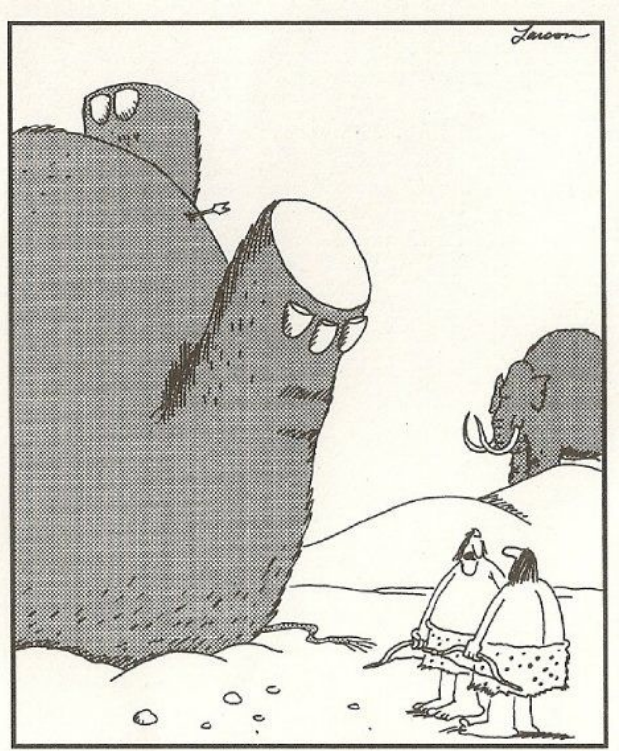
- Ever since Thag Simmons first thought, “Last time, we only sent two people to hunt the smilodon. Maybe this time we should send three?”



“Now this end is called the thagomizer . . . after the late Thag Simmons.”

why do we use data?

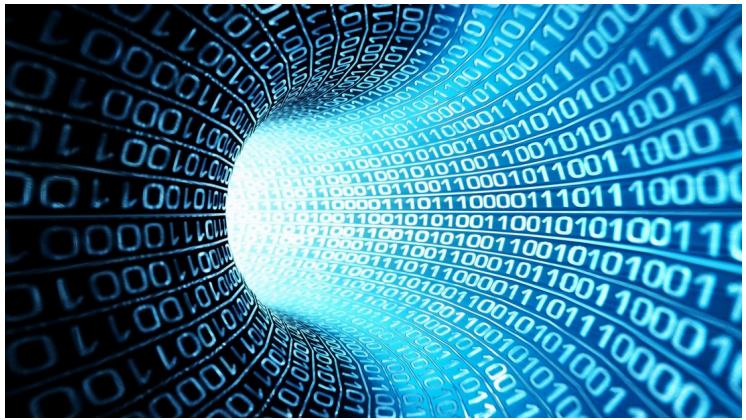
- Analyzing data helps us make decisions and take actions



"We should write that spot down."

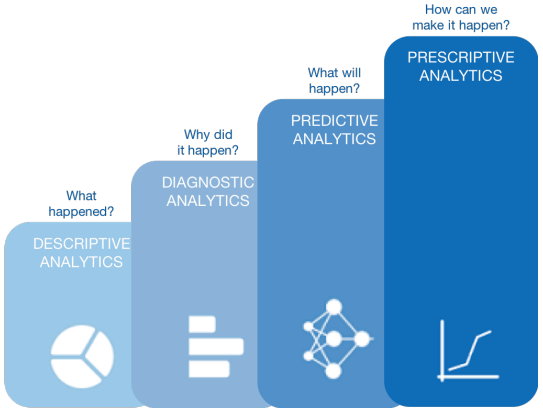
what has changed?

- There's a lot more data
- Machines can also collect (and in turn use) it
- And we're trying to do more with it



what is data science?

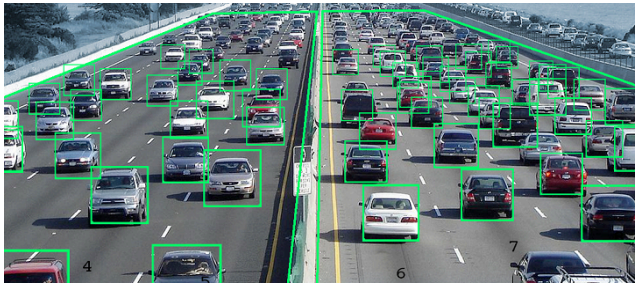
- Collecting data from a wide variety of sources and putting them into a consistent format?
- Making observations about patterns in data?
- Visualizing trends in data?
- Identifying similarities between data points?
- Making predictions about what will happen in the future?
- Prescribing courses of action to take based on forecasts?
- Developing new machine learning and data mining algorithms?
- Accelerating analysis algorithms?



what industries has it impacted?

- Hard to think of one that is *not* being positively impacted by data science!

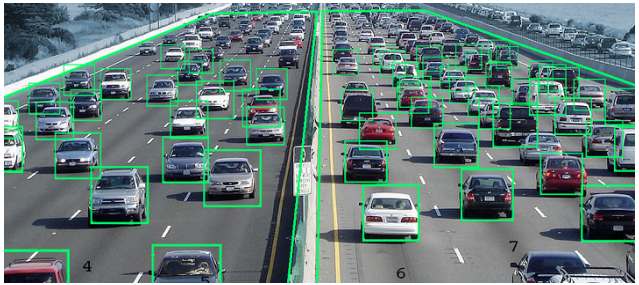
- Medicine: Analytics from wearable trackers, studying disease patterns, …
- Retail: Analyzing consumer behavior, predicting customer satisfaction, …
- Transportation: Mapping customer journeys, predicting equipment failures, …
- Education: Tracking student engagement, personalizing learning content, …



what industries has it impacted?

- Hard to think of one that is *not* being positively impacted by data science!

- Medicine: Analytics from wearable trackers, studying disease
- Retail: Analyzing behavior, predicting satisfaction, …
- … Agriculture too!!
- Education: Tracking student engagement, personalizing learning content, …



what about python?

- General purpose programming language, first appeared in the 90s
- Easily recognized by use of whitespace indentation rather than { } brackets to enhance readability
- Becoming the industry standard for data science (competing with R)
- Many useful, open-source libraries: numpy, pandas, matplotlib
- And standard control functions (e.g., loops) from lower-level languages to help structure programs

```
59  # Build the TensorFlow graph.
60  g = tf.Graph()
61  with g.as_default():
62    # Build the model.
63    model = show_and_tell_model.ShowAndTellModel(
64        model_config, mode="train", train_inception=FLAGS.train_inception)
65    model.build()
66
67    # Set up the learning rate.
68    learning_rate_decay_fn = None
69    if FLAGS.train_inception:
70      learning_rate = tf.constant(training_config.train_inception_learning_rate)
71    else:
72      learning_rate = tf.constant(training_config.initial_learning_rate)
73      if training_config.learning_rate_decay_factor > 0:
74        num_batches_per_epoch = (training_config.num_examples_per_epoch /
75                                 model_config.batch_size)
76        decay_steps = int(num_batches_per_epoch *
77                          training_config.num_epochs_per_decay)
```

some data analysis examples

data analysis in "practice"

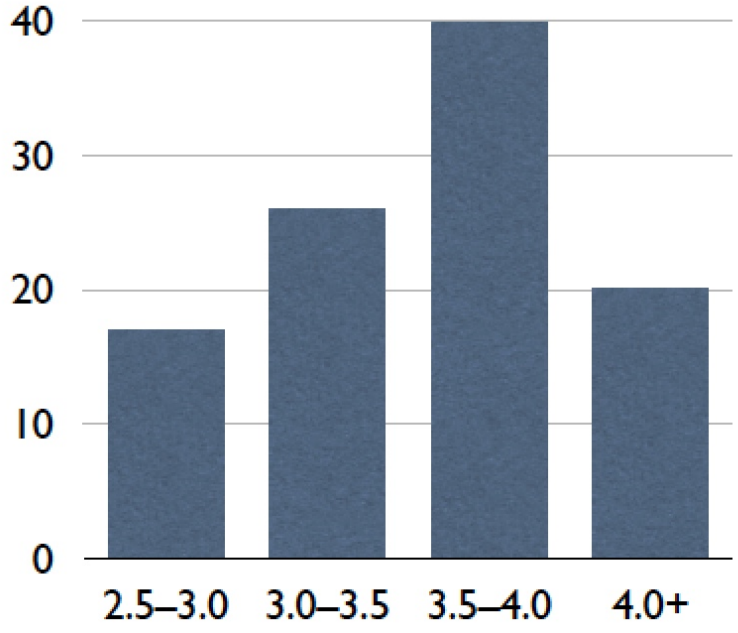
- Lets say we have a data set of applicants to Purdue

Name	High school GPA	SAT Math	SAT R/W	Residence
Jane Doe	4.7	760	700	Indiana
Purdue Pete	3.5	680	620	Indiana
B. O. Iler	3.0	800	650	Michigan
Engy Neer	4.2	750	590	North Carolina
Mark Faller	3.8	780	550	New Jersey
…	…	…	…	…

- What might we want to learn about them?

descriptive statistics

- Which students come from which states?
- What is the distribution of GPAs? SAT scores?
  - GPAs may need to be *normalized* to a consistent range across all schools
- Can build *histograms*, e.g., for the GPAs
  - But how do we know how big to make the buckets?



reasoning about data

- How do Purdue applicants compare to the national average?
  - *Mean* GPA of applicants: 3.6
- Is this high or low?
  - Can *sample* GPA of all high school students
- Suppose we collect 1000 GPAs and find a mean of 3.4
  - Does this mean Purdue students have a higher GPA on average?
- Need more information! In particular …
  - Was the sampling method we used *unbiased*?
  - What is the *variance* of the sample collected (i.e., the spread of GPAs)?
  - What *confidence interval* can be built for the population mean (i.e., what is the likely range of the true mean GPA)?

making predictions

- Can we predict how successful a particular applicant might be at Purdue?
  - How do we define success? GPA?
- Idea: Look at the application statistics of the *current seniors* and see if there is a relationship between these statistics and their current GPA
- One way to find a relationship is using *linear regression*
  - Might tell you something like: "a Purdue student's GPA can be predicted mostly by their high school GPA, with their SAT score having a lighter influence"
- Many other prediction algorithms exist too

Linear Regression: Single Variable

$$\hat{y} = \beta_0 + \beta_1 x + \epsilon$$

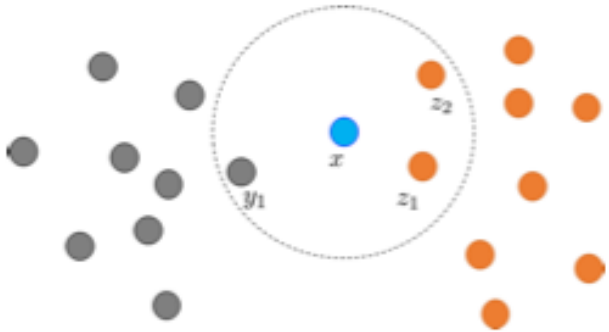
Predicted output · Coefficients · Input · Error

Linear Regression: Multiple Variables

$$\hat{y} = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p + \epsilon$$

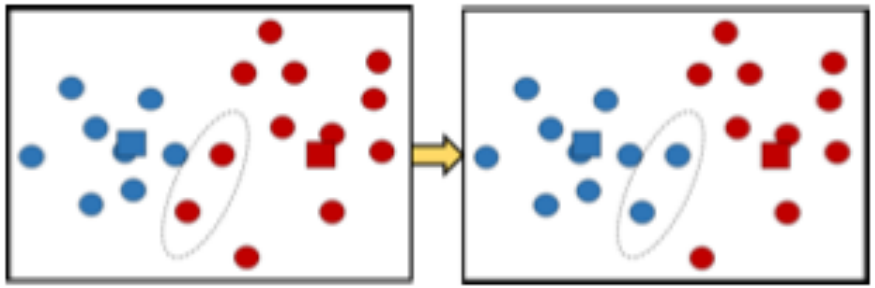
classification

- Can we make admissions decisions quicker through automation?
- Idea: Compare each applicant's statistics to past applicants that were admitted, and to those that were rejected
- Train a *classifier* to analyze these past applicants and maximize the ability to predict whether a student would be accepted or not
  - For example, a *k-nearest neighbor* classifier would assess whether a given applicant is more similar to the pool of admitted applicants or to the rejected applicants
  - Why might we run into trouble here?



clustering

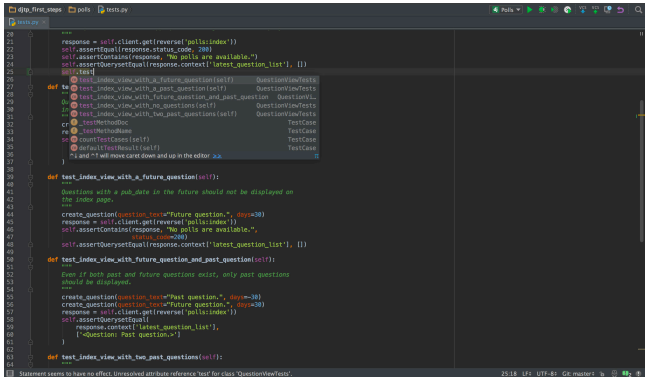
- What if we want to identify groups of students beyond "admitted" vs. "rejected"?
- Idea: See if students cluster together according to some measure of *distance*
  - Some students look more like "nearby" students than students that are "far away"
- Important question: What *features* of students should be considered for the clustering?
  - E.g., maybe don't consider something like hair color!
- With *k-means clustering*, *k* groups of students would be extracted based on "closeness"

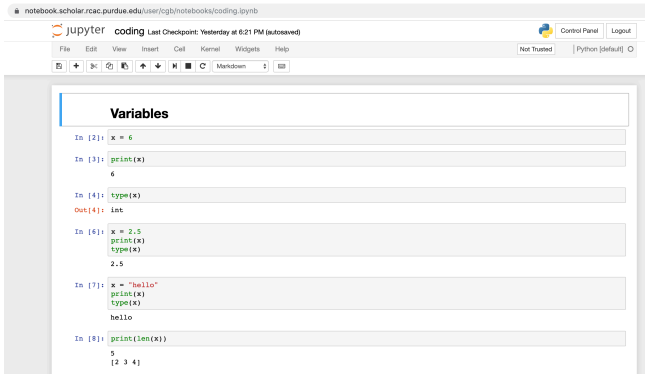


Python

coding in python

- Standard Integrated Development Environments (IDEs)
  - IDLE: Python's own, basic IDE
  - PyCharm: Code completion, unit tests, integration with git, many advanced development features
  - Many more!
- Jupyter Notebook (https://jupyter.org/)
  - Contains both computer code and rich text elements (paragraphs, figures, …)
  - Supports several dozen programming languages
  - Very useful for data science development!





basic variables

- No “declaration” command as in other programming languages
  - Variable is created when a value is assigned to it
  - Can change type after they have been set
- Few rules on naming: Can make them very descriptive!
  - Must start with a letter or underscore
  - Case-sensitive (purdue & Purdue are different)
- Combinations (+) work on all types

```
“xyz ” + “abc” = “xyz abc”
3.2 + 1 = 4.2
```

control statements

- Logical conditions

```
a == b, a != b, a < b,
a <= b, a > b, a >= b
```

- If, elif, else

```
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
else:
  print("a is greater than b”)
```

- while loop: Execute while condition is true

```
i = 1
while i < 6:
  print(i)
  i += 1
```

- for loop: Iterate over a sequence

```
for x in "banana":
  print(x)
```

- break: Stop a loop where it is and exit
- continue: Move to next iteration of loop

lists

- One of the four collection data types
  - Also tuples, sets, and dictionaries
- Lists are ordered, changeable, and allow duplicate members

```
thislist = ["apple", "banana", “apple”, “cherry”]
```

- Can pass in an integer index, or a range of indexes

```
thislist[0] = “apple"
thislist[-1] = “cherry”
thislist[1:3] = [“banana”, “apple”]
```

- Length using len() method

```
print(len(thislist))
```

- Adding items to a list

```
thislist.append(“orange”)
thislist.insert(1, “orange”)
```

- Removing items from a list

```
thislist.remove(“banana”)
thislist.pop(1)
```

- Defining lists with shorthand

```
new_list = 5 * [0]
new_list = range(5)
```

lists in for loops

- In other programming languages, for loop variables are integers
- In Python, can use any 'iterable' object

```
fruits = ["apple", "banana", "cherry"]

for x in fruits:

  if x == "banana":

    continue

  print(x)
```

- Nested loops can be used too

```
adj = ["red", "big", "tasty"]

fruits = ["apple", "banana", "cherry"]

for x in adj:

  for y in fruits:

    print(x, y)
```

- Can also iterate through a list of lists

```
data_list = [[1,2],[2,6],[5,7]]

for point in data_list:

    [x,y] = point

    z = x ** 2

    print(x,y,z)
```

- Can use the range function to iterate through integers

```
for x in range(2, 30, 3):

  print(x)
```

- Can use a list to index another list

```
ind = [1, 3, 5, 7]

values = [0] * 8

for i in ind:

  values[i] = i / 2
```

functions

- Block of code which runs when called
- Defined using def keyword

```
def my_function():
  print("Hello from a function”)
```

- Call a function using its name

```
my_function()
```

- Parameters can be passed as input to functions

```
def my_function(country):
  print("I am from " + country)
```

- To return a value, use the return statement

```
def my_function(x):
  return 5 * x


print(my_function(3))
print(my_function(5))
```

- For multiple arguments, can use keywords to specify order

```
def arithmetic(x,y,z):
  return (x+y)/z


print(arithmetic(z=3,x=2,y=4))
```

tuples

- Another of the four collection data types
- Tuples are ordered, **un**changeable, and allow duplicate members

```
thistuple = (“apple", "banana", “apple”, “cherry”)
```

- Indexed the same way as lists

```
thistuple[0] = “apple"
thistuple[-1] = “cherry”
thistuple[1:3] = (“banana”, “apple”)
```

- Once a tuple is created, items cannot be added or changed
  - Workaround: Change to list, back to tuple
- Check if item exists

```
if "apple" in thistuple:
  print("Yes, 'apple' is in the fruits tuple")
```

- Tuple with one item needs comma

```
thistuple = (“apple",) #Tuple
thistuple = (“apple") #Not a tuple
```

- Built in functions

```
thistuple.count(“apple")
thistuple.index(“apple")
```

sets

- Collection which is **un**ordered, (half) changeable, and does **not** allow duplicates
- Written with curly brackets

```
thisset = {“apple”, "banana", “cherry”}
```

- Cannot access items by index, but can loop through and check for items

```
for x in thisset:
  print(x)

print("banana" in thisset)
```

- Cannot change existing items, but can add and remove items

```
thisset.add(“orange")

thisset.update(["orange", "mango", “grapes"])

thisset.remove("banana")
```

- Also have set operations just like mathematical objects

```
set1 = {"a", "b", "c"}

set2 = {1, "b", 3}


set1.union(set2)  #Union

set1.intersection(set2)  #Intersection

set1.difference(set2)  #set1 \ set2

set1.issubset(set2)  #Testing if subset
```

dictionaries

- Collection which is **un**ordered, changeable, and indexed
- Also written with curly brackets, but have keys and values

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
```

- Access/change/add values of items by referring to the key name

```
thisdict[“model"]
thisdict[“year"] = 2019
thisdict[“color”] = "red"
```

- Can iterate through the keys, values, or both

```
for x in thisdict:
  print(thisdict[x])

for x in thisdict.values():
  print(x)

for x, y in thisdict.items():
  print(x, y)
```

- Like other collections, can create a dictionary of dictionaries

```
child1 = {"name" : “Emil", "year" : 2004 }
child2 = {"name" : “Tobias", "year" : 2007}
child3 = {"name" : “Linus", "year" : 2011}

myfamily = {“child1" : child1, "child2" : child2, "child3" : child3}
```

- Use the copy method (not direct assignment) to make a copy of a dictionary

```
mydict = dict(thisdict)
```

Now to Examples Using Jupyter