BASIC CONCEPTS AND TOOLS OF DATA SCIENCE

Prof. Mohammad Hajiaghayi & Arefeh Nasri
Wiki & Linkedin: @Mohammad Hajiaghayi
Twitter:@MTHajiaghayi
YouTube:@hajiaghayi [PLEASE SUBSCRIBE]
Instagram:@mhajiaghayi

Lectures #3 and #4

DATA/MSML602: Principles of Data Science
TuTh6:00pm – 8:30pm

TODAY'S CLASS

- Introduction to Probability Theory
- Linear Programming
- Introduction to Graphs
- Fundamental Tools of Data Science

Introduction to Probability Theory*

- Basics of probability theory
- Bayes' rule
- Random variable and distributions: Expectation and Variance

*: Some slides are adopted from slides by Rong Jin

Definition of Probability

- ***Experiment***: toss a coin twice
- ***Sample space***: possible outcomes of an experiment
  - S = {HH, HT, TH, TT}
- ***Event***: a subset of possible outcomes
  - A={HH}, B={HT, TH}
- ***Probability of an event*** : an number assigned to an event Pr(A)
  - Axiom 1: 0<= Pr(A) <= 1
  - Axiom 2: Pr(S) = 1, Pr(Ø)= 0
  - Axiom 3: For two events A and B, Pr(A∪B)= Pr(A)+Pr(B)-Pr(A∩B)
  - Proposition 1: Pr(~A)= 1- Pr(A)
  - Proposition 2: For every sequence of disjoint events $\Pr(\bigcup_i A_i) = \sum_i \Pr(A_i)$

Joint Probability

- For events A and B, **joint probability** Pr(AB) (also shown as $\Pr(A \cap B)$) stands for the probability that both events happen.
- Example: A={HH}, B={HT, TH}, what is the joint probability Pr(AB)?

  Zero

Independence

- Two events ***A and B are independent*** in case

$$\Pr(AB) = \Pr(A)\Pr(B)$$

- A set of events $\{A_i\}$ is ***independent*** in case

$$\Pr\left(\bigcap_i A_i\right) = \prod_i \Pr(A_i)$$

Independence

- Two events ***A and B are independent*** in case

$$\Pr(AB) = \Pr(A)\Pr(B)$$

- A set of events $\{A_i\}$ is ***independent*** in case

$$\Pr\left(\bigcap_i A_i\right) = \prod_i \Pr(A_i)$$

- Example: Drug test

	Women	Men
Success	200	1800
Failure	1800	200

A = {A patient is a Woman}

B = {Drug fails}

Will event A be independent from event B ?

Pr(A)=0.5, Pr(B)=0.5, Pr(AB)=9/20

Independence

- Consider the experiment of tossing a coin twice
- Example I:
  - A = {HT, HH}, B = {HT}
  - Will event A independent from event B?
- Example II:
  - A = {HT}, B = {TH}
  - Will event A independent from event B?
- Disjoint $\neq$ Independence
- If A is independent from B, B is independent from C, will A be independent from C?

Not necessarily, say A=C

Conditioning

- If A and B are events with Pr(A) > 0, the ***conditional probability of B given A*** is

$$\Pr(B \mid A) = \frac{\Pr(AB)}{\Pr(A)}$$

Conditioning

- If A and B are events with Pr(A) > 0, the ***conditional probability of B given A*** is

$$\Pr(B \mid A) = \frac{\Pr(AB)}{\Pr(A)}$$

- Example: Drug test

	Women	Men
Success	200	1800
Failure	1800	200

A = {Patient is a Woman}

B = {Drug fails}

Pr(B|A) = ?

Pr(A|B) = ?

Conditioning

- If A and B are events with Pr(A) > 0, the ***conditional probability of B given A*** is

$$\Pr(B \mid A) = \frac{\Pr(AB)}{\Pr(A)}$$

- Example: Drug test

	Women	Men
Success	200	1800
Failure	1800	200

A = {Patient is a Woman}

B = {Drug fails}

Pr(B|A) = 18/20

Pr(A|B) = 18/20

- Given A is independent from B, what is the relationship between Pr(A|B) and Pr(A)?

Pr(A|B)= P(A)

Conditional Independence

- Event A and B are ***conditionally independent given C*** in case

  $$\Pr(AB|C)=\Pr(A|C)\Pr(B|C)$$

- A set of events $\{A_i\}$ is conditionally independent given C in case

  $$\Pr(\cap_i \; A_i|C) = \Pi_i \Pr(A_i|C)$$

Conditional Independence (cont'd)

- Example: There are three events: A, B, C
  - $\Pr(A) = \Pr(B) = \Pr(C) = 1/5$
  - $\Pr(A,C) = \Pr(B,C) = 1/25$, $\Pr(A,B) = 1/10$
  - $\Pr(A,B,C) = 1/125$
  - Whether A, B are independent? $1/5 * 1/5 \neq 1/10$
  - Whether A, B are conditionally independent given C?

    $\Pr(A|C) = (1/25)/(1/5) = 1/5$, $\Pr(B|C) = (1/25)/(1/5) = 1/5$

    $\Pr(AB|C) = (1/125)/(1/5) = 1/25 = \Pr(A|C)\Pr(B|C)$
- A and B are independent $\neq$ A and B are conditionally independent

Outline

- Basics of probability theory
- Bayes' rule
- Random variables and distributions: Expectation and Variance

Bayes' Rule

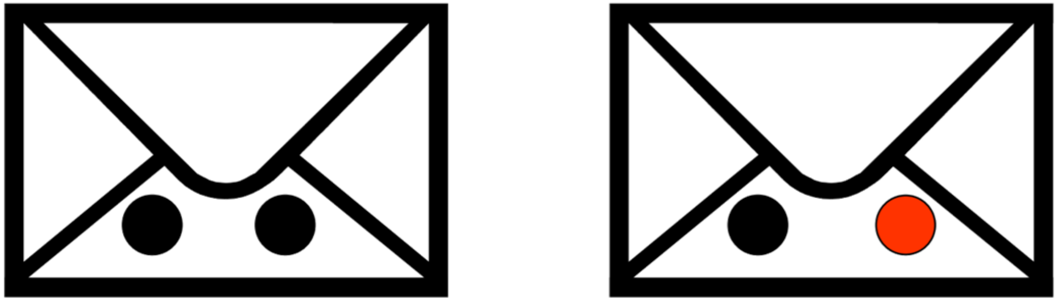
- Given two events A and B and suppose that Pr(A) > 0. Then

$$\Pr(A|B) = \frac{\Pr(AB)}{\Pr(B)} = \frac{\Pr(B|A)\Pr(A)}{\Pr(B)}$$

- Why do we make things this complicated?
  - Often P(B|A), P(A), P(B) are easier to get
  - Some names:
    - **Prior P(A)**: probability before any evidence
    - **Likelihood P(B|A)**: assuming A, how likely is the evidence
    - **Posterior P(A|B)**: conditional prob. after knowing evidence
    - **Inference**: deriving unknown probability from known ones

Inference with Bayes' Rule: Example

- In a bag there are two envelopes
  - one has a red ball (worth $100) and a black ball
  - one has two black balls. Black balls worth nothing



- You randomly grabbed an envelope, randomly took out one ball – it's black.
- At this point you're given the option to switch the envelope. To switch or not to switch?

Inference with Bayes' Rule: Example

- E: envelope, 1=(R,B), 2=(B,B)
- B: the event of drawing a black ball
- $P(E|B) = P(B|E)*P(E) / P(B)$
- We want to compare $P(E=1|B)$ vs. $P(E=2|B)$
- $P(B|E=1) = 0.5$, $P(B|E=2) = 1$
- $P(E=1)=P(E=2)=0.5$
- $P(B)=3/4$ (it in fact doesn't matter for the comparison)
- $P(E=1|B)=1/3$, $P(E=2|B)=2/3$
- After seeing a black ball, the posterior probability of this envelope being 1 (thus worth $100) is smaller than it being 2
- Thus you should switch

Bayes' Rule: More Complicated

- Suppose that B_1, B_2, … B_k form a partition of S:

$$B_i \bigcap B_j = \varnothing; \quad \bigcup_i B_i = S$$

Suppose that $Pr(B_i) > 0$ and $Pr(A) > 0$. Then

$$\Pr(B_i \mid A) = \frac{\Pr(A \mid B_i)\Pr(B_i)}{\Pr(A)}$$

Bayes' Rule: More Complicated

- Suppose that B_1, B_2, … B_k form a partition of S:

$$B_i \bigcap B_j = \varnothing; \quad \bigcup_i B_i = S$$

Suppose that $\Pr(B_i) > 0$ and $\Pr(A) > 0$. Then

$$\Pr(B_i \mid A) = \frac{\Pr(A \mid B_i)\Pr(B_i)}{\Pr(A)}$$

$$= \frac{\Pr(A \mid B_i)\Pr(B_i)}{\sum_{j=1}^{k} \Pr(AB_j)}$$

Bayes' Rule: More Complicated

- Suppose that B_1, B_2, … B_k form a partition of S:

$$B_i \bigcap B_j = \varnothing; \quad \bigcup_i B_i = S$$

Suppose that $\Pr(B_i) > 0$ and $\Pr(A) > 0$. Then

$$\Pr(B_i \mid A) = \frac{\Pr(A \mid B_i)\Pr(B_i)}{\Pr(A)}$$

$$= \frac{\Pr(A \mid B_i)\Pr(B_i)}{\sum_{j=1}^{k} \Pr(AB_j)}$$

$$= \frac{\Pr(A \mid B_i)\Pr(B_i)}{\sum_{j=1}^{k} \Pr(B_j)\Pr(A \mid B_j)}$$

Outline

- Basics of probability theory
- Bayes’ rule
- Random variable and probability distribution: Expectation and Variance

Random Variable and Distribution

- A ***random variable X*** is a numerical outcome of a random experiment
- The ***distribution*** of a random variable is the collection of possible outcomes along with their probabilities:
  - Discrete case: $\Pr(X = x) = p_\theta(x)$
  - Continuous case: $\Pr(a \le X \le b) = \int_a^b p_\theta(x)dx$
- The ***support*** of a discrete distribution is the set of all x for which $\Pr(X=x) > 0$
- The ***joint distribution*** of two random variables X and Y is the collection of possible outcomes along with the joint probability $\Pr(X=x, Y=y)$.

Random Variable: Example

- Let S be the set of all sequences of three rolls of a die. Let X be the sum of the number of dots on the three rolls.
- What are the possible values for X?
- Pr(X = 3) = 1/6*1/6*1/6=1/216,
- Pr(X = 5) = ?

Expectation

- A random variable X~Pr(X=x). Then, its expectation is

$$E[X] = \sum_x x \Pr(X = x)$$

  - In an empirical sample, $x_1, x_2, \ldots, x_N$,

$$E[X] = \frac{1}{N} \sum_{i=1}^{N} x_i$$

- Continuous case: $E[X] = \int_{-\infty}^{\infty} x p_\theta(x) dx$
- In the discrete case, expectation is indeed the average of numbers in the support weighted by their probabilities
- Expectation of sum of random variables

$$E[X_1 + X_2] = E[X_1] + E[X_2]$$

Expectation: Example

- Let S be the set of all sequence of three rolls of a die. Let X be the sum of the number of dots on the three rolls.
- Exercise: What is E(X)?

- Let S be the set of all sequence of three rolls of a die. Let X be the product of the number of dots on the three rolls.
- Exercise: What is E(X)?

Variance

- The variance of a random variable X is the expectation of $(X-E[X])^2$:

$$Var(X)=E[(X-E[X])^2]$$
$$=E[X^2+E[X]^2-2XE[X]]=$$
$$=E[X^2]+E[X]^2-2E[X]E[X]$$
$$=E[X^2]-E[X]^2$$

Bernoulli Distribution

- The outcome of an experiment can either be success (i.e., 1) and failure (i.e., 0).
- Pr(X=1) = p, Pr(X=0) = 1-p, or

$$p_\theta(x) = p^x (1-p)^{1-x}$$

- E[X] = p, Var(X) = E[X^2]-E[X]^2 =p-p^2

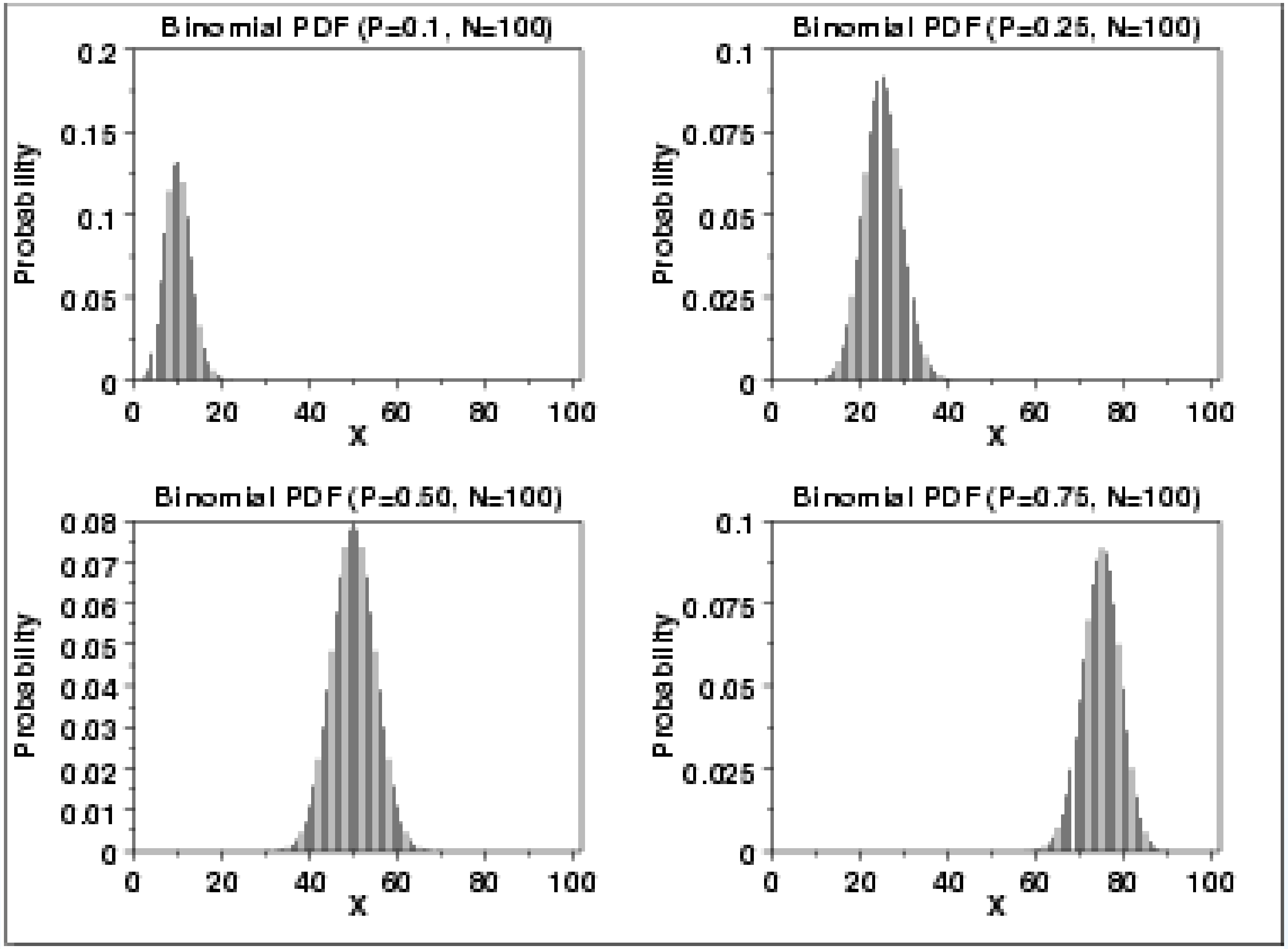
Binomial Distribution

- n draws of a Bernoulli distribution
  - X_i~Bernoulli(p), $X=\sum_{i=1}^{n} X_i$, X~Bin(p, n)
- Random variable X stands for the number of times that experiments are successful.

$$\Pr(X = x) = p_\theta(x) = \begin{cases} \binom{n}{x} p^x (1-p)^{n-x} & x = 1,2,\dots,n \\ 0 & \text{otherwise} \end{cases}$$

- E[X] = np, Var(X) = np(1-p)

Plots of Binomial Distribution



Poisson Distribution

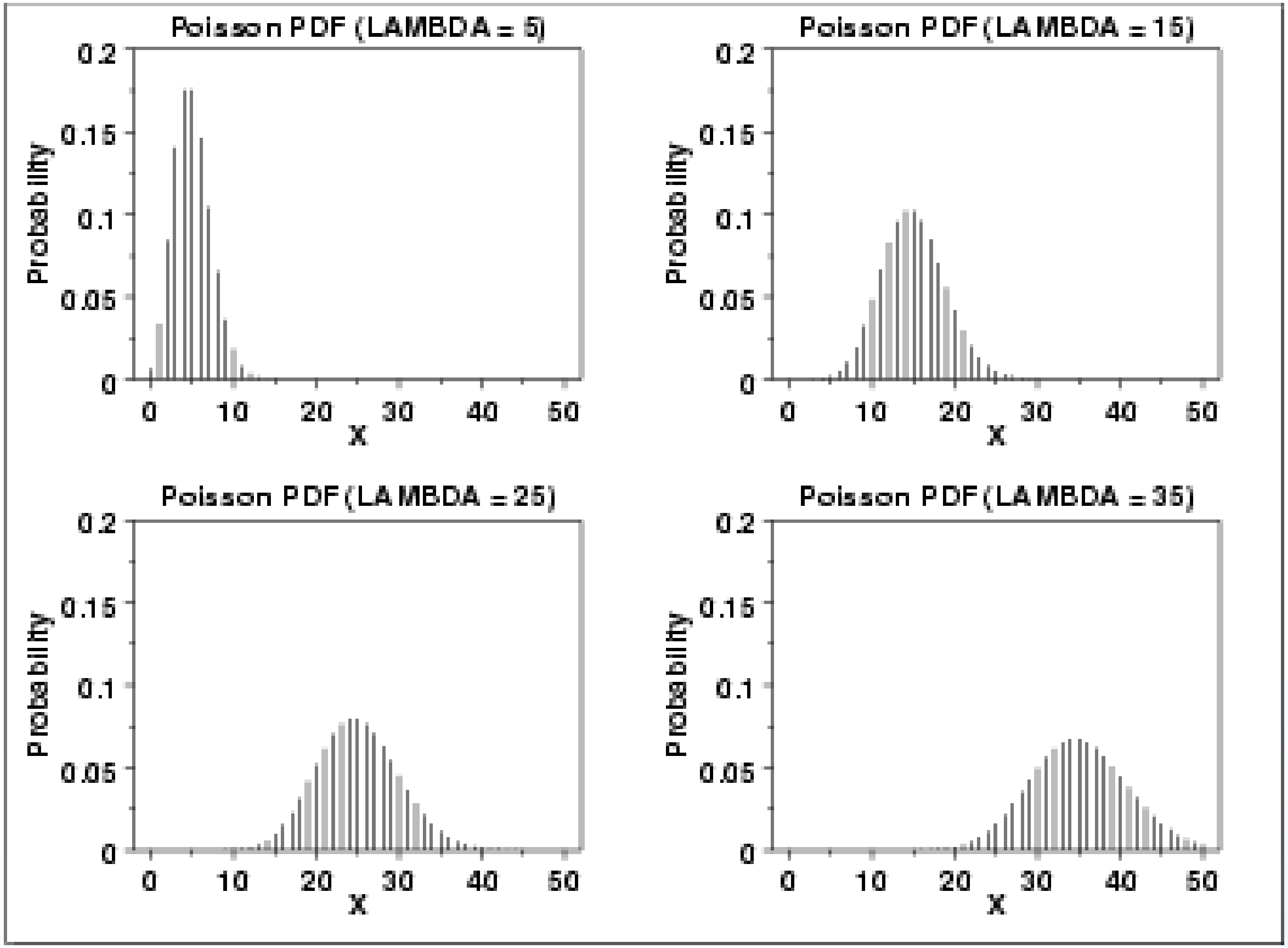
- Coming from Binomial distribution
  - Fix the expectation λ=np
  - Let the number of trials n$\rightarrow\infty$

  A Binomial distribution will become a Poisson distribution

$$\Pr(X = x) = p_{\theta}(x) = \begin{cases} \frac{\lambda^x}{x!} e^{-\lambda} & x \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

- E[X] = λ, Var(X) = λ

Plots of Poisson Distribution



Normal (Gaussian) Distribution

- $X \sim N(\mu, \sigma^2)$

$$p_\theta(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{ -\frac{(x-\mu)^2}{2\sigma^2} \right\}$$

$$\Pr(a \le X \le b) = \int_a^b p_\theta(x)dx = \int_a^b \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{ -\frac{(x-\mu)^2}{2\sigma^2} \right\} dx$$

- $E[X] = \mu$, $Var(X) = \sigma^2$
- If $X_1 \sim N(\mu_1, \sigma^2_1)$ and $X_2 \sim N(\mu_2, \sigma^2_2)$, for $X = X_1 + X_2 \sim N(\mu_1 + \mu_2, \sigma^2_1 + \sigma^2_2)$
- Note that Binomial distributions are Normal (Gaussian)

Probability in Python

colab.research.google.com/drive/1nnbCQMppiGfkTaRw-TxZWW9eq7xBtXqd#scrollTo=zinyvxMwP6EH

Untitled1.ipynb

File Edit View Insert Runtime Tools Help All changes saved

```python
import numpy as np
import numpy.random as rm
import scipy.stats as st
ar1=np.full(4,3)
print(ar1)

X=1000*rm.randn(20)+10000 #1000 is the standard deviation of normal distribution, 500 is the number of samples points and 10000 is the mean
Y= rm.randint(0,2,size=20)

d1=X
d2=Y
cor,_=st.pearsonr(d1,d2)
cor1,_=st.spearmanr(d1,d2)
print(d1)
print(d2)
print(cor1)
cor=st.pearsonr(d1,d2)
print(cor)
```

```
[3 3 3 3]
[10441.2789784  10353.44378245  9363.50548998 10606.05278723
  9817.14266103 10074.81506252 10801.28349835  9847.72135591
 11137.64477568 10327.55568368 10488.89720533 10185.89337601
  8083.94227971  9979.90596431 10161.12657669 10384.2755884
  8651.56661396 10636.47181246  9269.26895539  8879.84276296]
[0 1 0 1 1 1 0 0 0 1 1 0 1 0 1 0 0 1 0 0]
0.13072174640400053
(0.10398006545187807, 0.6626502989542163)
```

Linear Programming

- minimize or maximize a linear objective
- subject to linear equalities and inequalities

Example. Max is in a pie eating contest that lasts 1 hour. Each torte that he eats takes 2 minutes. Each apple pie that he eats takes 3 minutes. He receives 4 points for each torte and 5 points for each pie. What should Max eat so as to get the most points?

Step 1. Determine the decision variables

- Let x be the number of tortes eaten by Max.
- Let y be the number of pies eaten by Max.

Max's linear program

Step 2. Determine the *objective function*

Step 3. Determine the *constraints*

Maximize $z = 4x + 5y$ (objective function)

subject to $2x + 3y \leq 60$ (constraint)

$x \geq 0$; $y \geq 0$ (non-negativity constraints)

A ***feasible solution*** satisfies all of the constraints.

$x = 10$, $y = 10$ is feasible; $x = 10$, $y = 15$ is ***infeasible***.

An ***optimal solution*** is the best feasible solution.

The optimal solution is $x = 30$, $y = 0$.

Terminology

- ***Decision variables:*** e.g., x and y.
  - In general, these are quantities you can control to improve your objective which should completely describe the set of decisions to be made.
- ***Constraints:*** e.g., $2x + 3y \leq 24$, $x \geq 0$, $y \geq 0$
  - Limitations on the values of the decision variables.
- ***Objective Function.*** e.g., $4x + 5y$
  - Value measure used to rank alternatives
  - Seek to maximize or minimize this objective
  - examples: maximize NPV, minimize cost

Linear Programs

- A *linear function* is a function of the form:

$$f(x_1, x_2, \ldots, x_n) = c_1x_1 + c_2x_2 + \ldots + c_nx_n$$
$$= \sum_{i=1 \text{ to } n} c_ix_i$$

e.g., $3x_1 + 4x_2 - 3x_4$.

- A mathematical program is a *linear program (LP)* if the objective is a linear function and the constraints are linear equalities or inequalities.

e.g., $3x_1 + 4x_2 - 3x_4 \geq 7$

$x_1 - 2x_5 = 7$

- Typically, an LP has non-negativity constraints.

An integer program is a linear program plus constraints that some or all of the variables are integer valued.

- Maximize $3x_1 + 4x_2 - 3x_3$

$$3x_1 + 2x_2 - x_3 \geq 17$$

$$3x_2 - x_3 = 14$$

$x_1 \geq 0$, $x_2 \geq 0$, $x_3 \geq 0$ and

x_1, x_2, x_3 are all integers

Complexity of LP & IP

1. Simplex Method runs in exponential time.
2. Ellepsoid Method runs in $O(n^6)$ time.
3. Interior Point Method runs in $O(n^{3.5})$ time.
4. Unfortunately, Integer Programming (IP) is NP-complete.
5. **Let's see a basic Linear Programming in Python with PuLP** https://www.realpythonproject.com/basic-linear-programming-in-python-with-pulp/

Introduction to Graphs

Figure: Road Networks

Figure: Internet

Figure: Social Networks

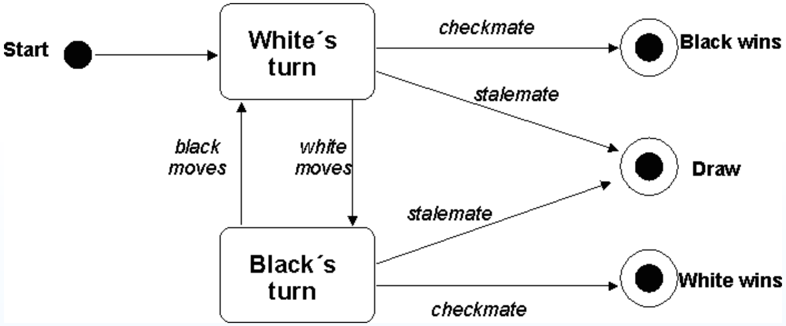
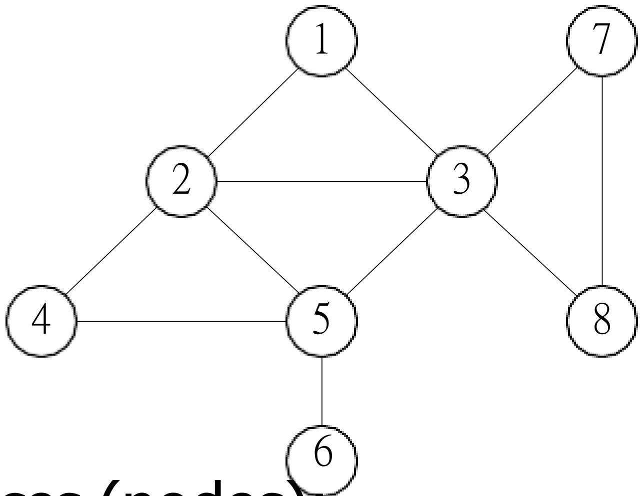


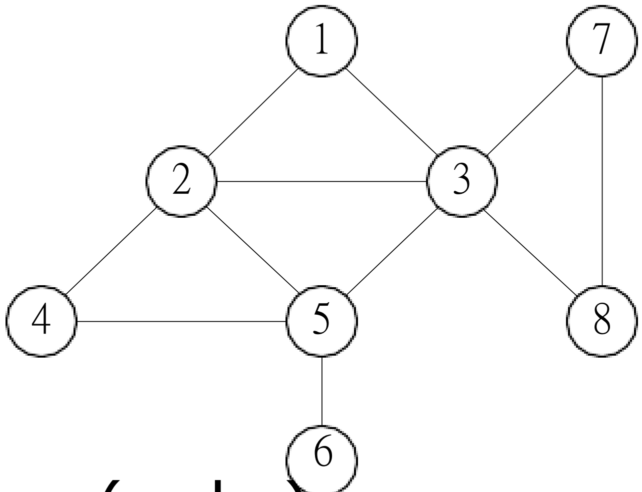
Figure: Transition Graphs

(Undirected) Graph $G = (V, E)$



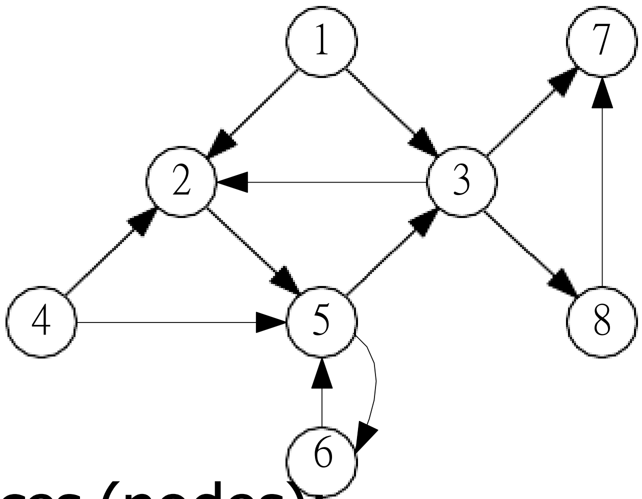
- V: set of vertices (nodes);
- E: pairwise relationships among V;
  - (undirected) graphs: relationship is symmetric, E contains subsets of size 2

(Undirected) Graph $G = (V, E)$



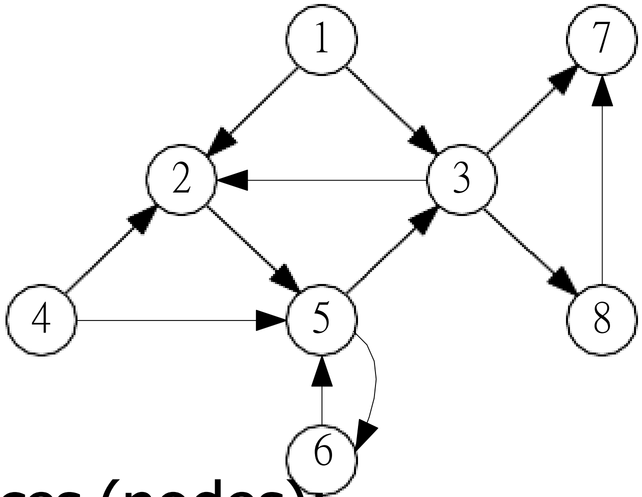
- V: set of vertices (nodes);
  - $V = \{1, 2, 3, 4, 5, 6, 7, 8\}$
- E: pairwise relationships among V;
  - (undirected) graphs: relationship is symmetric, E contains subsets of size 2
  - $E = \{\{1,2\}, \{1,3\}, \{2,3\}, \{2,4\}, \{2,5\}, \{3,5\}, \{3,7\}, \{3,8\}, \{4,5\}, \{5,6\}, \{7,8\}\}$

Directed Graph $G = (V, E)$



- V: set of vertices (nodes);
  - $V = \{1, 2, 3, 4, 5, 6, 7, 8\}$
- E: pairwise relationships among V;
  - directed graphs: relationship is asymmetric, E contains ordered pairs

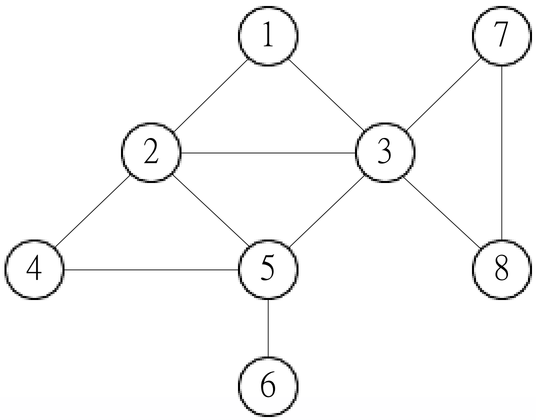
Directed Graph $G = (V, E)$



- V: set of vertices (nodes);
  - $V = \{1, 2, 3, 4, 5, 6, 7, 8\}$
- E: pairwise relationships among V;
  - directed graphs: relationship is asymmetric, E contains ordered pairs
  - $E = \{(1, 2), (1, 3), (3, 2), (4, 2), (2, 5), (5, 3), (3, 7), (3, 8),$
    $(4, 5), (5, 6), (6, 5), (8, 7)\}$

Abuse of Notations

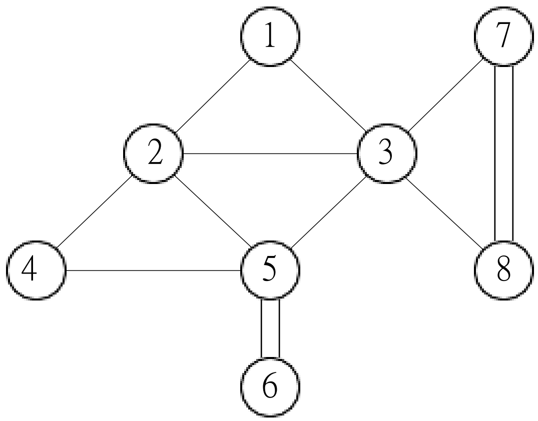
- For (undirected) graphs, we often use (i, j) to denote the set $\{i, j\}$.
- We call (i, j) an unordered pair; in this case $(i, j) = (j, i)$.



- $E = \{(1, 2), (1, 3), (2, 3), (2, 4), (2, 5), (3, 5), (3, 7), (3, 8), (4, 5), (5, 6), (7, 8)\}$

- Social Network : Undirected
- Transition Graph : Directed
- Road Network : Directed or Undirected
- Internet : Directed or Undirected

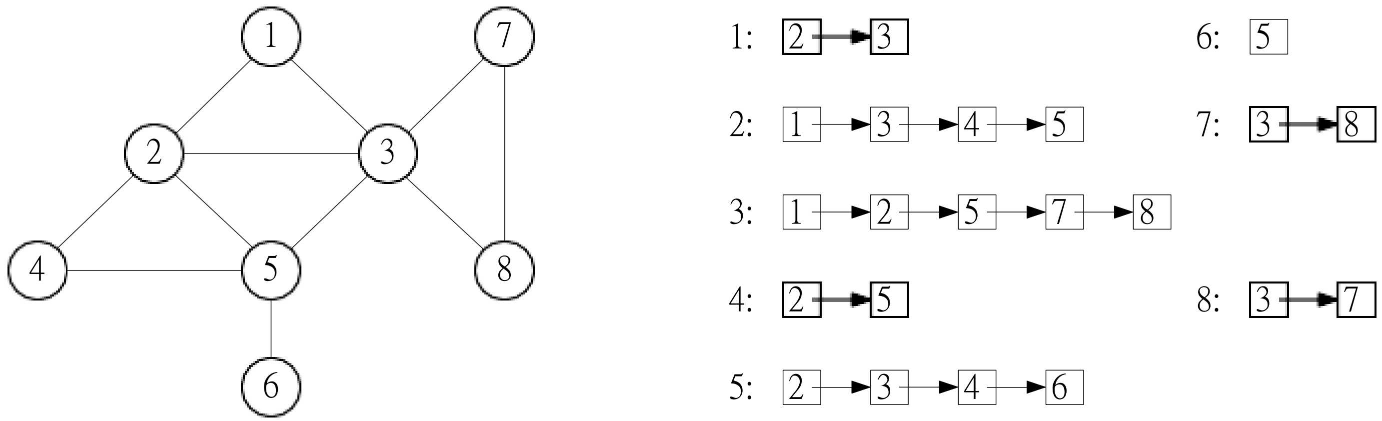
Representation of Graphs



	1	2	3	4	5	6	7	8
1	0	1	1	0	0	0	0	0
2	1	0	1	1	1	0	0	0
3	1	1	0	0	1	0	1	1
4	0	1	0	0	1	0	0	0
5	0	1	1	1	0	1	0	0
6	0	0	0	0	1	0	0	0
7	0	0	1	0	0	0	0	1
8	0	0	1	0	0	0	1	0

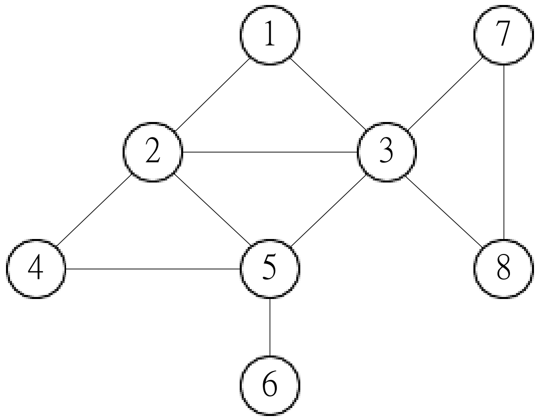
- Adjacency matrix
  - $n \times n$ matrix, $A[u,v] = 1$ if $(u,v) \in E$ and $A[u,v] = 0$ otherwise
  - A is symmetric if graph is undirected

Representation of Graphs



- Adjacency matrix
  - $n \times n$ matrix, $A[u, v] = 1$ if $(u, v) \in E$ and $A[u, v] = 0$ otherwise
  - A is symmetric if graph is undirected
- Linked lists
  - For every vertex v, there is a linked list containing all neighbours of v.

Representation of Graphs



1: 2 3
2: 1 3 4 5
3: 1 2 5 7 8
4: 2 5
5: 2 3 4 6
6: 5
7: 3 8
8: 3 7

d : (2, 4, 5, 2, 4, 1, 2, 2)

- Adjacency matrix
  - $n \times n$ matrix, $A[u, v] = 1$ if $(u, v) \in E$ and $A[u, v] = 0$ otherwise
  - A is symmetric if graph is undirected
- Linked lists
  - For every vertex v, there is a linked list containing all neighbours of v.
  - When graph is static, can use array of variant-length arrays.

Comparison of Two Representations

- Assuming we are dealing with undirected graphs
- n: number of vertices
- m: number of edges, assuming $n - 1 \le m \le n(n-1)/2$
- d_v: number of neighbors of v

	Matrix	Linked Lists
memory usage		
time to check $(u, v) \in E$		
time to list all neighbours of v		

Comparison of Two Representations

- Assuming we are dealing with undirected graphs
- n: number of vertices
- m: number of edges, assuming $n - 1 \le m \le n(n-1)/2$
- d_v: number of neighbors of v

	Matrix	Linked Lists
memory usage	$O(n^2)$	
time to check $(u, v) \in E$		
time to list all neighbours of v		

Comparison of Two Representations

- Assuming we are dealing with undirected graphs
- n: number of vertices
- m: number of edges, assuming $n - 1 \leq m \leq n(n-1)/2$
- d_v: number of neighbors of v

	Matrix	Linked Lists
memory usage	$O(n^2)$	$O(m)$
time to check $(u, v) \in E$		
time to list all neighbours of v		

Comparison of Two Representations

- Assuming we are dealing with undirected graphs
- n: number of vertices
- m: number of edges, assuming $n-1 \le m \le n(n-1)/2$
- d_v: number of neighbors of v

	Matrix	Linked Lists
memory usage	$O(n^2)$	$O(m)$
time to check $(u,v) \in E$	$O(1)$	
time to list all neighbours of v		

Comparison of Two Representations

- Assuming we are dealing with undirected graphs
- n: number of vertices
- m: number of edges, assuming $n-1 \le m \le n(n-1)/2$
- d_v: number of neighbors of v

	Matrix	Linked Lists
memory usage	$O(n^2)$	$O(m)$
time to check $(u,v) \in E$	$O(1)$	$O(d_u)$
time to list all neighbours of v		

Comparison of Two Representations

- Assuming we are dealing with undirected graphs
- n: number of vertices
- m: number of edges, assuming $n-1 \le m \le n(n-1)/2$
- d_v: number of neighbors of v

	Matrix	Linked Lists
memory usage	$O(n^2)$	$O(m)$
time to check $(u, v) \in E$	$O(1)$	$O(d_u)$
time to list all neighbours of v	$O(n)$	

Comparison of Two Representations

- Assuming we are dealing with undirected graphs
- n: number of vertices
- m: number of edges, assuming $n - 1 \le m \le n(n-1)/2$
- d_v: number of neighbors of v

	Matrix	Linked Lists
memory usage	$O(n^2)$	$O(m)$
time to check $(u, v) \in E$	$O(1)$	$O(d_u)$
time to list all neighbours of v	$O(n)$	$O(d_v)$

Connectivity Problem

Input: graph $G = (V, E)$, (using linked lists)
two vertices $s, t \in V$

Output: whether there is a path connecting s to t in G

Connectivity Problem

Input: graph $G = (V, E)$, (using linked lists)

two vertices $s, t \in V$

Output: whether there is a path connecting s to t in G

- Algorithm: starting from s, search for all vertices that are reachable from s and check if the set contains t

Connectivity Problem

Input: **graph $G = (V, E)$, (using linked lists) two vertices $s, t \in V$**

Output: whether there is a path connecting s to t in G

- Algorithm: starting from s, search for all vertices that are reachable from s and check if the set contains t
  - Breadth-First Search (BFS)

Connectivity Problem

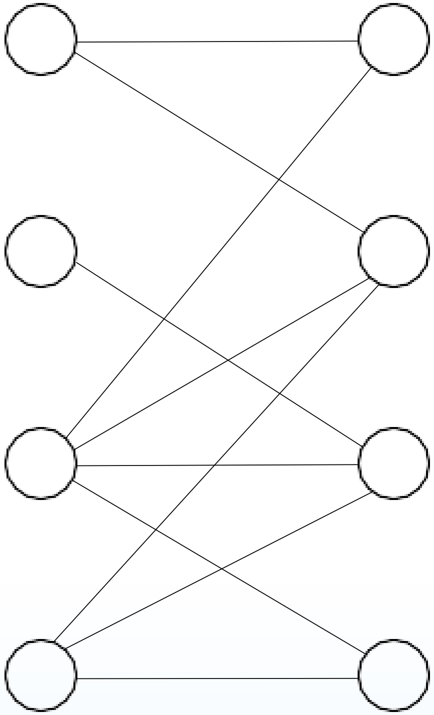
Input: **graph $G = (V, E)$, (using linked lists) two vertices $s, t \in V$**

Output: whether there is a path connecting s to t in G

- Algorithm: starting from s, search for all vertices that are reachable from s and check if the set contains t
  - Breadth-First Search (BFS)
  - Depth-First Search (DFS)

Bipartite Graphs

Def. A graph $G = (V, E)$ is a bipartite graph if there is a partition of V into two sets L and R such that for every edge $(u, v) \in E$, we have either $u \in L, v \in R$ or $v \in L, u \in R$.



Graphs (Networks) in Python

NetworkX at https://networkx.org/

Details at https://networkx.org/documentation/stable/reference/introduction.html

NetworkX

Network Analysis in Python

NetworkX is a Python package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks.

Software for complex networks

- Data structures for graphs, digraphs, and multigraphs
- Many standard graph algorithms
- Network structure and analysis measures
- Generators for classic graphs, random graphs, and synthetic networks
- Nodes can be "anything" (e.g., text, images, XML records)
- Edges can hold arbitrary data (e.g., weights, time-series)
- Open source 3-clause BSD license
- Well tested with over 90% code coverage
- Additional benefits from Python include fast prototyping, easy to teach, and multi-platform

BASIC TOOLS OF DATA SCIENCE

- Basic Technology Stack and Best Practices
  - Python, Jupyter Notebook,
  - GitHub
  - Cloud Computing
  - Containers (e.g., Docker)

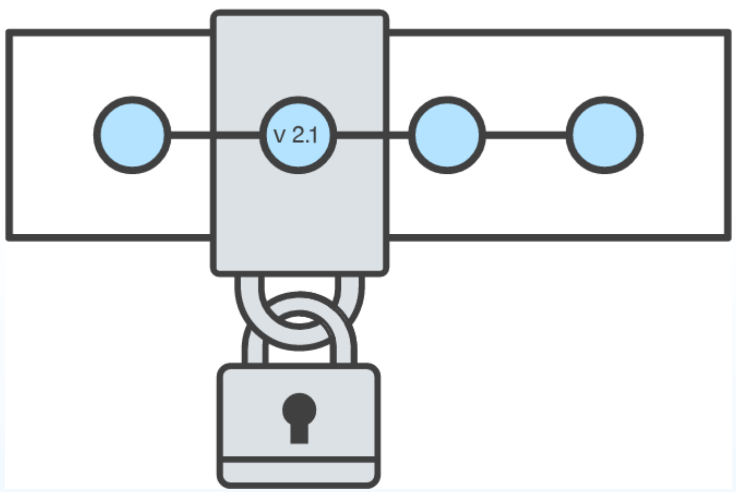
Python & Jupyter Notebook

```
import numpy as np
import numpy.random as rm
import scipy.stats as st
ar1=np.full(4,3)
print(ar1)

X=1000*rm.randn(20)+10000 #1000 is the standard deviation of normal distribution, 500 is the number of samples points and 10000 is the mean
Y= rm.randint(0,2,size=20)

d1=X
d2=Y
cor,_=st.pearsonr(d1,d2)
cor1,_=st.spearmanr(d1,d2)
print(d1)
print(d2)
print(cor1)
cor=st.pearsonr(d1,d2)
print(cor)
```

```
[3 3 3 3]
[10441.2789784  10353.44378245  9363.50548998 10606.05278723
  9817.14266103 10074.81506252 10801.28349835  9847.72135591
 11137.64477568 10327.55568368 10488.89720533 10185.89337601
  8083.94227971  9979.90596431 10161.12657669 10384.2755884
  8651.56661396 10636.47181246  9269.26895539  8879.84276296]
[0 1 0 1 1 1 0 0 0 1 1 0 1 0 1 0 0 1 0 0]
0.13072174640400053
(0.10398006545187807, 0.6626502989542163)
```

GOALS OF VERSION CONTROL

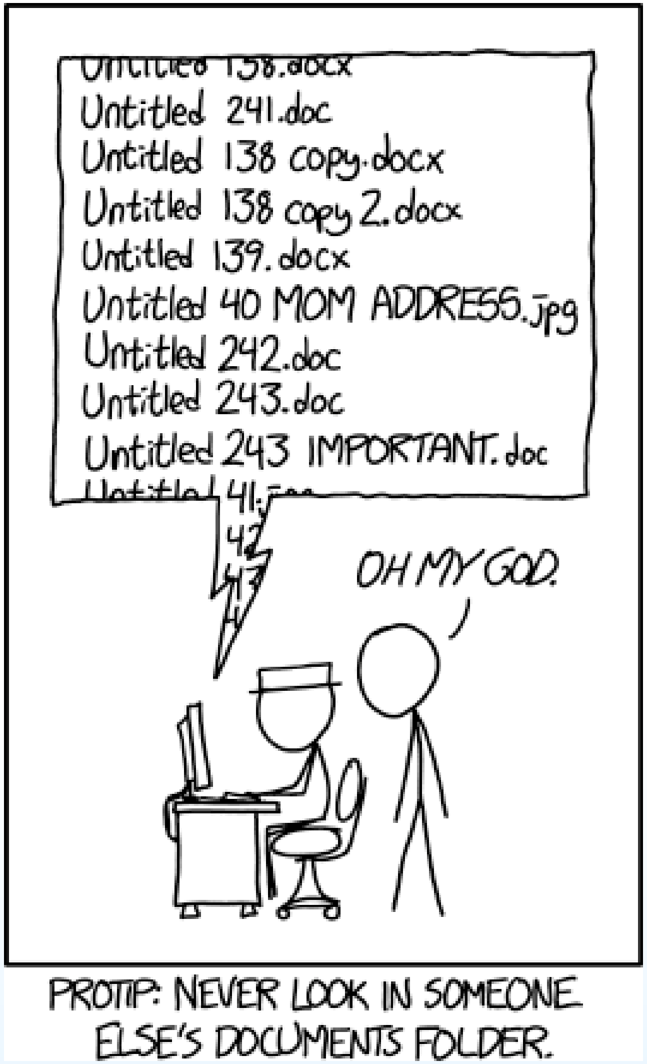
- Be able to search through revision history and retrieve previous versions of any file in a project
- Be able to share changes with collaborators on a project
- Be able to confidently make large changes to existing files



atlassian.com/git/tutorials/what-is-version-control

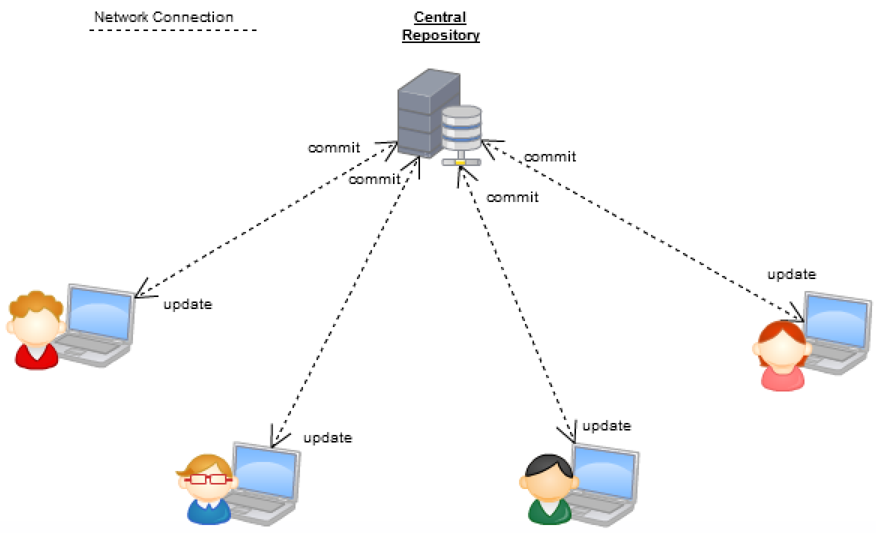
NAMED FOLDERS APPROACH

- Can be hard to track
- Memory-intensive
- Can be slow
- Hard to share
- No record of authorship



CENTRALIZED VERSION CONTROL SYSTEMS

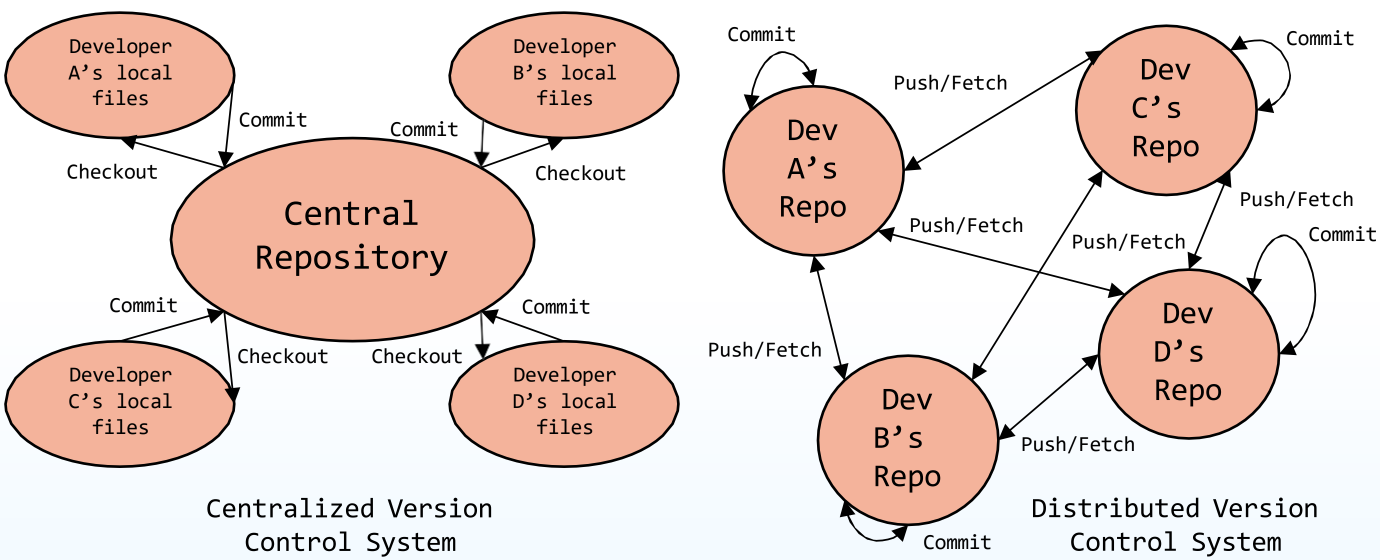
- A central, trusted repository determines the order of commits ("versions" of the project)
- Collaborators "push" changes (commits) to this repository.
- Any new commits must be compatible with the most recent commit. If it isn't, somebody must "merge" it in.



- Examples:, CVS, SVN (advanced & newer system than CVS), Perforce (commercial)

DISTRIBUTED VERSION CONTROL SYSTEMS (DVCS)

- No central repository
- Every repository has every commit
- Examples: Git, Mercurial



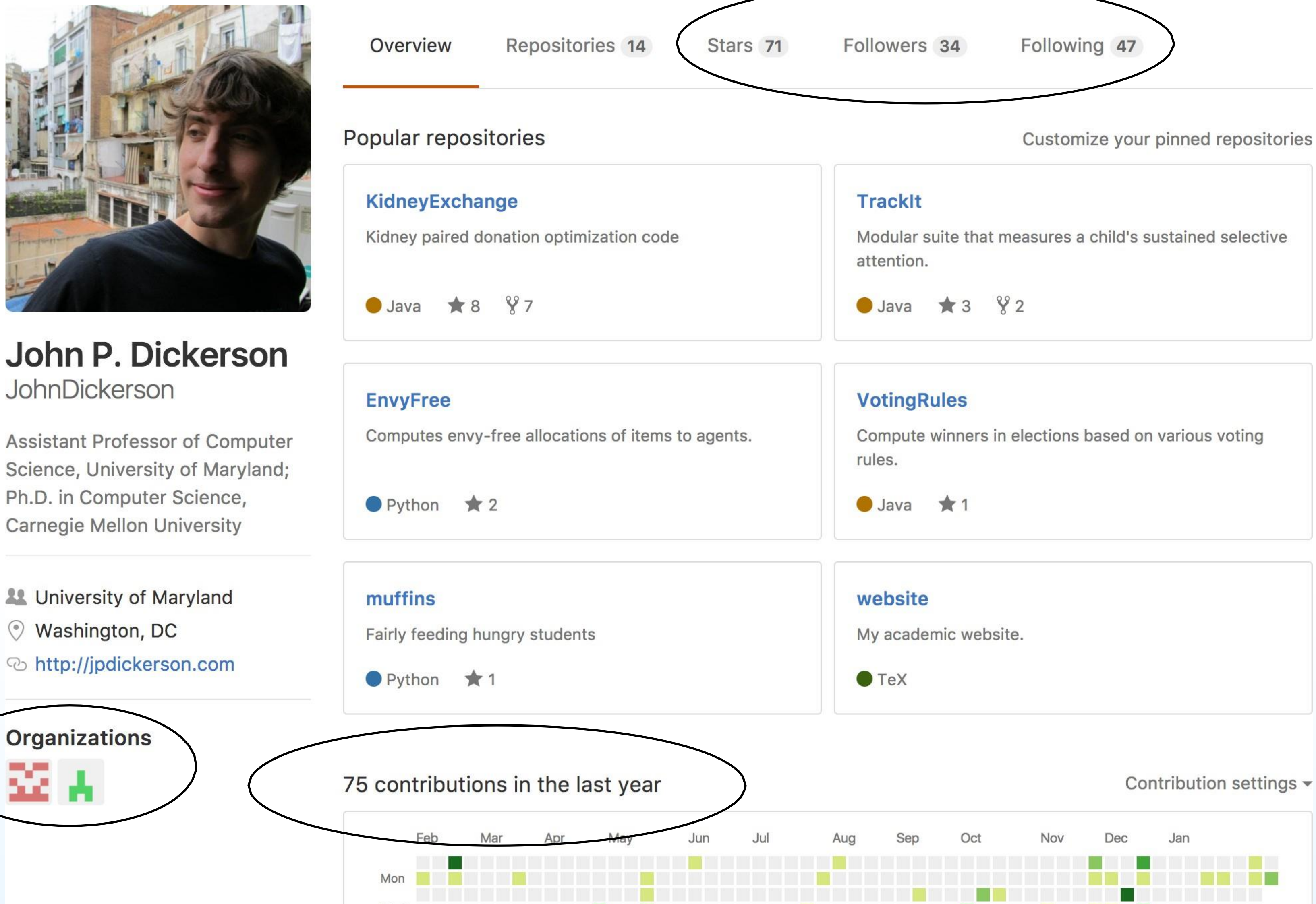
WHAT IS GIT

- Git is a ***distributed*** version control system
- Developed as a repository system for both local and remote changes
- Allows teammates to work simultaneously on a project
- Tracks each commit, allowing for a detailed documentation of the project along every step
- Allows for advanced merging and branching operations

SO YOU WANT SOMEBODY ELSE TO HOST THIS FOR YOU …

- Git: general distributed version control system
- GitHub / BitBucket / GitLab / …: hosting services for git repositories
- In general, GitHub is the most popular:
  - Lots of big projects (e.g., Python, Bootstrap, Angular, D3, node, Django, Visual Studio)
  - Lots of ridiculously awesome projects (e.g., https://github.com/maxbbraun/trump2cash)
- There are reasons to use the competitors esp. BitBucket(e.g., private repositories, access control)
- BitBucket website: https://bitbucket.org/
- BitBucket commands: https://confluence.atlassian.com/bitbucketserver/basic-git-commands-776639767.html

"SOCIAL CODING"



REVIEW: HOW TO USE

- Git commands for everyday usage are relatively simple
  - git pull
    - Get the latest changes to the code
  - git add .
    - Add any newly created files to the repository for tracking
  - git add –u
    - Remove any deleted files from tracking and the repository
  - git commit –m ‘Changes’
    - Make a version of changes you have made
  - git push
    - Deploy the latest changes to the central repository
- Make a repo on GitHub and clone it to your machine:
  - https://guides.github.com/activities/hello-world/

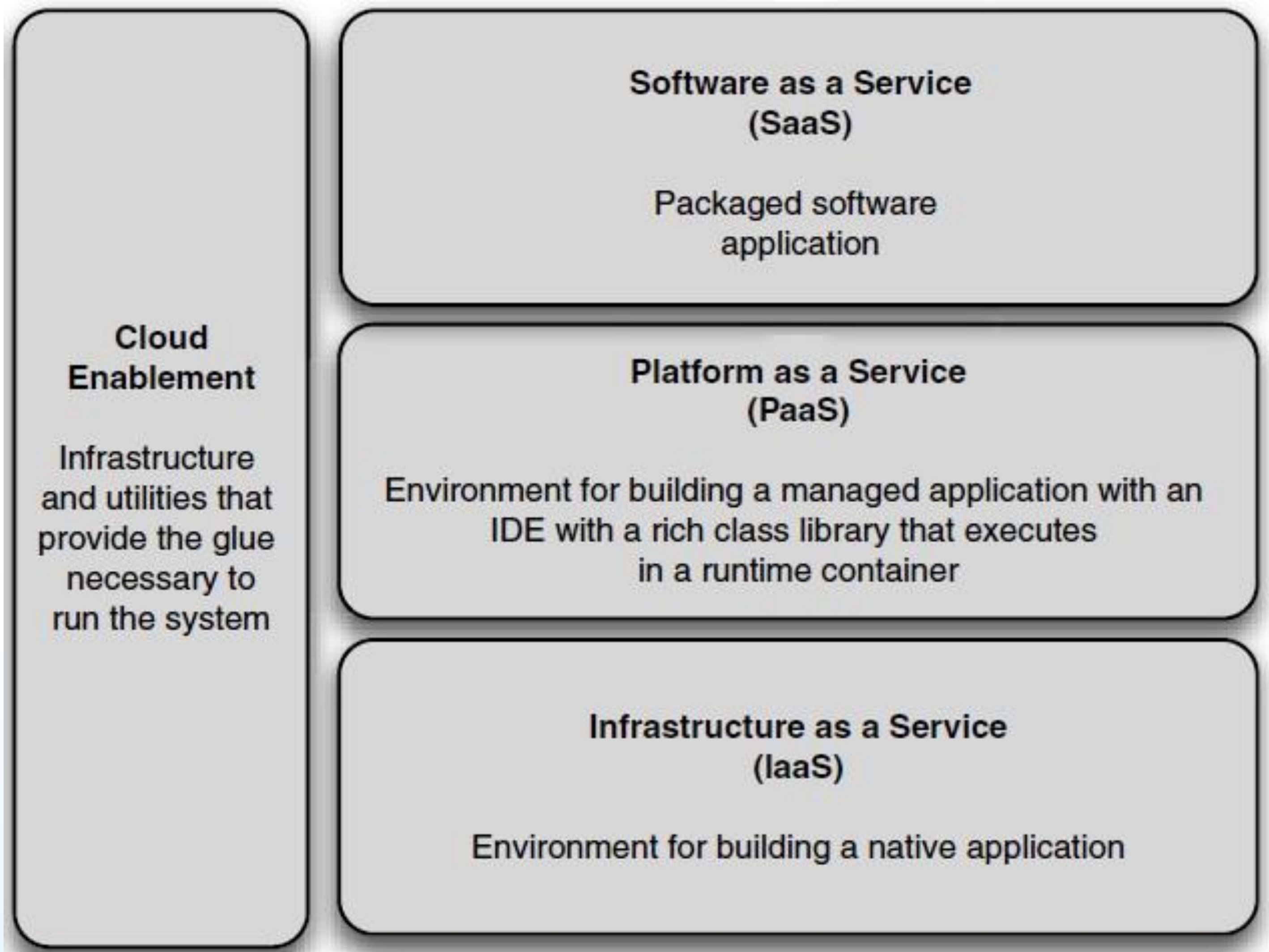
CLOUD COMPUTING

- Computing as a "service" rather than a "product"
  - Everything happens in the "cloud": both storage and computing
  - Personal devices (laptops/tablets) simply interact with the cloud
- Advantages
  - Device agonstic – can seamlessly move from one device to other
  - Efficiency/scalability: programming frameworks allow easy scalability (relatively speaking)
  - Reliability
  - Cost: "pay as you go" allows renting computing resources as needed – much cheaper than building your own systems

CLOUD COMPUTING

- Basic ideas have been around for a long time (going back to 1960's)
  - Mainframes + thin clients (more by necessity)
  - Grid computing a few year ago
  - Peer-to-peer
  - Client-server models
  - …
- But it finally works as we wished for…
  - Why now?... A convergence of several key pieces over the last few years
  - Does it really? … Still many growing pains

X-AS-A-SERVICE



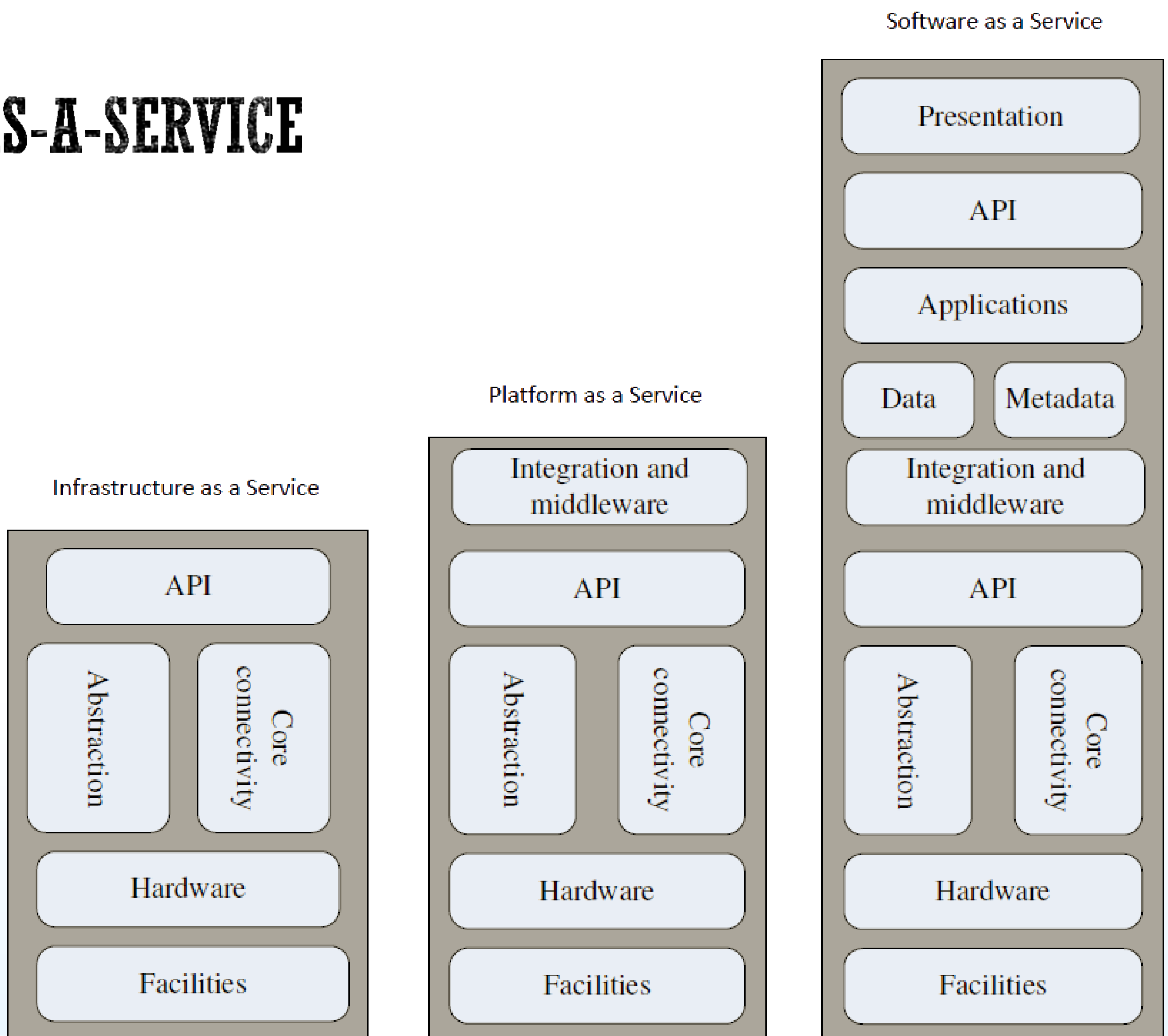
EXAMPLES

IaaS — infrastructure as a service. You can find examples of typical IaaS services e.g. at OVH as Public or Private Cloud, or at AWS as cloud computing.

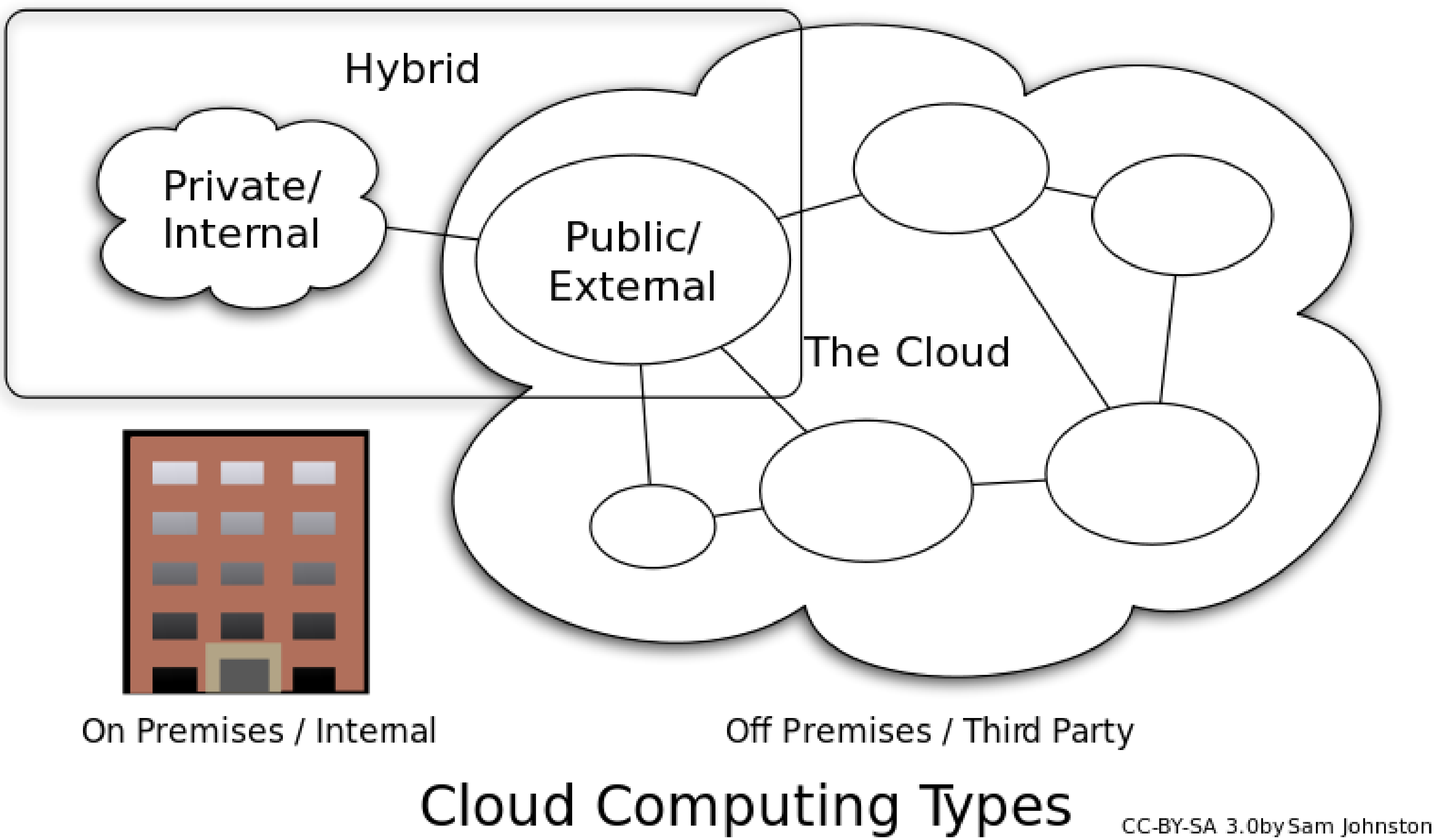
PaaS — platform as a service. Cloud development environment Codenvy; Google App Engine, Microsoft Azure or AWS applications; Docker; serverless applications AWS, Oracle databases, etc.

SaaS — software as a service. For end users: Office 365 from Microsoft, Google service.

X-AS-A-SERVICE



CLOUD TYPES



Hybrid
Private/
Internal
Public/
External
The Cloud
On Premises / Internal
Off Premises / Third Party
Cloud Computing Types
CC-BY-SA 3.0 by Sam Johnston

DATA CENTERS

- The key infrastructure piece that enables CC
- Everyone is building them
- Huge amount of work on deciding
them



VIRTUALIZATION

- Virtual machines (e.g., running Windows inside a Mac) etc. has been around for a long time
  - Used to be very slow…
  - Only recently became efficient enough to make it a key for CC
- Basic idea: run virtual machines on your servers and sell time on them
  - That's how Amazon EC2 runs
- Many advantages:
  - Security: virtual machines serves as almost impenetrable boundary
  - Multi-tenancy: can have multiple VMs on the same server
  - Efficiency: replace many underpowered machines with a few high-power machines

VIRTUALIZATION

- Consumer VM products include VMWare, Parallels (for Mac), VirtualBox etc…

- Amazon uses "Xen" running on Redhat machines (may be old information)
  - They support both Windows and Linux Virtual Machines

- Some tricky things to keep in mind:
  - Harder to reason about performance (if you care)
  - Identical VMs may deliver somewhat different performance

- Much continuing work on the virtualization technology itself

DOCKER

- Hottest thing right now…
  - Avoid the overheads of virtualization altogether