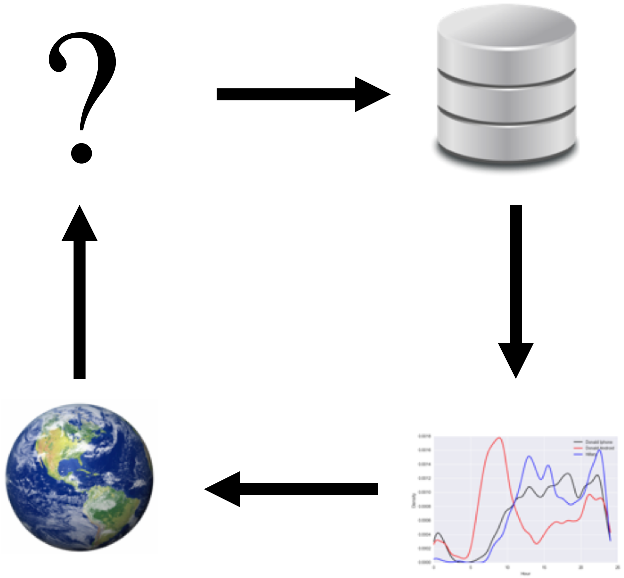
Data Science 100

Midterm Review



Slides by:

Joseph E. Gonzalez, Deb Nolan, & Fernando Perez

jegonzal@berkeley.edu

deborah_nolan@berkeley.edu

fernando.perez@berkeley.edu

Exam Format Details

- **When:** 11:00-12:30PM Thursday 8^{th}
  - 80 minutes long
- **Where:** in lecture (Wheeler)
  - **DSP details** over email
  - Makeup exams have been schedule
- **What to bring:**
  - Berkeley Cal Id Card (we have to check…)
  - Pencils + eraser
  - Study Guide -- 1 page front and back
    - You may type it but miniaturizing lectures is not a good idea
- **What to study:** Everything up to and including lec. 14
  - Homework, labs, section notes …

Review

Topics Students Asked About

- Loss Functions and Loss Minimization
- Gradient Descent
- Do I need to program?
- Bad Plots (jiggling, stacking etc…)
- Transformations
- Everything else …

Modeling and Estimation

Summary of Model Estimation

1. **Define the Model:** simplified representation of the world
   - ➢ Use domain knowledge but … ***keep it simple!***
   - ➢ Introduce **parameters** for the unknown quantities
2. **Define the Loss Function:** measures how well a particular instance of the model "fits" the data
   - ➢ We introduced L^2, L^1, and Huber losses for each record
   - ➢ Take the average loss over the entire dataset
3. **Minimize the Loss Function:** find the parameter values that minimize the loss on the data
   - ➢ Analytically using calculus
   - ➢ Numerically using gradient descent

Define the Model

- Motivating example of computing the percentage tip
  - We explored the constant tip model
- A more interesting model:

$$\texttt{percentage tip} = \theta_1^* + \theta_2^* * \texttt{total bill}$$

Rationale:
Larger bills result in larger tips and people tend to to be more careful or stingy on big tips.

Parameter Interpretation:
- θ_1: Base tip percentage
- θ_2: Reduction/increase in tip for an increase in total bill.

Recommendation Systems Model

Not on the midterm … but we will review it briefly here

- How do we recommend movies to people?
  - Collect user ratings for a bunch of movies

User u's star rating for movie m

$= \theta^*$: Model 1 (kind of boring …)

$= \theta_1^* + \theta_2^* \times \mathbb{I}\left[\texttt{hasBrad}(m)\right] + \theta_3^* \times \texttt{boxOfficeRevenue}(m)$: Model 2 (properties of movie)

$= \theta_1^* + \theta_2^* \times \mathbb{I}\left[\texttt{hasBrad}(m) \textbf{ AND } \texttt{female}(u)\right] + \theta_3^* \times \texttt{boxOfficeRevenue}(m)$

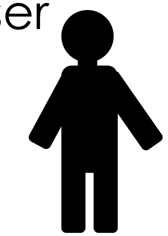
: Model 3 (properties of movie and user)

- Using the model

If we knew the parameters: (we don't)

$$\theta_1^* = 2.4$$
$$\theta_2^* = 1.3$$
$$\theta_3^* = 1.0 \times 10^{-8}$$

Female User



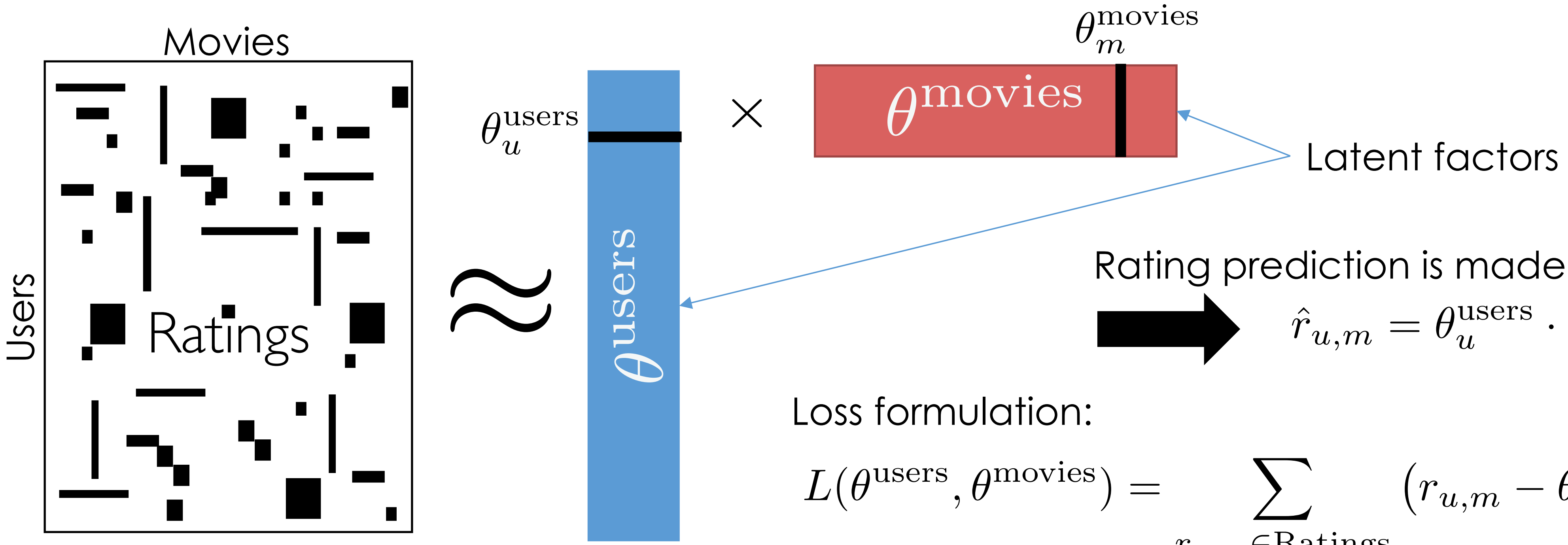
Staring Brad Pitt
boxOfficeRevenue:
60M

$$2.4 + 1.3 + 10^{-8} \times (60 \times 10^6)$$
$$= 4.3$$

Recommendation Systems Model

Not on the midterm … but we will review it briefly here

- What if we don't have any information about the movies or the users? All we have are ratings.



Rating prediction is made by:

$$\hat{r}_{u,m} = \theta_u^{\text{users}} \cdot \theta_m^{\text{movies}}$$

Loss formulation:

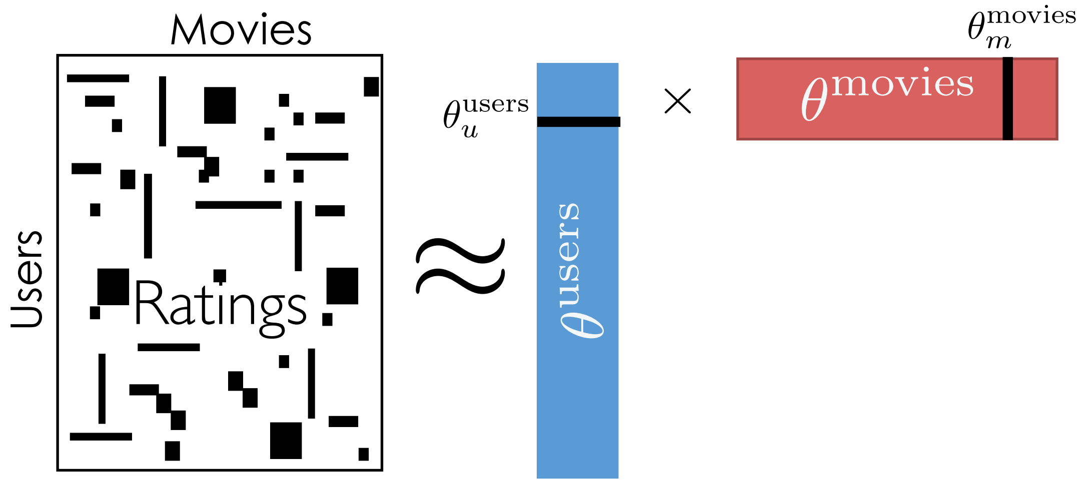
$$L(\theta^{\text{users}}, \theta^{\text{movies}}) = \sum_{r_{u,m} \in \text{Ratings}} \left(r_{u,m} - \theta_u^{\text{users}} \cdot \theta_m^{\text{movies}}\right)^2$$

How do we estimate model parameters?

$$\texttt{percentage tip} = \theta_1^* + \theta_2^* * \texttt{total bill}$$

$$\text{User } u\text{'s star rating for movie } m \;= \theta^*$$

$$= \theta_1^* + \theta_2^* \times \mathbb{I}\left[\texttt{hasBrad}(m) \textbf{ AND } \texttt{female}(u)\right] + \theta_3^* \times \texttt{boxOfficeRevenue}(m)$$



1. Define a model
   - Parametric models (so far …)
2. Define an objective (the loss function)
   - Choice has impact on answer (tradeoff)
3. Optimize the objective
   - Calculus
   - Numerically (gradient descent)

Loss Functions

- **Loss function**: a function that characterizes the cost, error, or loss resulting from a choice of model and parameters.

Squared Loss (L^2)

$$L\left(\theta, y\right) = \left(y - \theta\right)^2$$

Absolute Loss (L^2)

$$L\left(\theta, y\right) = \left|y - \theta\right|$$

Huber Loss

$$L_\alpha\left(\theta, y\right) = \begin{cases} \frac{1}{2}\left(y - \theta\right)^2 & \left|y - \theta\right| < \alpha \\ \alpha\left(\left|y - \theta\right| - \frac{\alpha}{2}\right) & \text{otherwise} \end{cases}$$

Calculus for Loss Minimization

- General Procedure:
  - Verify that function is convex (we often will assume this…)
  - Compute the derivative
  - Set derivative equal to zero and solve for the parameters
- Using this procedure we discovered **the loss minimizers:**

$$\hat{\theta}_{L^2} = \frac{1}{n}\sum_{I=1}^{n} y_i = \textbf{mean}(\mathcal{D}) \qquad \hat{\theta}_{L^1} = \textbf{median}(\mathcal{D})$$

Example: Minimizing Average L² Loss

Average Loss (L²)

1. $$L_{\mathcal{D}}(\theta) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \theta)^2$$

Derivative of the Average Loss (L²)

2. $$\frac{\partial}{\partial\theta}L_{\mathcal{D}}(\theta) = \frac{1}{n}\sum_{i=1}^{n}\frac{\partial}{\partial\theta}(y_i - \theta)^2$$

$$= -\frac{2}{n}\sum_{i=1}^{n}(y_i - \theta)$$

Set derivative = 0 and solve for θ…

3. $$0 = -\frac{2}{n}\sum_{i=1}^{n}(y_i - \theta)$$

$$0 = \left(\sum_{i=1}^{n} y_i\right) - n\theta$$

$$\hat{\theta} = \frac{1}{n}\sum_{i=1}^{n} y_i$$

Essential Calculus: The Chain Rule

- How do I compute the derivative of composed functions?

$$\frac{\partial}{\partial \theta} h(\theta) = \frac{\partial}{\partial \theta} f\left(g(\theta)\right)$$

$$= \left(\frac{\partial}{\partial u} f(u) \Big|_{u=g(\theta)} \right) \frac{\partial}{\partial \theta} g(\theta)$$

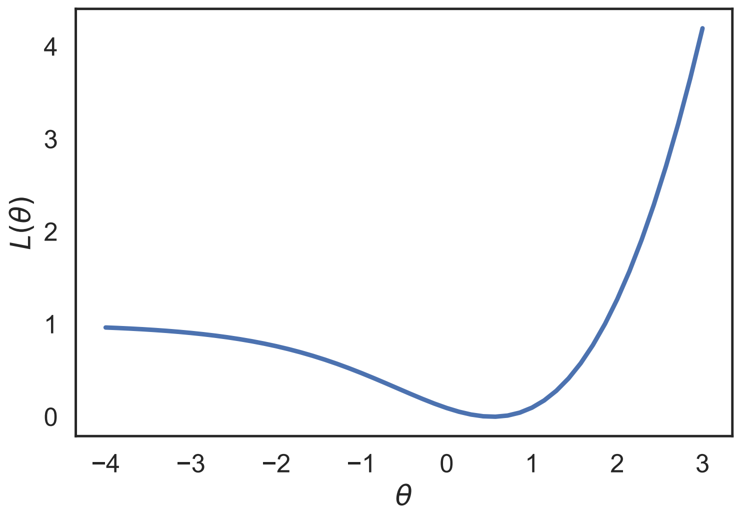
Derivative of *f* evaluated at *g(θ)*

Derivative of *g(θ)*

Know how to calculate derivatives of logs, exponents, and exponentials.

Exercise of Calculus

- Minimize: $L(\theta) = \left(1 - \log\left(1 + \exp(\theta)\right)\right)^2$



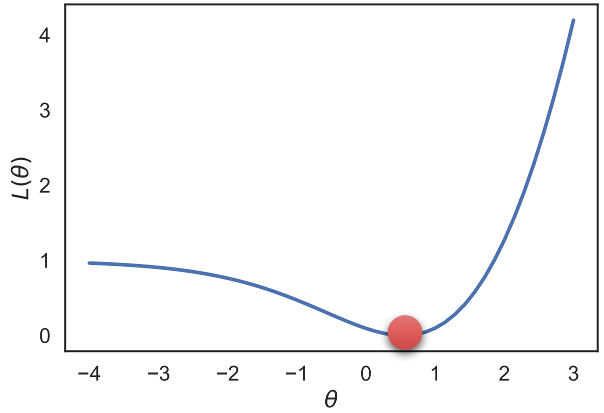
- Take the derivative:

$$
\begin{aligned}
\frac{\partial}{\partial \theta} L(\theta) &= \frac{\partial}{\partial \theta}\left(1 - \log\left(1 + \exp(\theta)\right)\right)^2 \\
&= 2\left(1 - \log\left(1 + \exp(\theta)\right)\right) \frac{\partial}{\partial \theta}\left(1 - \log\left(1 + \exp(\theta)\right)\right) \\
&= 2\left(1 - \log\left(1 + \exp(\theta)\right)\right) (-1) \frac{\partial}{\partial \theta} \log\left(1 + \exp(\theta)\right) \\
&= 2\left(1 - \log\left(1 + \exp(\theta)\right)\right) \frac{-1}{1 + \exp(\theta)} \frac{\partial}{\partial \theta}\left(1 + \exp(\theta)\right) \\
&= 2\left(1 - \log\left(1 + \exp(\theta)\right)\right) \frac{-1}{1 + \exp(\theta)} \exp(\theta)
\end{aligned}
$$

- Take the derivative:

$$\frac{\partial}{\partial\theta}L(\theta) = 2\left(1-\log\left(1+\exp(\theta)\right)\right)\frac{-1}{1+\exp(\theta)}\exp(\theta)$$

$$= -2\left(1-\log\left(1+\exp(\theta)\right)\right)\frac{\exp(\theta)}{1+\exp(\theta)}$$



- Set derivative equal to zero and solve for parameter

$$-2\left(1-\log\left(1+\exp(\theta)\right)\right)\frac{\exp(\theta)}{1+\exp(\theta)} = 0 \quad \Rightarrow \quad 1-\log\left(1+\exp(\theta)\right) = 0$$

Solving for parameters

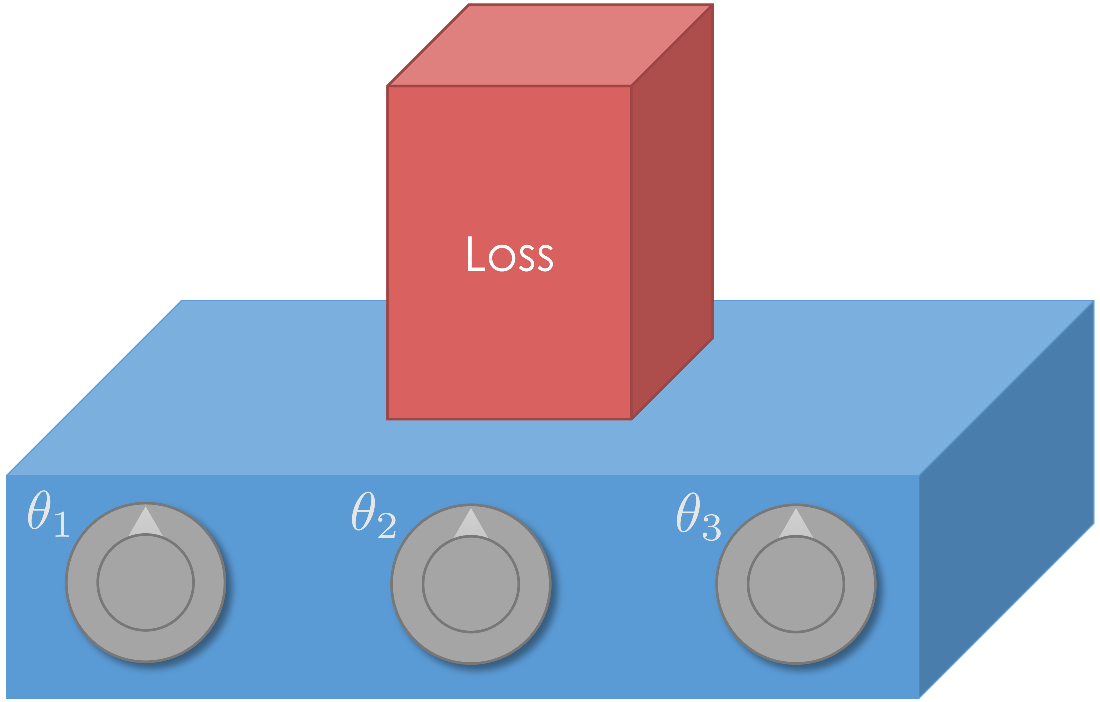
$$\begin{aligned} \log\left(1+\exp(\theta)\right) &= 1 \\ 1+\exp(\theta) &= \exp(1) \\ \exp(\theta) &= \exp(1)-1 \\ \theta &= \log\left(\exp(1)-1\right) \approx 0.541 \end{aligned}$$

Minimizing the Loss

- Calculus techniques can be applied generally …
- Guaranteed to minimize the loss when **loss** is convex in the parameters
- May not always have an analytic solution …

Gradient Descent

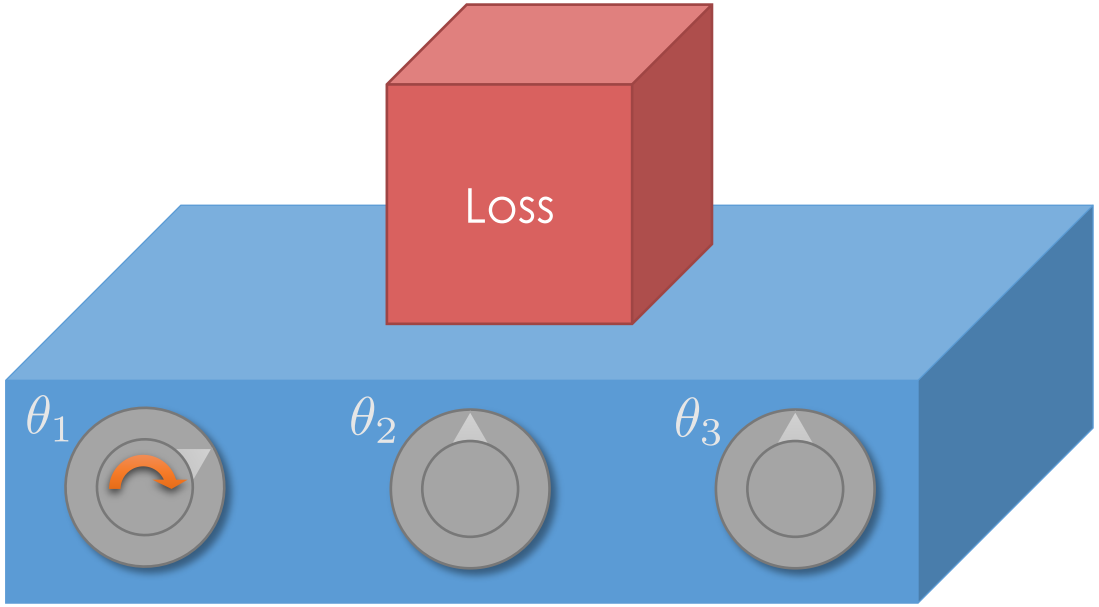
Intuition



Goal: Minimize the loss by turning the knobs.

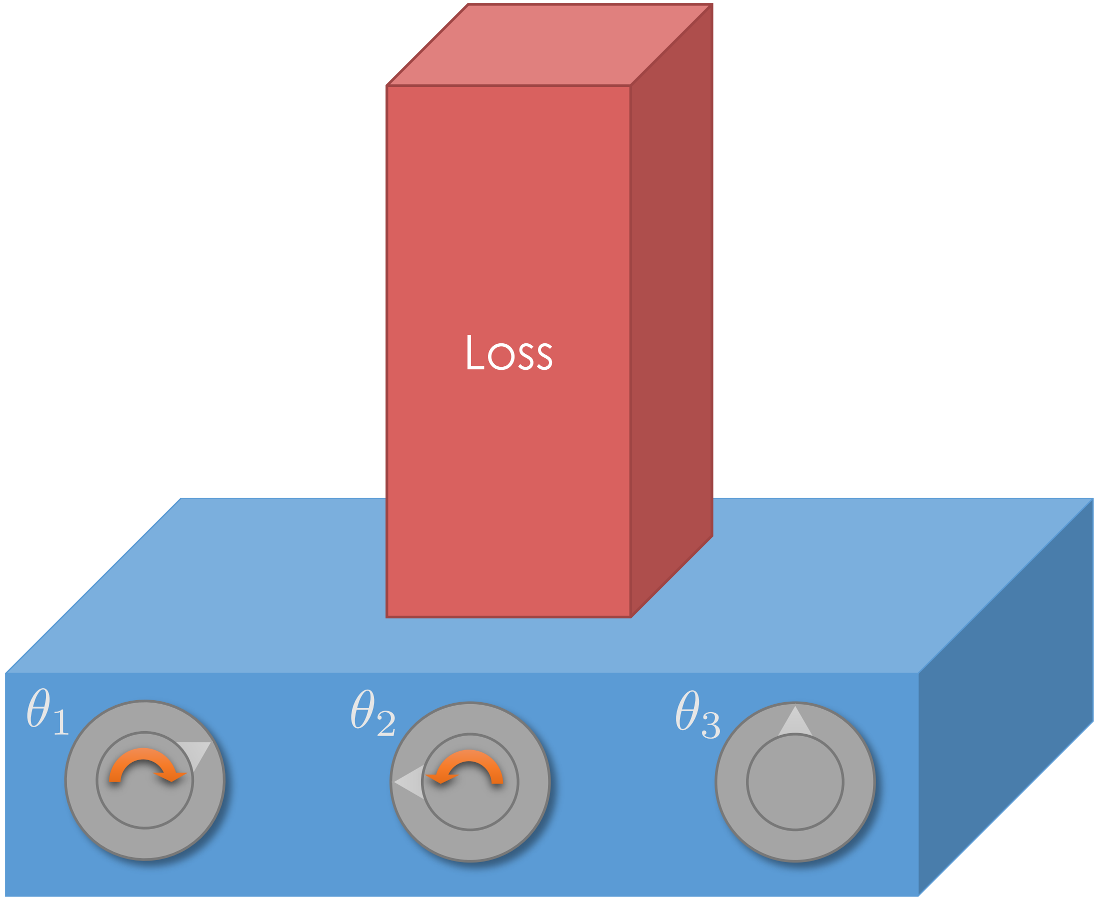
Try the loss game (its free)!

Intuition



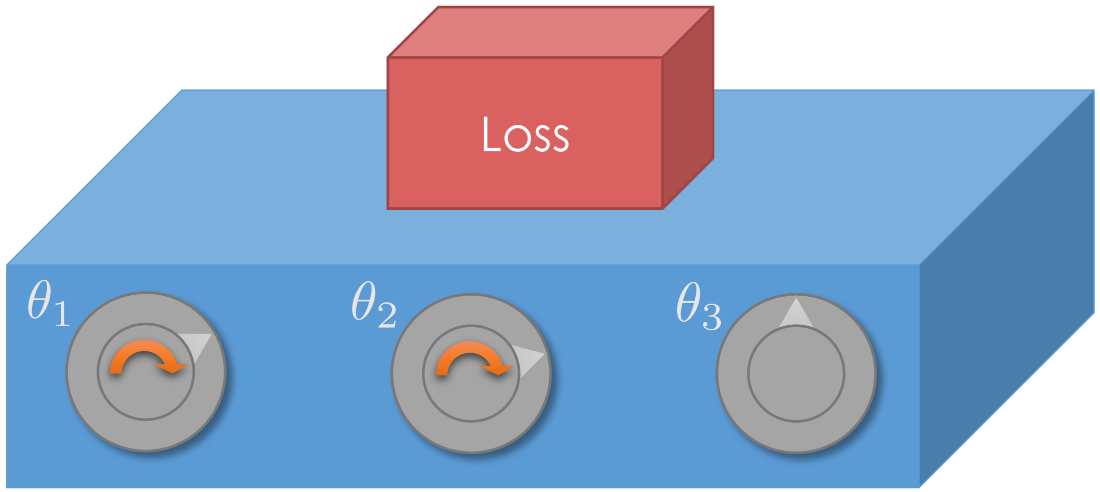
Try the loss game (its free)!

Intuition



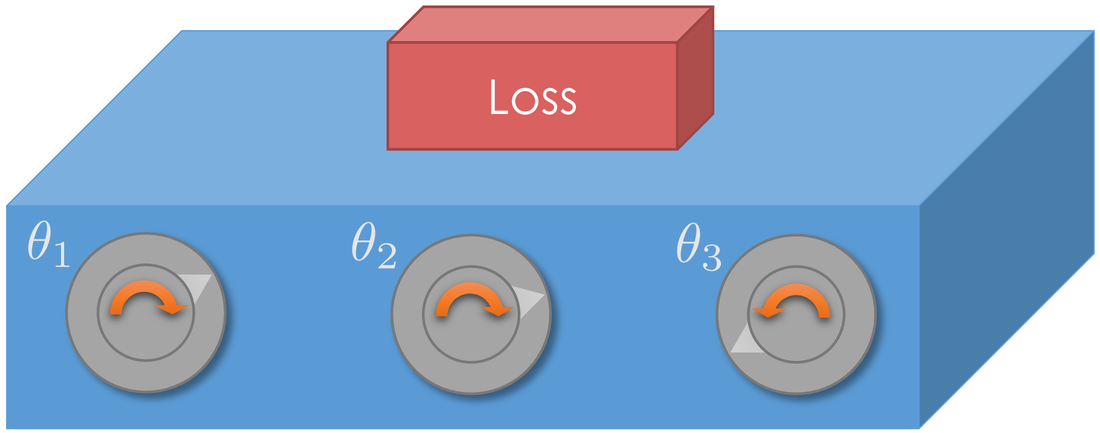
Try the loss game (its free)!

Intuition



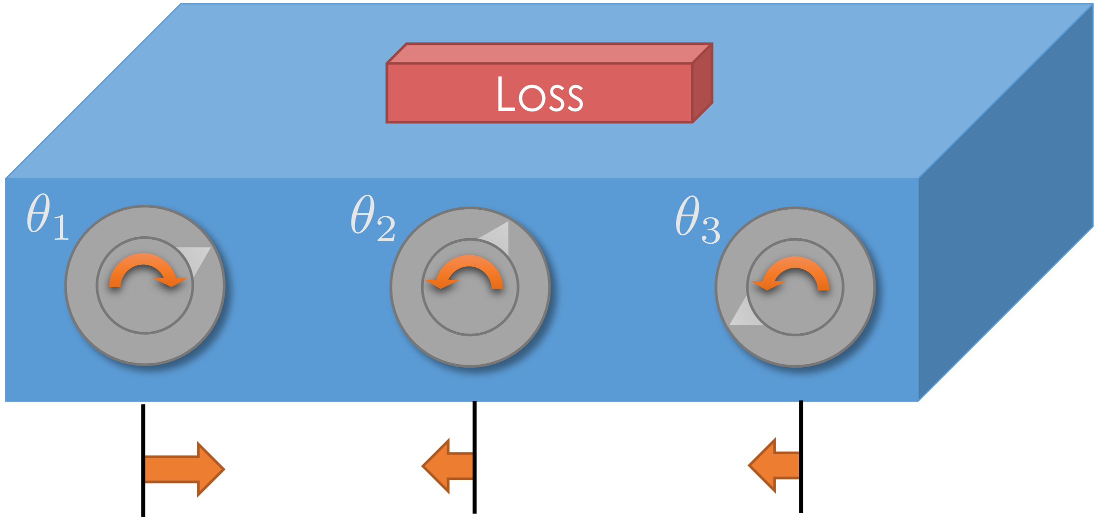
Try the loss game (its free)!

Intuition



Try the loss game (its free)!

Intuition

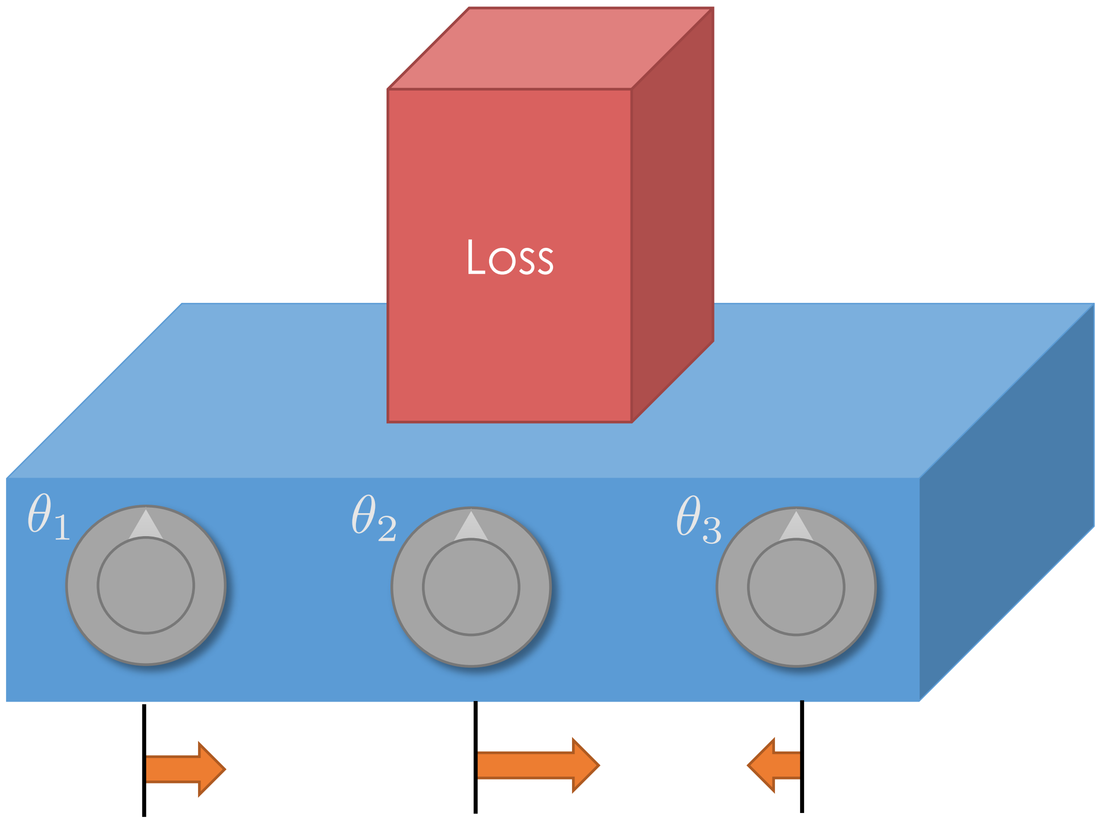


What if we knew which way to turn the knob and an idea of how far?

This is the Gradient!

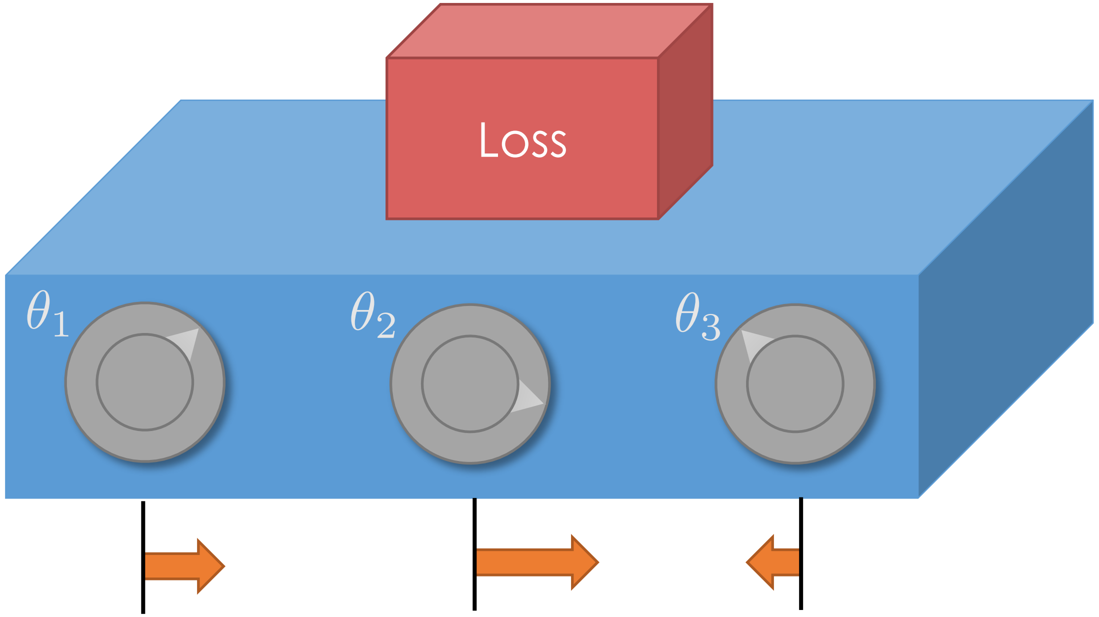
Try the loss game (its free)!

Intuition



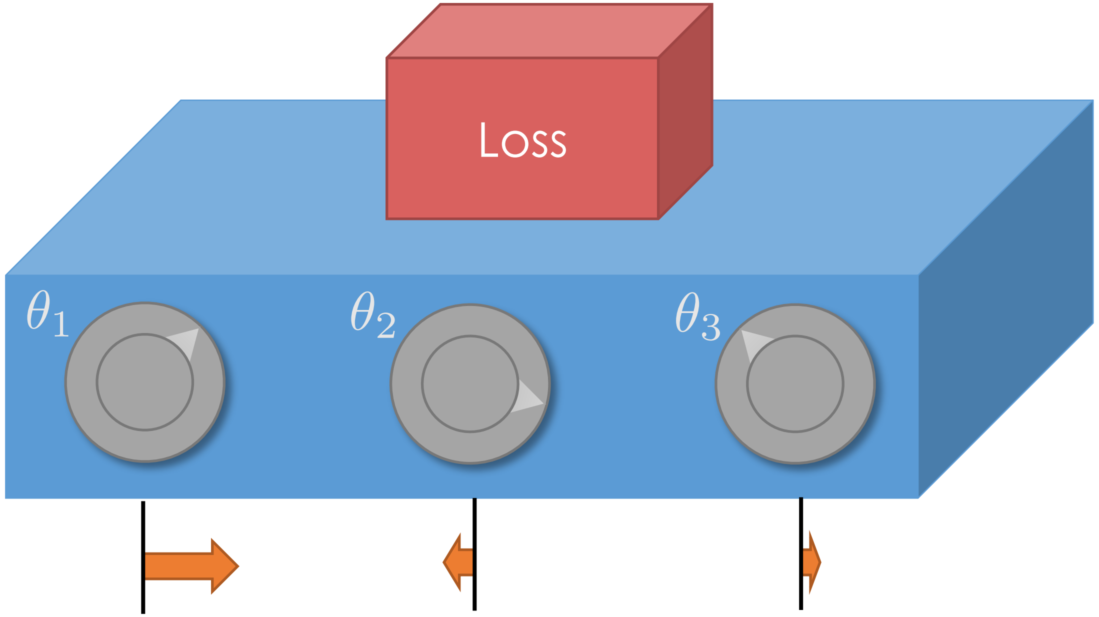
Try the loss game (its free)!

Intuition



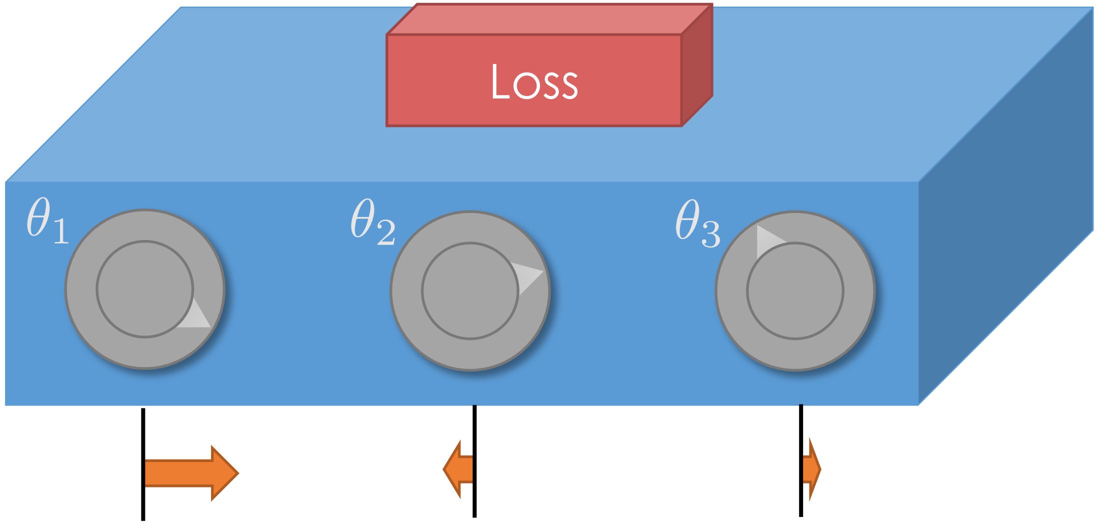
Try the loss game (its free)!

Intuition



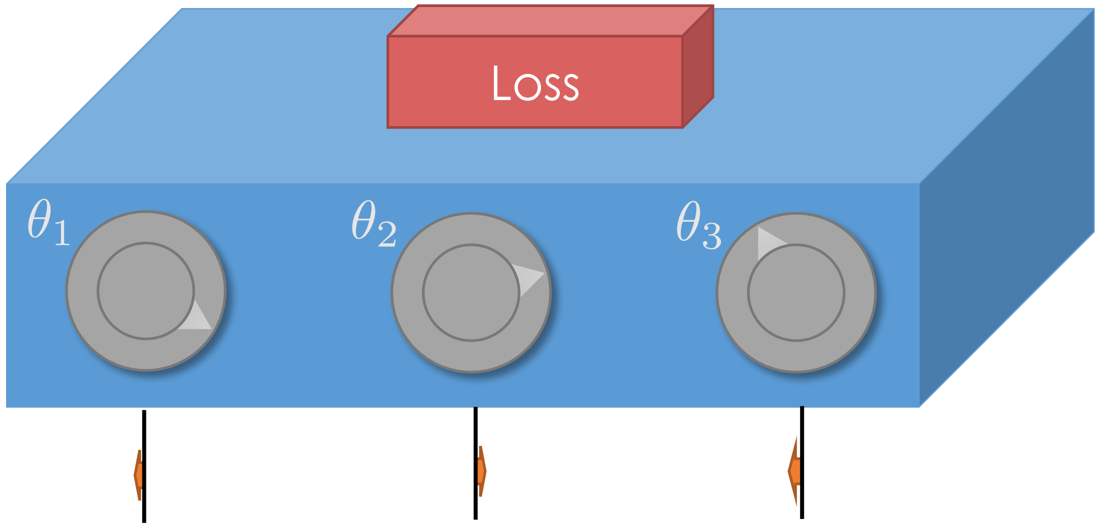
Try the loss game (its free)!

Intuition



Try the loss game(its free)!

Intuition



This is the Gradient descent!

Try the loss game (its free)!

Quick Review: **Gradients**

Loss function

$$f : \mathbb{R}^p \to \mathbb{R}$$

For Example:

$$f(\theta_1, \theta_2, \theta_3) = a\theta_1 + b\theta_2 + c\theta_2\theta_3^2$$

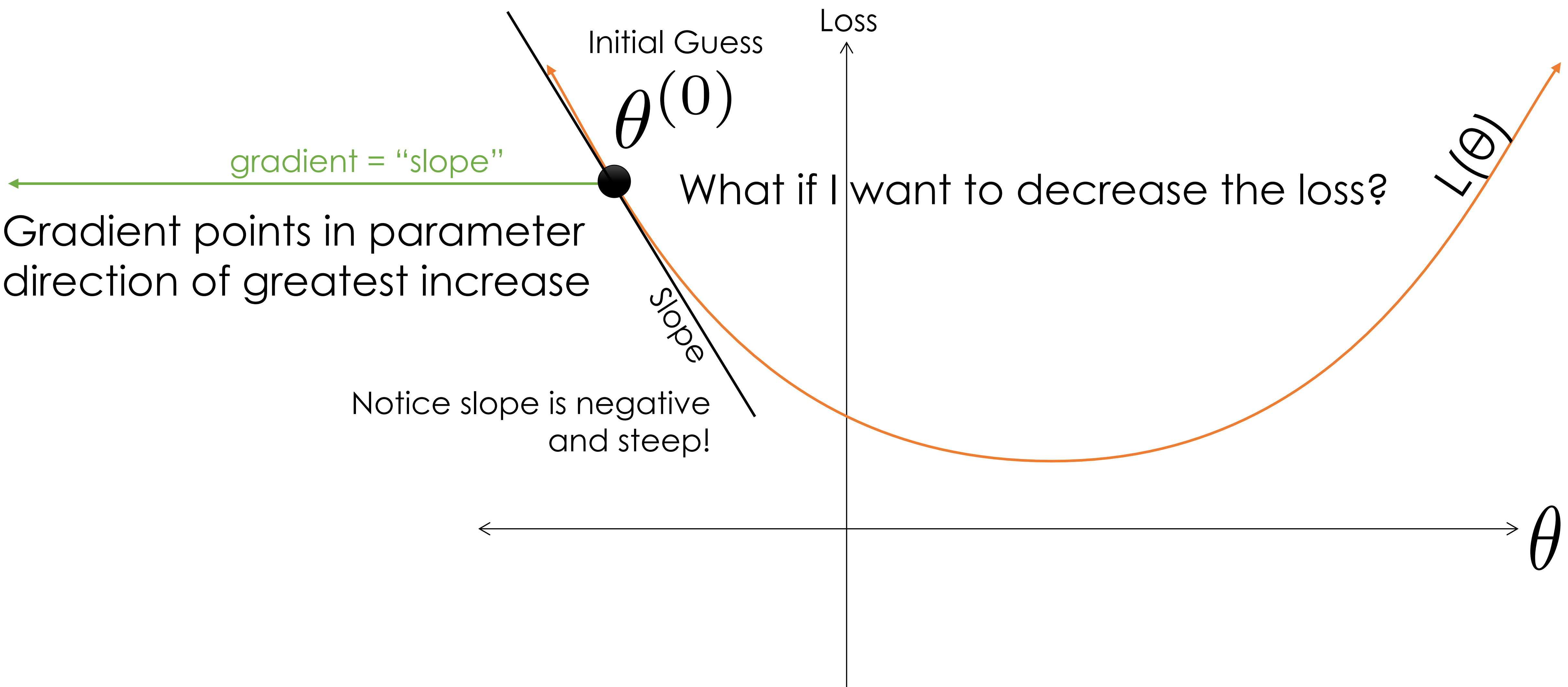
- Gradient: $g : \mathbb{R}^p \to \mathbb{R}^p$

$$\nabla_\theta f(\theta_1, \theta_2, \theta_3) = \left[a, b + c\theta_3^2, 2c\theta_2\theta_3\right]$$

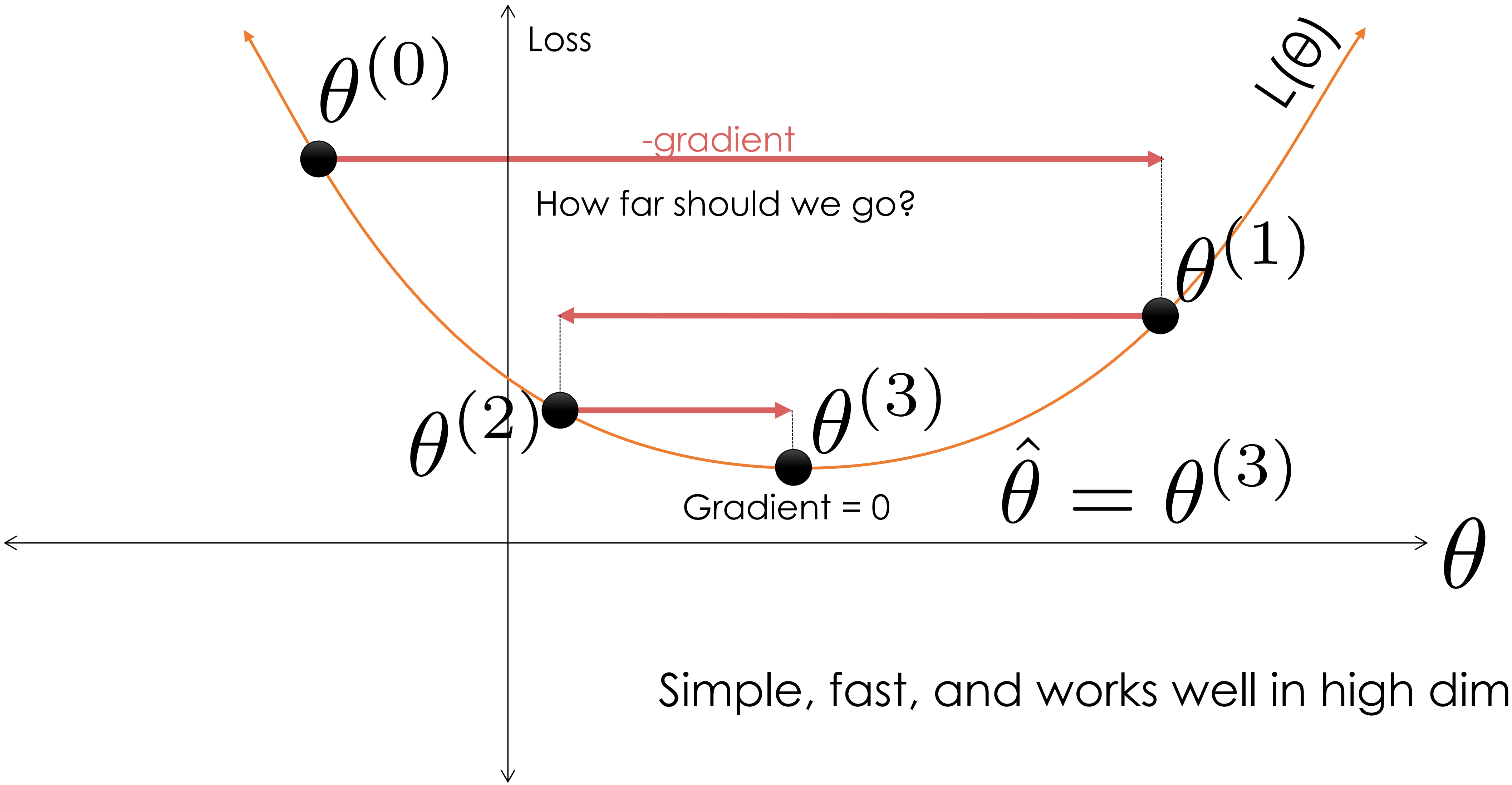
$$g(\theta) = \nabla_\theta f(\theta)$$

$$= \left[\frac{\partial}{\partial \theta_1} f(\theta)|_\theta, \ldots, \frac{\partial}{\partial \theta_3} f(\theta)|_\theta\right]$$

Gradient Descent Intuition



Gradient Descent Intuition



Simple, fast, and works well in high dimensions.

The Gradient Descent Algorithm

$\theta^{(0)} \leftarrow$ initial vector (random, zeros …)

For τ from 0 to convergence:

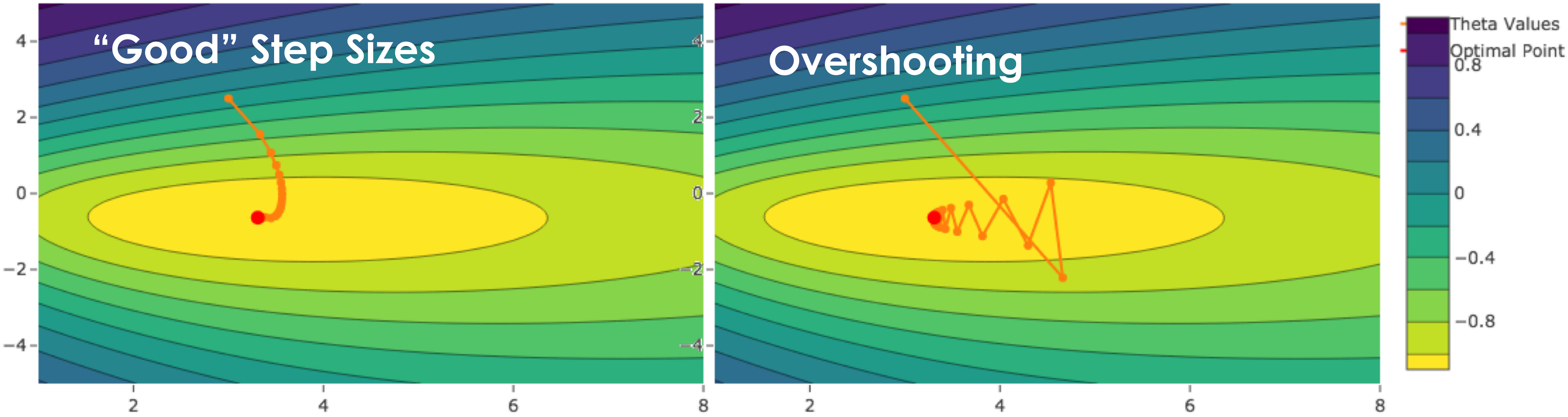
$$\theta^{(\tau+1)} \leftarrow \theta^{(\tau)} - \rho(\tau) \left(\nabla_\theta \mathbf{L}(\theta) \Big|_{\theta=\theta^{(\tau)}} \right)$$

Evaluated at

- **ρ(τ)** is the step size (learning rate)
  - typically $1/\tau$
- Converges when gradient is ≈ 0 (or we run out of patience)

Gradient Descent Solution Paths

- ➢ Orange line is path taken by gradient descent
  - ➢ Contours are from loss on two parameter model



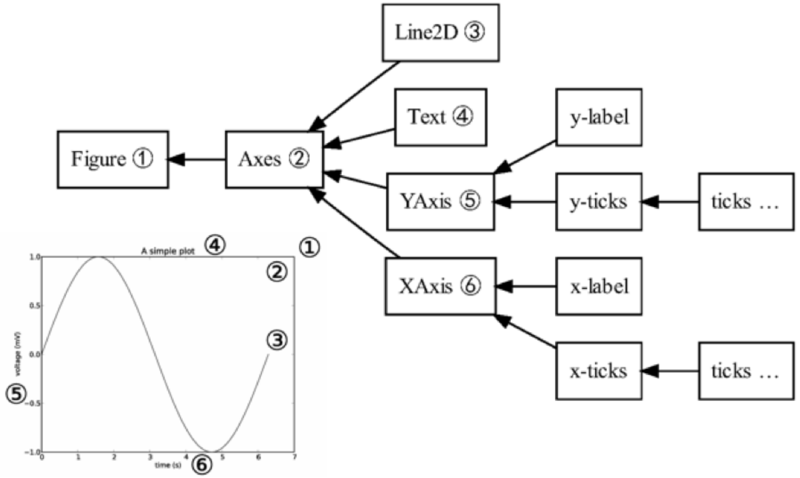
Code

Python + Numpy + Pandas + Seaborn + SQL + Regex +HTTP

Coding on the Exam

- You will not be required to write large programs
- You **will** be required to write "*one line*" programs:
  - long line … `df.groupby(…)[…].count(…).sort_values()`
  - DataFrame transformations (merge, groupby, value_counts, pivot_table, loc, mean, min, max, count, slicing)
  - Regular expressions
  - String Manipulation
- Should be comfortable reading python code and explaining what is happening.
- We will provide code cheat sheet for complex functions
  - See practice exam questions …

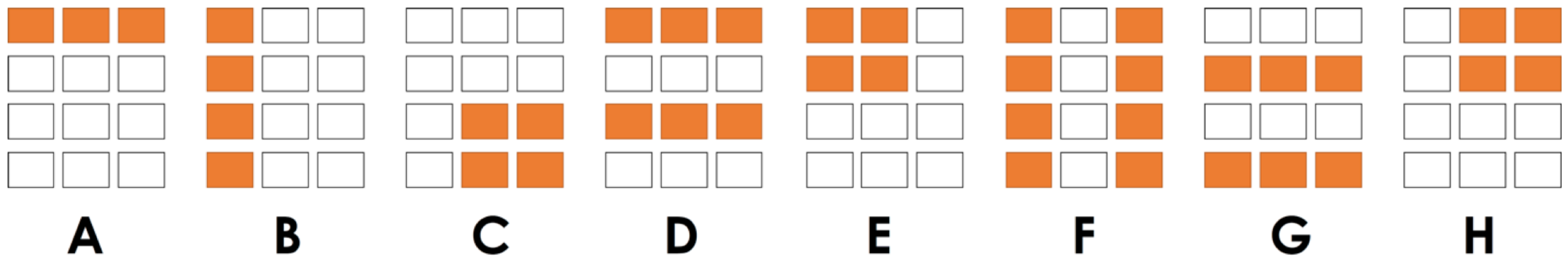
Python Code for Plotting



- Basic elements of a plot in matplotlib
  - plt.xlabel, plt.ylabel, …
- Be able to read basic plot code
- Review homeworks and lab on plotting

Numpy and Pandas

- Review basic slicing commands and Boolean indexing

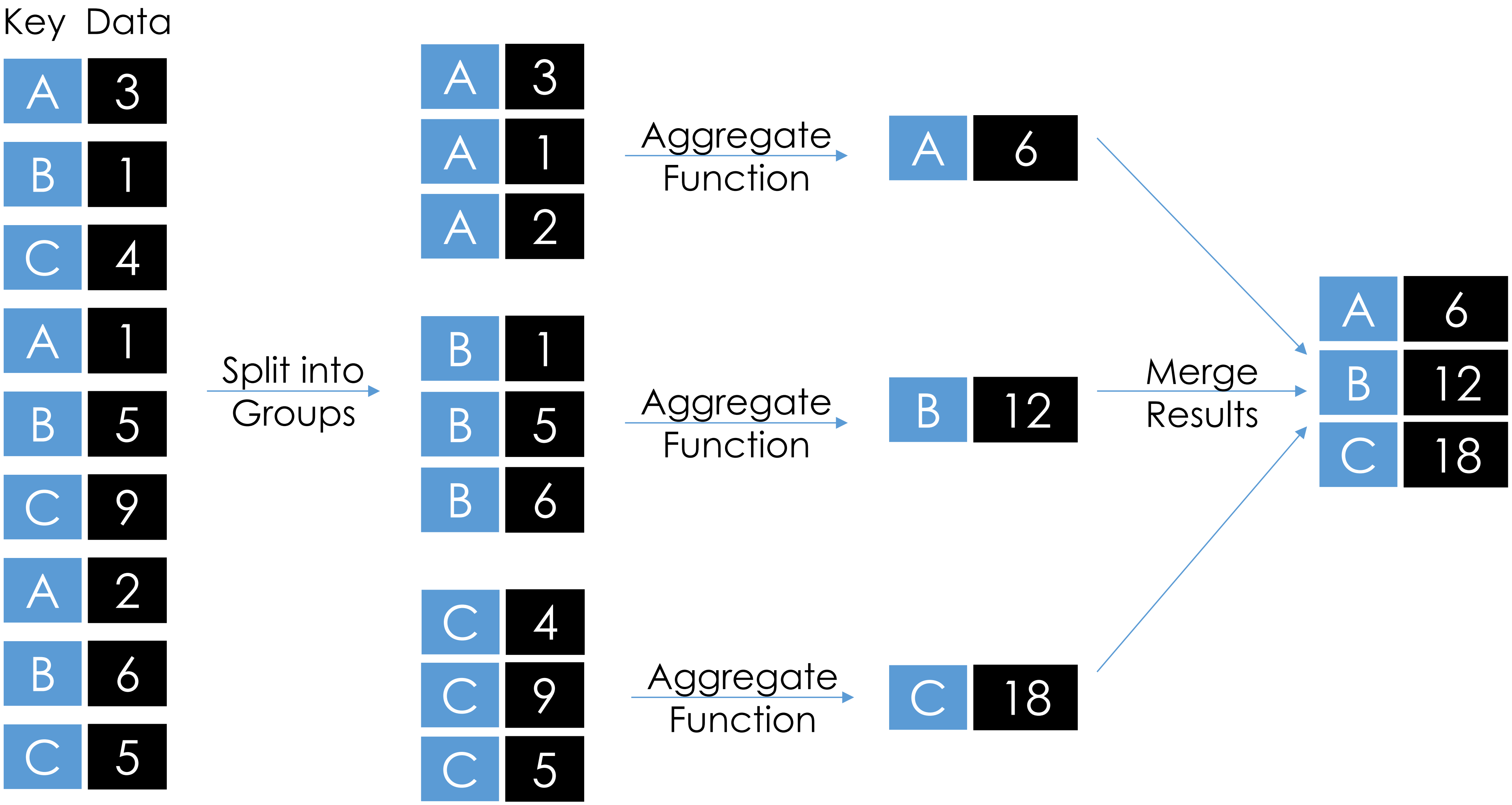


- df.loc[row names, cols names] (index lookup)
- df.iloc[row locations, column locations] (integer lookup)
- **Key functions:** *sum, mean, variance, arange*

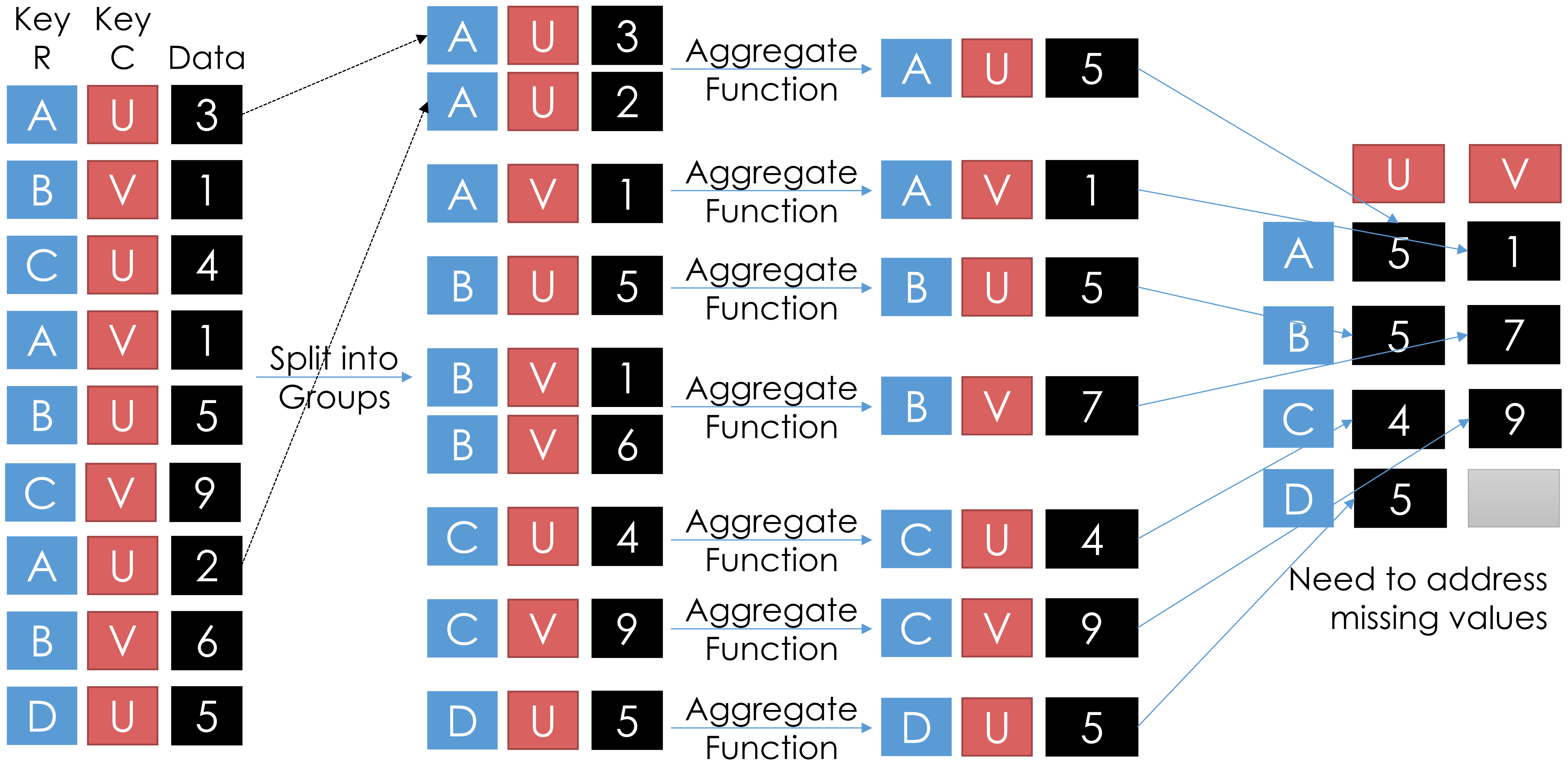
Pandas

- Review *column selection* and *Boolean slicing on rows*
- Review ***groupby***, **merge**, and **pivot_table**:
  - *df.groupby([‘state’, ‘gender’])[[‘age’, ‘height’]].mean()*
  - dfA.merge(dfB, on=‘key’, how=‘outer’)
  - df.pivot_table(index, columns, values, aggfunc, fill_value)
- Understand rough usage of basic plotting commands
  - plot, bar, histogram …
  - sns.distplot

Group By – manipulating granularity



Pivot – A kind of Group By Operation



Joining data across tables

- Joins are a way to connect data across multiple tables

Purchases.csv

OrderNum	ProdID	Quantity
1	**42**	3
1	**999**	2
2	**42**	1

$\bowtie$ Join

Products.csv

ProdID	Cost
42	3.14
999	2.72

=

Joined Table

OrderNum	ProdID	Quantity	Cost
1	**42**	3	3.14
1	**999**	2	2.72
2	**42**	1	3.14

SQL Coding

- You will not be required to write substantial amounts of SQL
  - Previous exams had harder (to grade) SQL questions
- You will need to read “interesting” SQL queries
  - WITH table_name AS (…) SELECT …
  - Interesting multiway joins
- You should be familiar with basic schema concepts
  - Data types
  - Foreign key relationships

Regular Expressions

- You will be given the regex guide on the practice midterm
- You should be able to construct regular expressions to match particular patterns
- You should be able to read regular expressions and determine what they match

Data Visualization

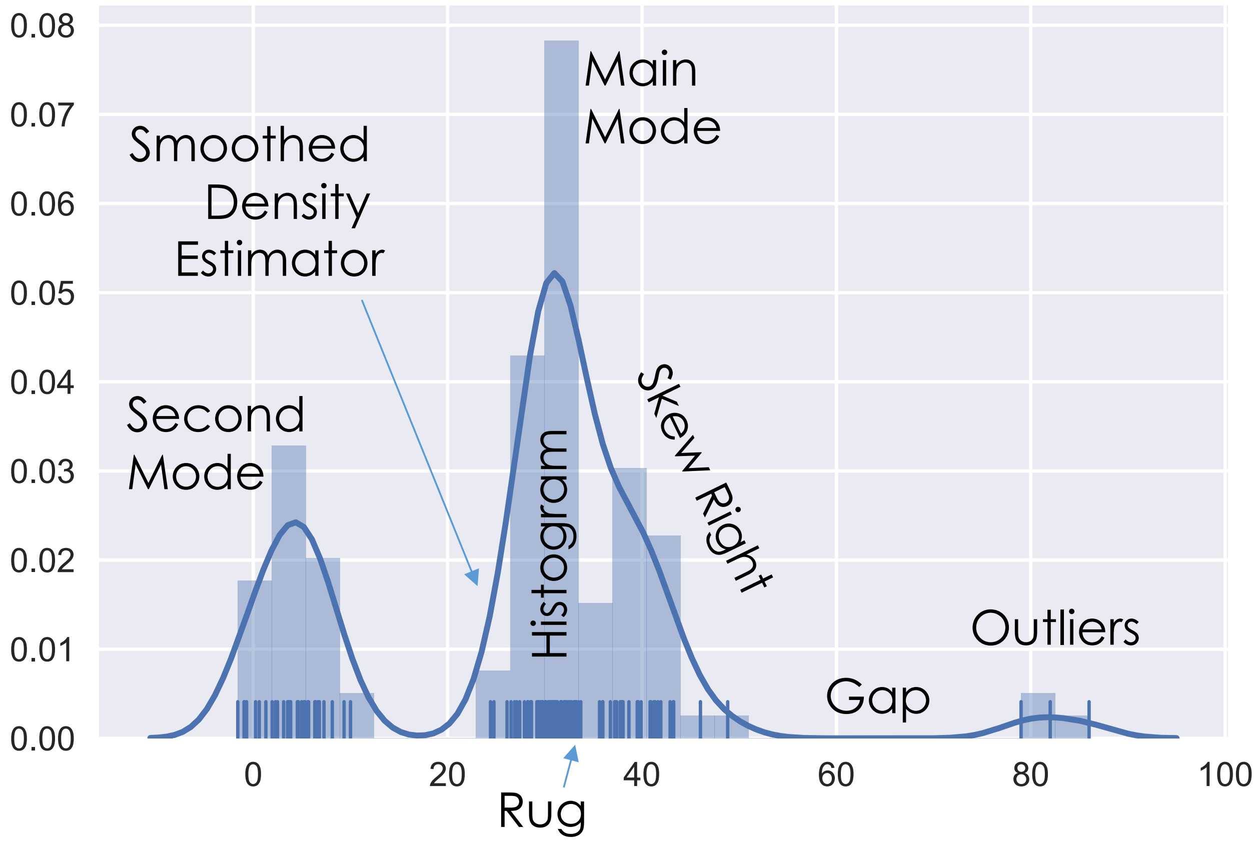
Visualizing Univariate Relationships

- **Quantitative Data**
  - Histograms, Box Plots, Rug Plots, Smoothed Interpolations (KDE – Kernel Density Estimators)
  - Look for symmetry, skew, spread, modes, gaps, outliers…
- **Nominal & Ordinal Data**
  - Bar plots (sorted by frequency or ordinal dimension)
  - Look for skew, frequent and rare categories, or invalid categories
  - Consider grouping categories and repeating analysis

Histograms, Rug Plots, and KDE Interpolation

Describes distribution of data – relative prevalence of values

- Histogram
  - relative frequency of values
  - Tradeoff of bin sizes
- Rug Plot
  - Shows the actual data locations
- Smoothed density estimator
  - Tradeoff of "bandwidth" parameter (more on this later)

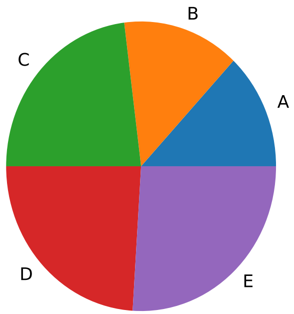


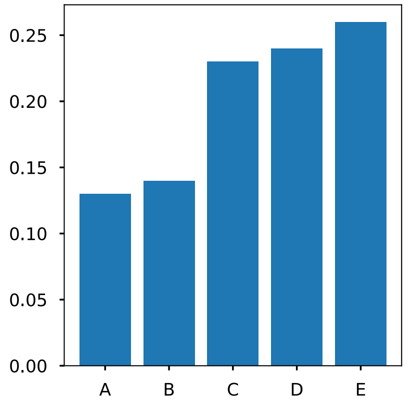
Techniques of Visualization

- **Scale:** ranges of values and how they are presented
  - Units, starting points, zoom, …
- **Conditioning:** breakdown visualization across dimensions for comparison (e.g., separate lines for males and females)
- **Perception**
  - **Length**: encode relative magnitude (best for comparison)
  - **Color**: encode conditioning and additional dimensions and
- **Transformations:** to linearize relationships highlight important trends
  - Symmetrize distribution
  - Linearize relationships (e.g., Tukey Mosteller Bulge)
- Things to avoid stacking, jiggling, chart junk, and over plotting

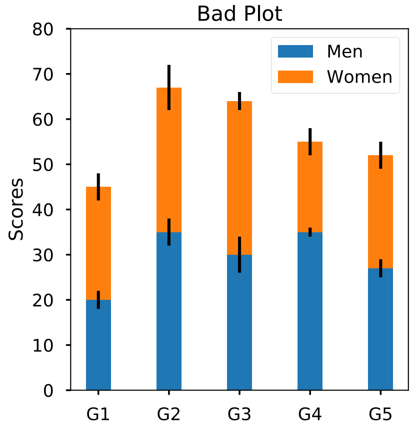
Bad Plot Terminology

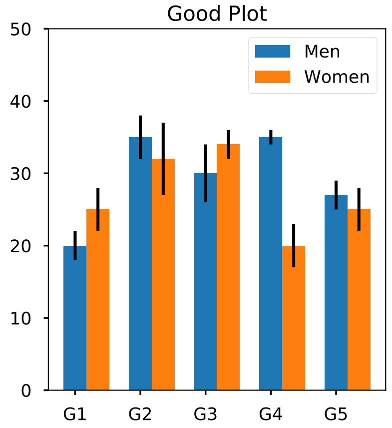
Pie charts → Bar charts



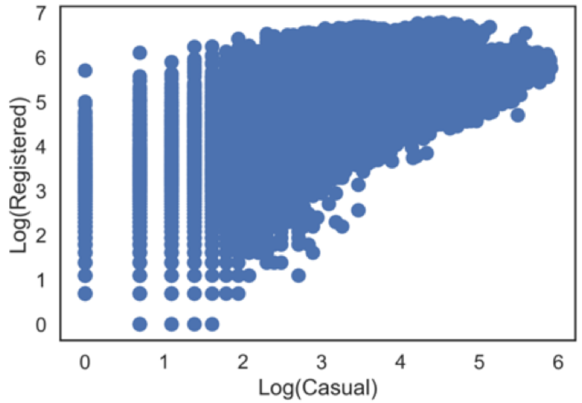


Eliminate Stacking and Jiggling





Over plotting



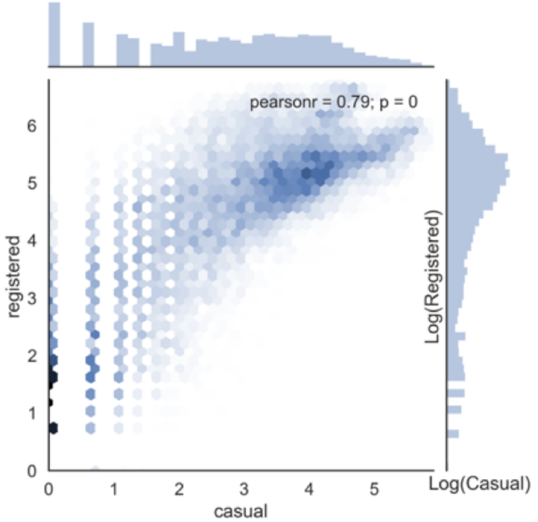
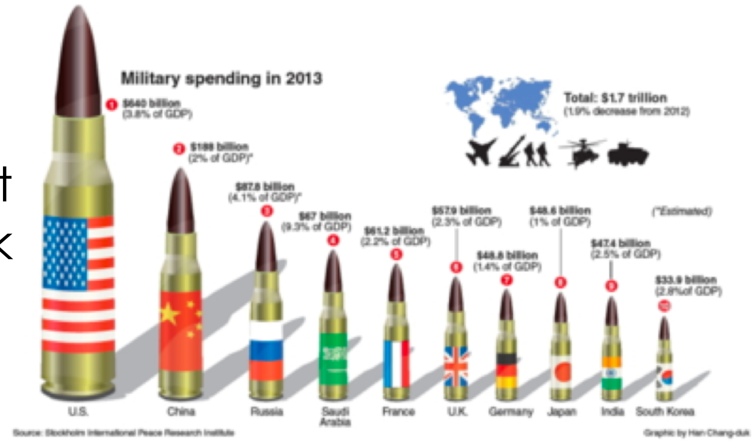
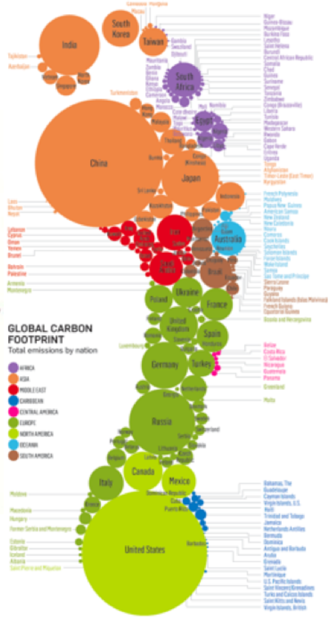


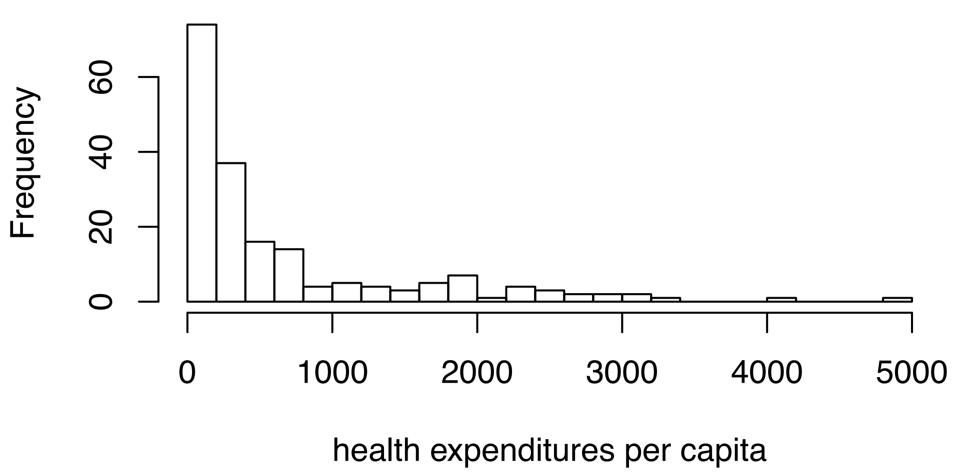
Chart Junk

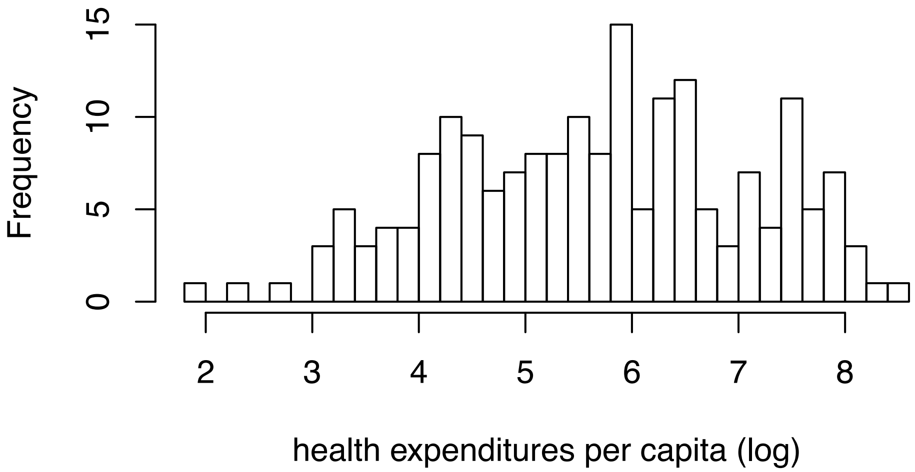


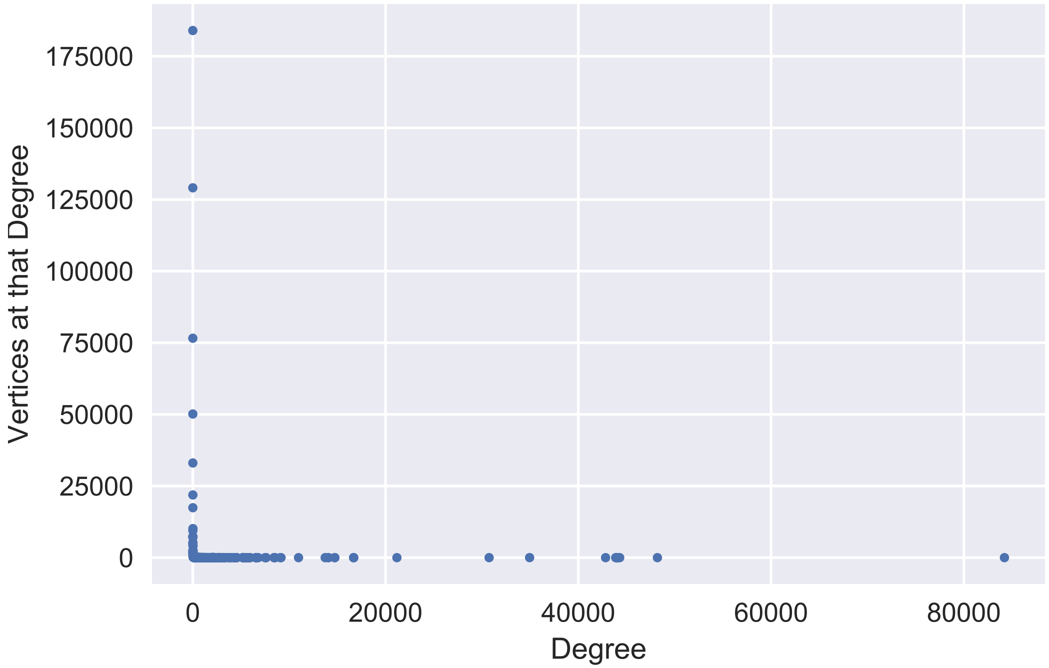


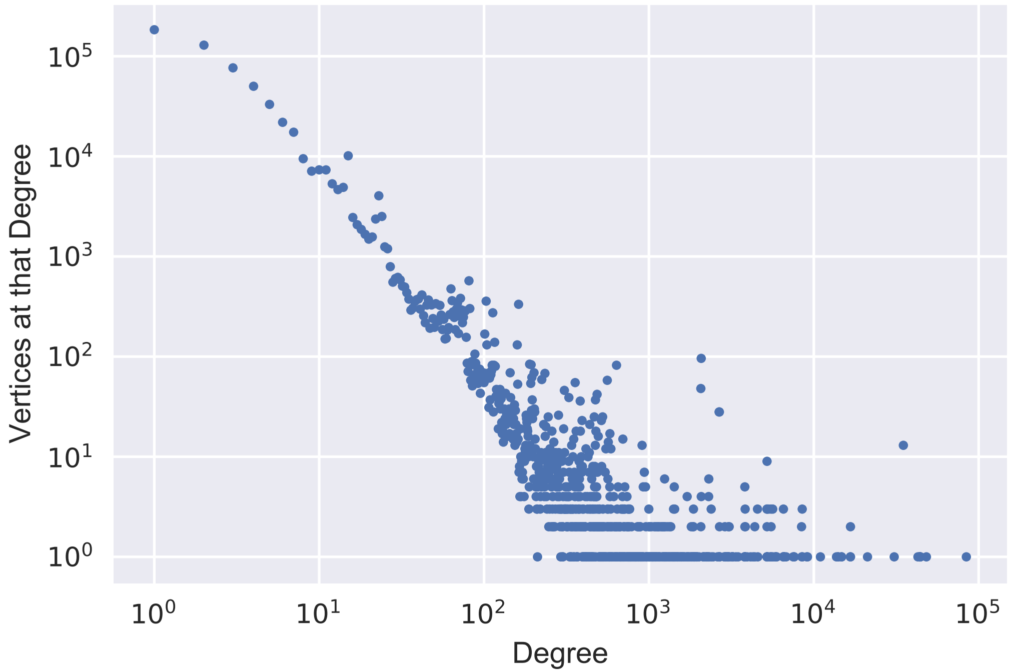
Area Perception + Chart Junk

Log Transformations

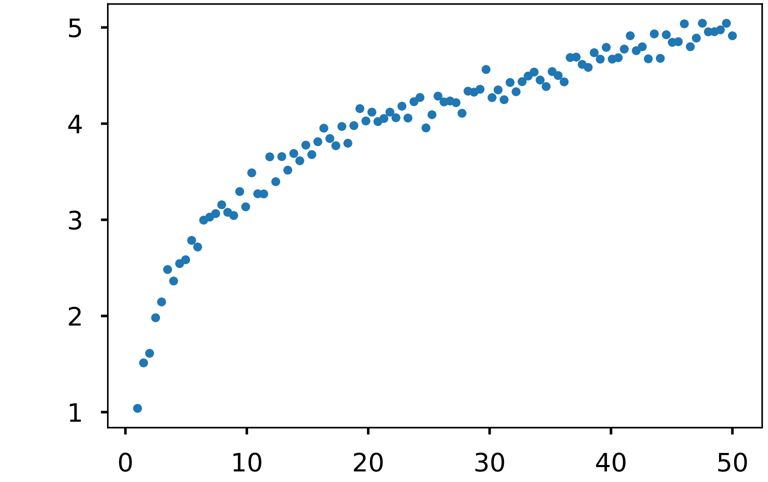


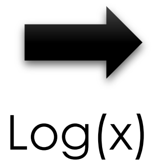


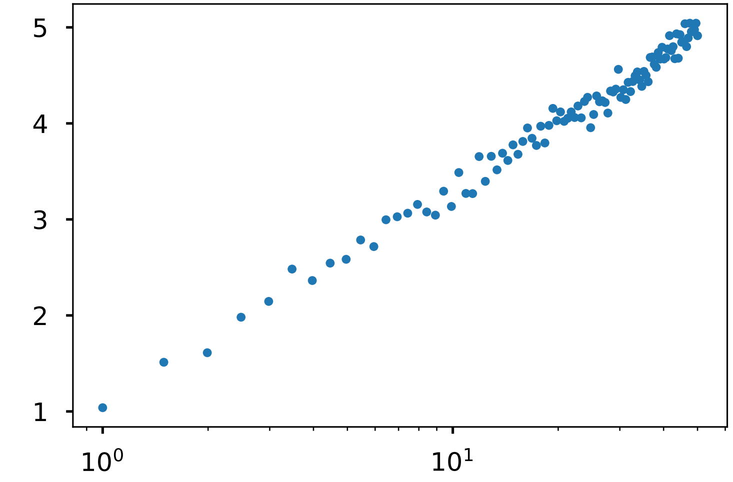


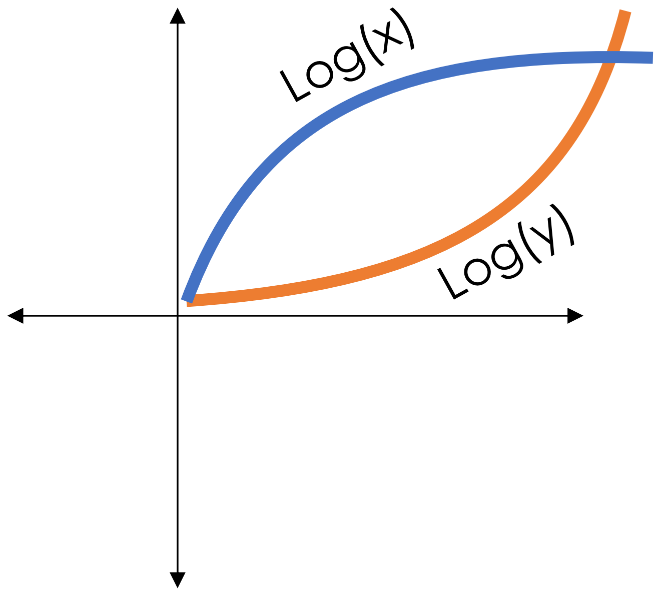


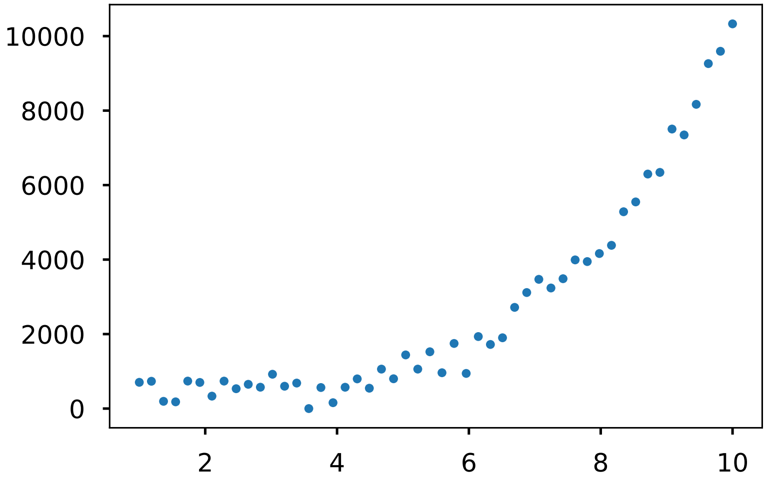
Linearizing Relationships

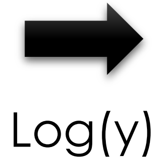


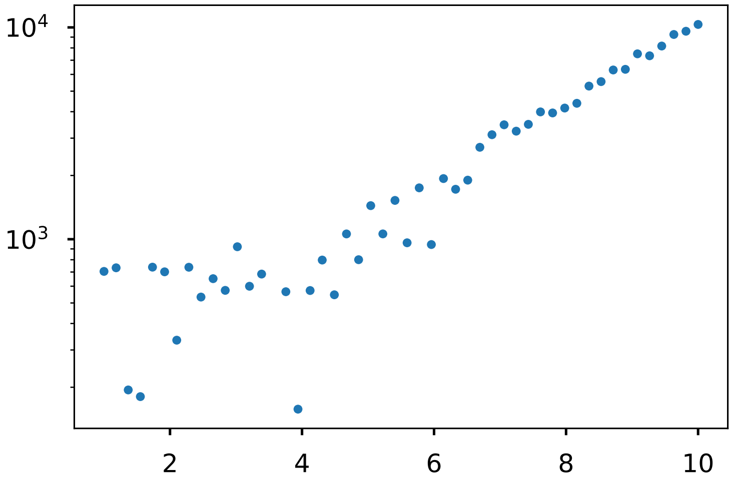










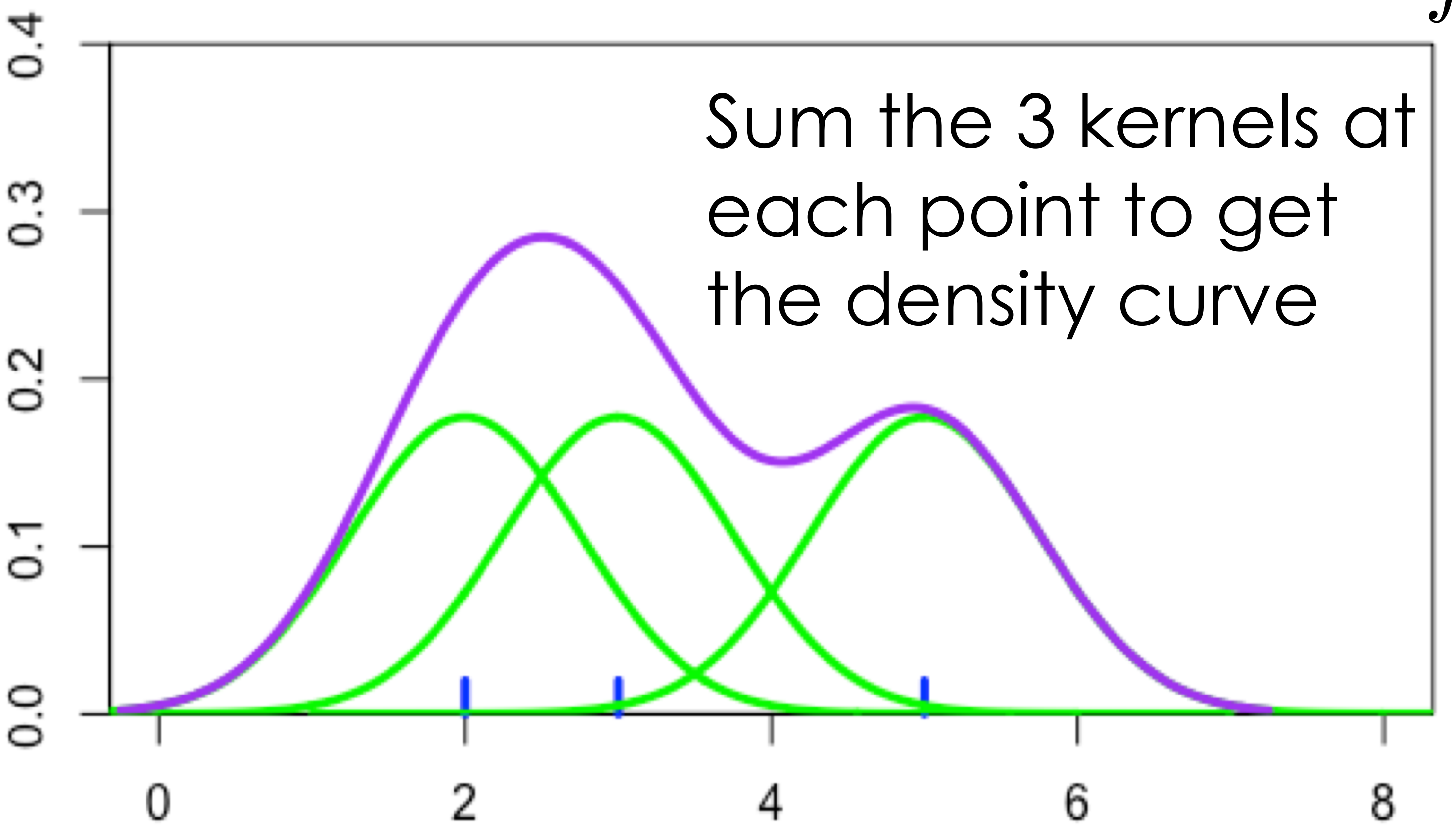


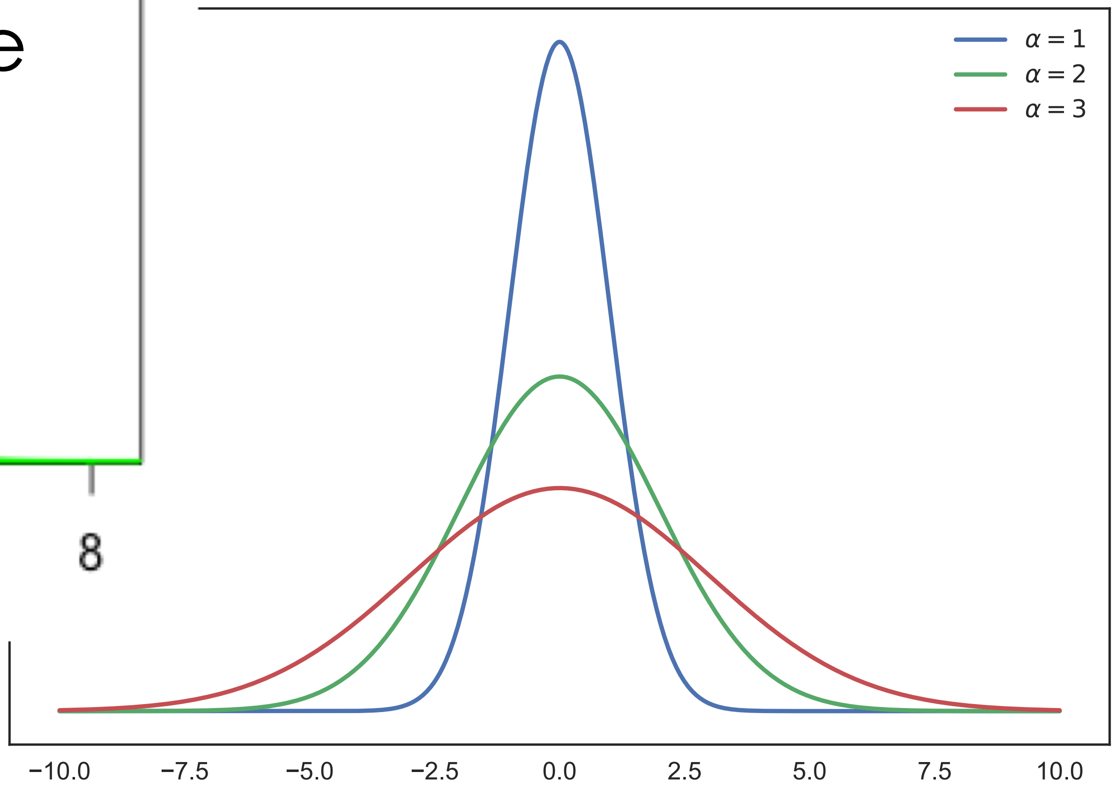
Dealing with Big Data

- **Big n** (many rows)
  - Aggregation & Smoothing – compute summaries over groups/regions
    - Sliding windows, kernel density smoothing
  - Set transparency or use contour plots to avoid over-plotting
- **Big p** (many columns)
  - Create new hybrid columns that summarize multiple columns
    - **Example**: total sources of revenue instead of revenue by product
  - Use dimensionality reduction techniques to automatically derive columns that preserve the relationships between records (e.g., distances)
    - PCA – not required to know PCA for the exam.

Kernel Density Estimator

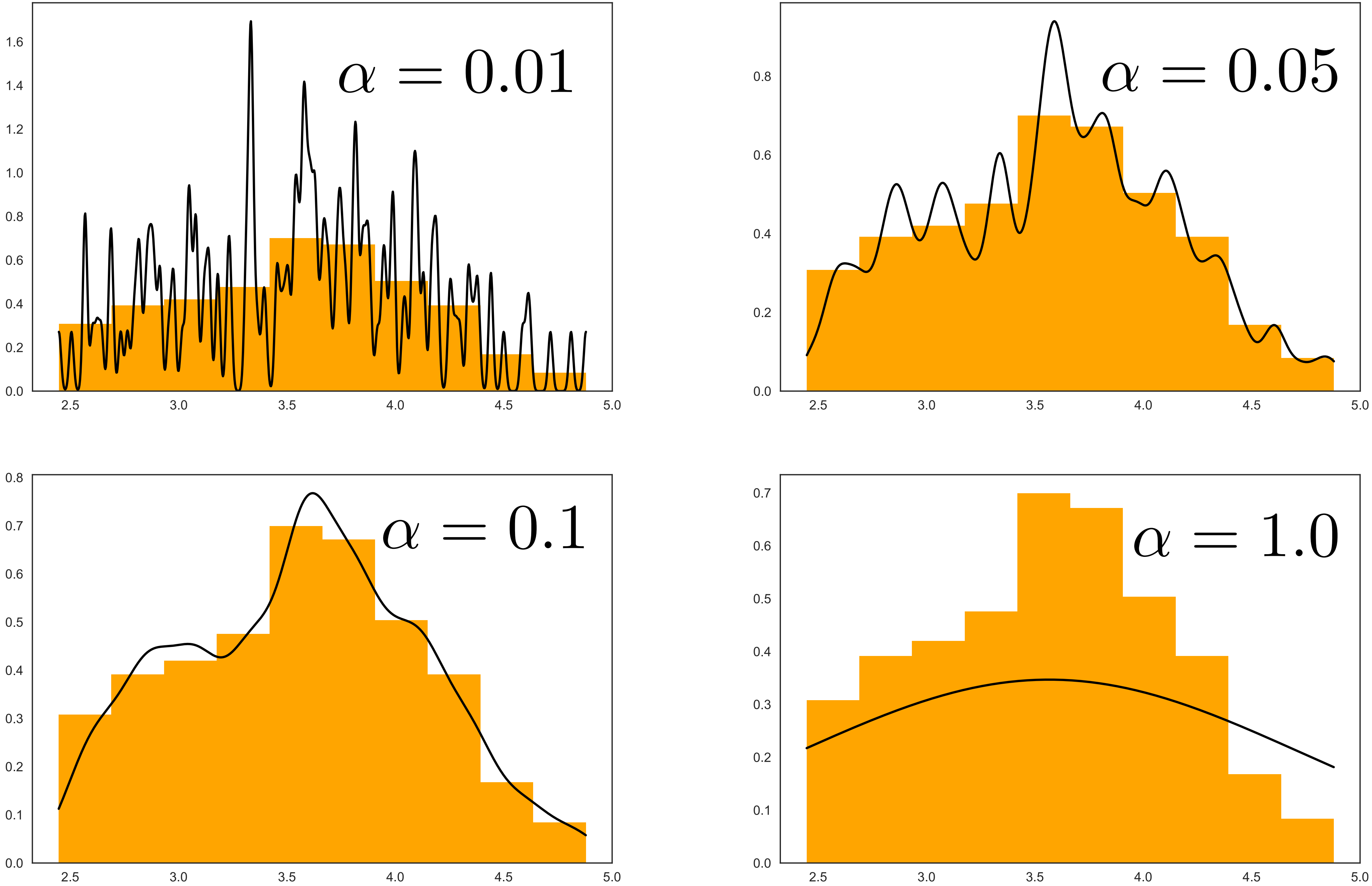
$$f(x) = \frac{1}{n}\sum_{i=1}^{n} K_\alpha(x - x_i)$$





$$K_\alpha(r) = \frac{1}{\sqrt{2\pi\alpha^2}}\exp\left(-\frac{r^2}{2\alpha^2}\right)$$

Gaussian Kernels


$\alpha = 0.01$
$\alpha = 0.05$
$\alpha = 0.1$
$\alpha = 1.0$
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
0.1
0.3
0.5
0.7
2.5
3.0
3.5
4.0
4.5
5.0

Sampling the Population

Data Collection and Sampling

- **Census:** the *complete* ***population of interest***
  - Important to identify the population of interest

Probability Samples:

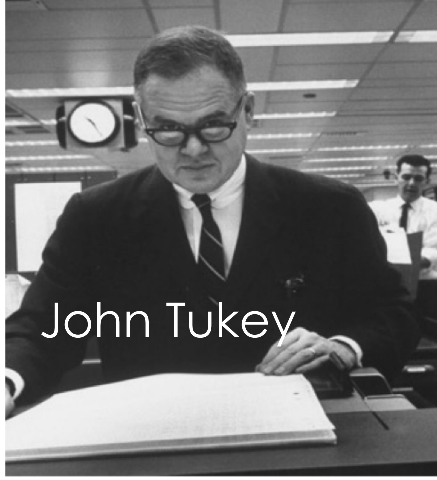
- **Simple Random Sample (SRS):** a random subset where every subset has equal chance of being chosen
- **Stratified Sample:** population is partition into strata and a SRS is taken within each strata
  - Samples from each strata don't need to be the same size
- **Cluster Sample:** divide population into groups, take an SRS of groups, and elements from each group are selected
  - Often take all elements (one-stage) may sample within groups (two-stage)

Non Probability Samples

- **Administrative Sample**: *data collected to support an administrative purpose and not for research*
  - Bigger isn't always better → bias still an issue at scale
- **Voluntary Sample:** self-selected participation
  - Sensitive to self selection bias
- **Convenience Sample:** the data you have …
  - often administrative

Data Cleaning and EDA

Exploratory Data Analysis

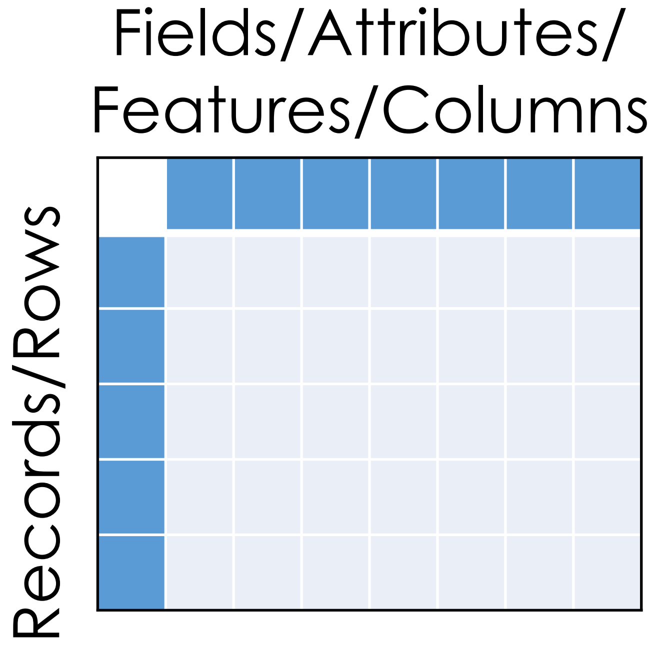


- Goals of EDA
  - **Validate** the **data collection** and preparation
  - **Confirm understanding** of the data
  - Search for **anomalies** or where data is **surprising**
- Iterative Exploratory Process
  - Analyze **summary statistics** and **data distributions**
  - **Transform** and **analyze relationships** between variables
  - **Segment data** across informative dimensions (granularity)
  - Use <u>**visualizations**</u> to build a deeper understanding

Key Data Properties to Consider in EDA

- **Structure --** *the "shape" of a data file*
- **Granularity --** *how fine/coarse is each datum*
- **Scope --** *how (in)complete is the data*
- **Temporality --** *how is the data situated in time*
- **Faithfulness --** *how well does the data capture "reality"*

Rectangular Structure



We prefer rectangular data for data analysis (why?)

- Regular structures are easy manipulate and analyze
- A big part of data cleaning is about transforming data to be more rectangular

Two main variants

1. Tables (a.k.a. data-frames in R/Python and relations in SQL)
    - Named columns with different types
    - Manipulated using data transformation languages
        - map, filter, group by, join, project,
2. Matrices
    - Numeric data of the same type
    - Manipulated using linear algebra

Kinds of Data

Note that categorical data can also be numbers and quantitative data may be stored as strings.

Quantitative Data

Numbers with meaning ratios or intervals.

Examples:

- Price
- Quantity
- Temperature
- Date
- …

Categorical Data

Ordinal

Categories with orders but no consistent meaning if magnitudes or intervals

Examples:

- Preferences
- Level of education
- …

Nominal

Categories with no specific ordering.

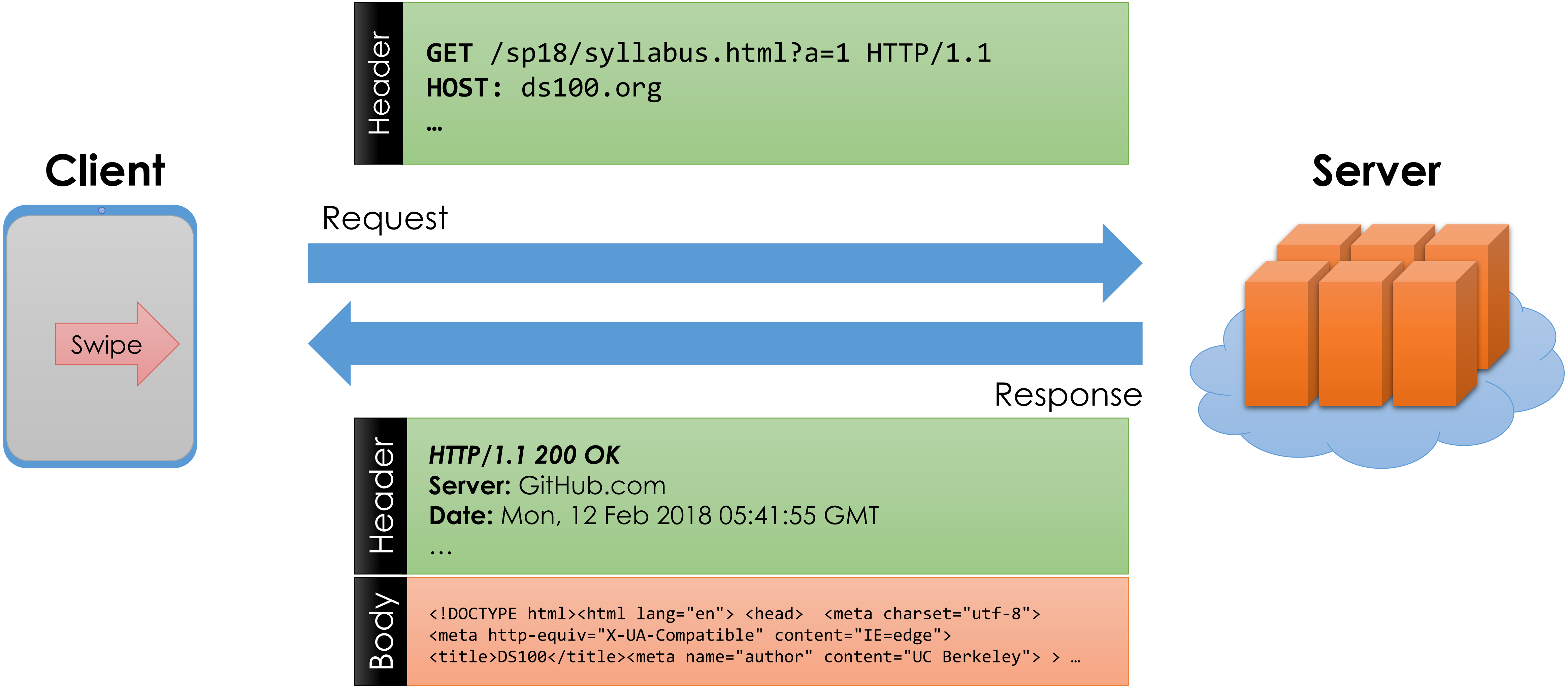
Examples:

- Political Affiliation
- CalD number
- …

Web Technologies

XML/JSON/HTTP/REST

Request – Response Protocol



Request Types (Main Types)

- Know differences between put and get
- **GET** – *get information*
  - Parameters passed in URI (limited to ~2000 characters)
    - `/app/user_info.json?username=mejoeyg&version=now`
    - Request body is typically ignored
  - Should not have side-effects (e.g., update user info)
  - Can be cached in on server, network, or in browser (bookmarks)
- **POST** – *send information*
  - Parameters passed in URI and BODY
  - May and typically will have side-effects
  - Often used with web forms.

HTML/XML/JSON

- Most services will exchange data in HTML, XML, or JSON
- Nested data formats (review JSON notebook)
  - Understand how JSON objects map to python objects (HWs)
    - JSON List → Python List
    - JSON Dictionary → Python Dictionary
    - JSON Literal → Python Literal
- Review basic XML formatting requirements:
  - Well nested tags, no spaces, case sensitive,
- Be able to read XML and JSON and identify basic bugs

String Manipulation and Regular Expressions

Regex Reference Sheet

^ match beginning of string (unless used for negation [^ …])

$ match end of string character

? match preceding character or subexpression at most once

+ match preceding character or subexpression one or more times

***** match preceding character or subexpression zero or more times

. matches any character **except newline**

[ ] match any single character inside
- match a range of characters [a-c]

( ) used to create sub-expressions

\b match boundary between words

\w match a "word" character (letters, digits, underscore). **\W** is the complement

\s match a whitespace character including tabs and newlines. **\S** is the complement

\d match a digit. **\D** is the complement

You should know these.

Greedy Matching

- **Greedy matching:** * and **+** match as many characters as possible using the preceding subexpression in the regular expression before going to the next subexpression.
- Example
  - <.*> matches <body>text</body>
- ? The modifier suffix makes * and **+** non-greedy.
  - <.*?> matches <body>text</body>

SQL

Relational Database Management Systems

- Traditionally DBMS referred to relational databases

ORACLE MySQL TERADATA Microsoft SQL Server PostgreSQL SQLite

- ***Logically*** organize data in **relations** (tables)

Sales relation:

Name	Prod	Price
Sue	iPod	$200.00
Joey	Bike	$333.99
Alice	Car	$999.00

Tuple (row)

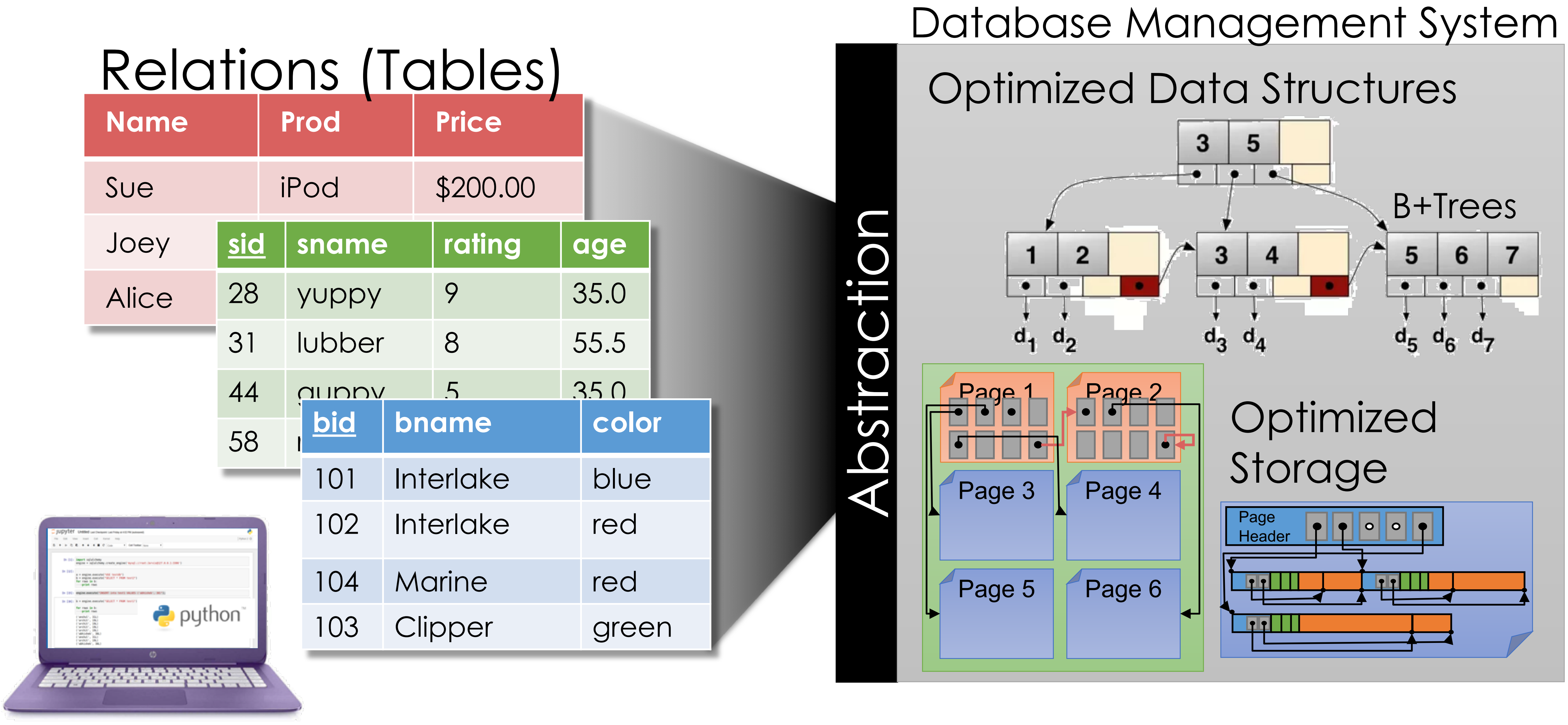
Attribute (column)

Describes *<u>relationship:</u>*

Name *purchased* ***Prod*** *at* ***Price.***

How is data **physically** stored?

Relational Data **Abstraction**



Physical Data Independence:

Database management systems **hide how data is stored** from end user applications

→ System can **optimize storage** and **computation** without changing applications.

Big Idea in Data Structures
Data Systems & Computer Science

It wasn't always like this …

SQL is **Declarative:**

SQL Keywords

What I want	`SELECT name, gpa`
From what source	`FROM students`
Under what conditions	`WHERE dept = 'CS'`
How should it be presented	`ORDER BY gpa`

Say ***what*** you want, not ***how*** to get it.

Relational Terminology

- *Database*: Set of Relations (i.e., one or more tables)
- *Attribute* (*Column*)
- *Tuple* (*Record, Row*)
- *Relation* (*Table*):
  - *Schema*: the set of column names, their types, and any constraints
  - *Instance*: data satisfying the schema
- *Schema of database* is set of schemas of its relations

Keys to Connect Data

- Often data will reference other pieces of data
- **Primary key:** *the column or set of columns in a table that determine the values of the remaining columns*
  - **Primary keys are unique**
  - Examples: SSN, ProductIDs, …
- **Foreign keys:** the column or sets of columns that reference primary keys in other tables.

Purchases.csv

OrderNum	ProdID	Quantity
1	42	3
1	999	2
2	42	1

Foreign Key

Orders.csv

OrderNum	CustID	Date
1	171345	8/21/2017
2	281139	8/30/2017

Products.csv

ProdID	Cost
42	3.14
999	2.72

Primary Key

Customers.csv

CustID	Addr
171345	Harmon..
281139	Main ..

The Data Definition Language

sid	sname	rating	age
1	Fred	7	22
2	Jim	2	39
3	Nancy	8	27

bid	bname	color
101	Nina	red
102	Pinta	blue
103	Santa Maria	red

sid	**bid**	**day**
1	102	9/12
2	102	9/13

```
CREATE TABLE Sailors (
   sid INTEGER,
   sname CHAR(20),
   rating INTEGER,
   age REAL,
   PRIMARY KEY (sid));
```

Columns have **names** and **types**

Specify **Primary Key** column(s)

```
CREATE TABLE Boats (
   bid INTEGER,
   bname CHAR (20),
   color CHAR(10),
   PRIMARY KEY (bid));
```

Specify **Foreign Key** relationships

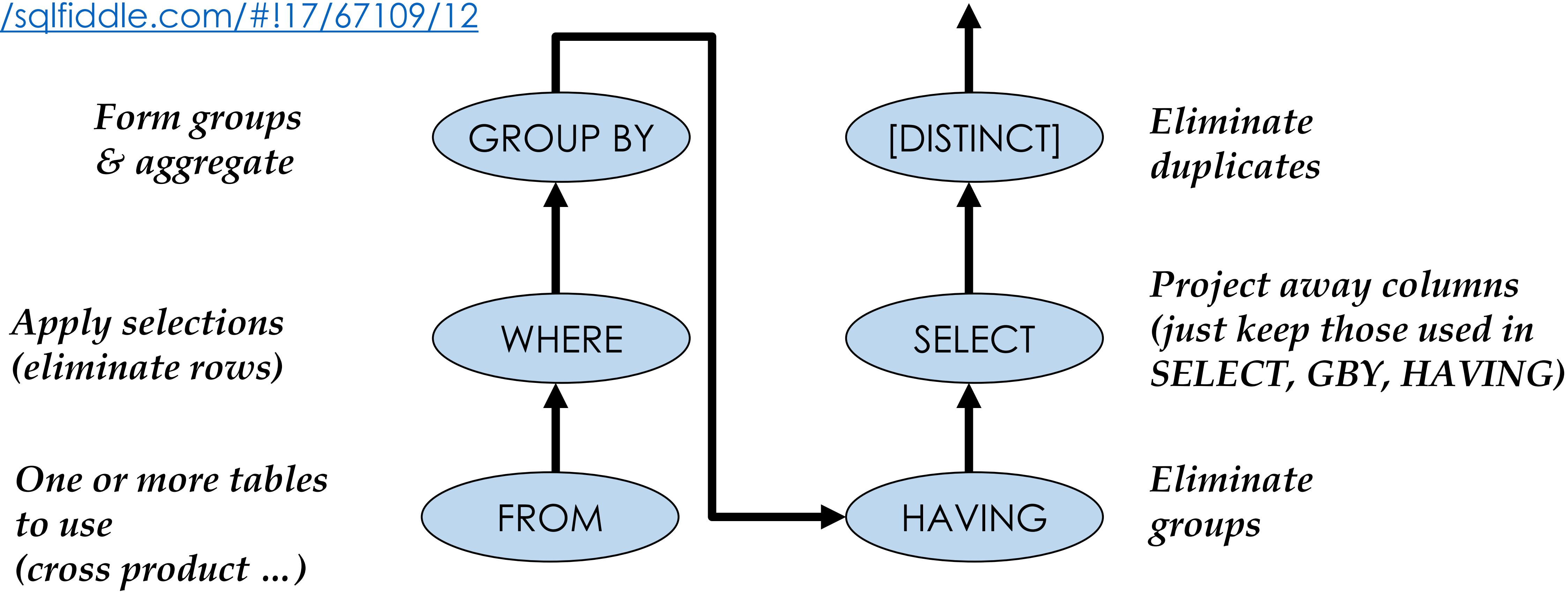
```
 CREATE TABLE Reserves (
   sid INTEGER,
   bid INTEGER,
   day DATE,
  PRIMARY KEY (sid, bid, day),
  FOREIGN KEY (sid) REFERENCES Sailors,
  FOREIGN KEY (bid) REFERENCES Boats);
```

Semicolon at end of commands

Conceptual SQL Evaluation

SELECT [DISTINCT] *target-list*
FROM *relation-list*
WHERE *qualification*
GROUP BY *grouping-list*
HAVING *group-qualification*

Try Queries Here
http://sqlfiddle.com/#!17/67109/12



Join Queries

```
SELECT [DISTINCT] <column expression list>
  FROM <table1 [AS t1], ... , tableN [AS tn]>
[WHERE <predicate>]
[GROUP BY <column list>
 [HAVING <predicate>] ]
[ORDER BY <column list>];
```

1. FROM : compute ***cross product*** of tables.
2. WHERE : Check conditions, discard tuples that fail.
3. SELECT : Specify desired fields in output.

- Note: likely a terribly inefficient strategy!
  - Query optimizer will find more efficient plans.

Return Sailors (S) and the dates of their Reservations (R)

```sql
SELECT S.sname, R.day
FROM Reserves AS R, Sailors AS S
WHERE S.sid = R.sid
```

Symbol for join
(Rel. Alg.)

$$R1 \bowtie S1$$

R:

sid	bid	day
22	101	10/10/96
58	103	11/12/96

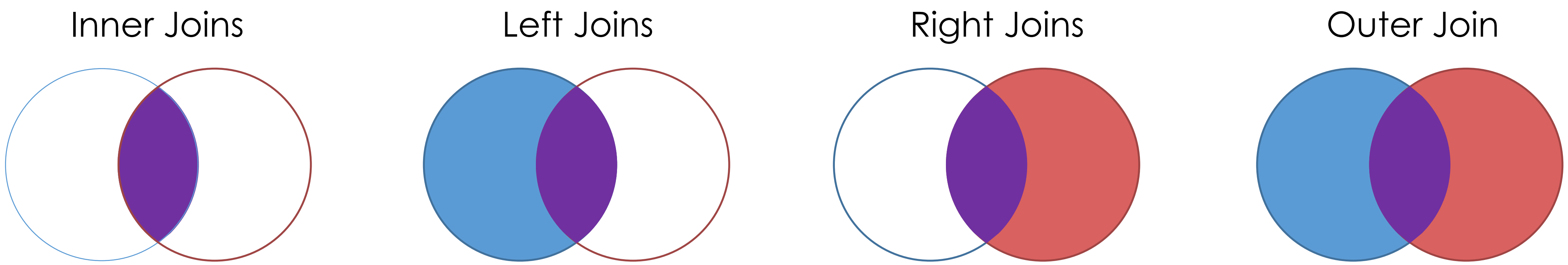
S:

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

sid	bid	day	sid	sname	rating	age
22	101	10/10/96	22	dustin	7	45.0
~~22~~	~~101~~	~~10/10/96~~	~~31~~	~~lubber~~	~~8~~	~~55.5~~
~~22~~	~~101~~	~~10/10/96~~	~~58~~	~~rusty~~	~~10~~	~~35.0~~
~~58~~	~~103~~	~~11/12/96~~	~~22~~	~~dustin~~	~~7~~	~~45.0~~
~~58~~	~~103~~	~~11/12/96~~	~~31~~	~~lubber~~	~~8~~	~~55.5~~
58	103	11/12/96	58	rusty	10	35.0

http://sqlfiddle.com/#!17/53815/1140/0

Kinds of Joins



Review the slides and syntax for each join type

```
SELECT r.sid, b.bid, b.bname
FROM Reserves3 r FULL JOIN Boats2 b
ON r.bid = b.bid
```

Reserves3

sid	bid	day
22	101	1996-10-10
95	103	1996-11-12
38	42	2010-08-21

Boats2

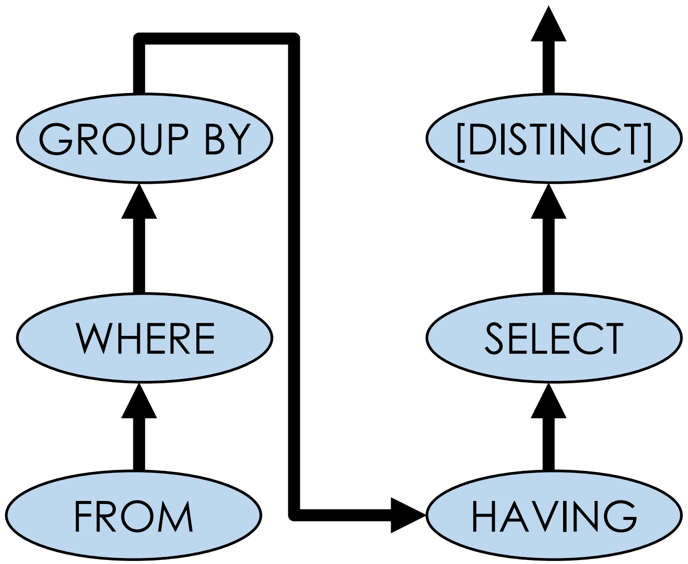
bid	bname	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Result:

sid	bid	bname
22	101	Interlake
95	103	Clipper
38	**(null)**	**(null)**
(null)	104	Marine
(null)	102	Interlake

Putting it all together

```
SELECT c.name, AVG(g.grade) AS avg_g, COUNT(*) AS size
  FROM grades AS g, classes AS c
 WHERE g.class_id = c.class_id AND
       g.year = "2006"
 GROUP BY g.class_id
HAVING COUNT(*) > 2
 ORDER BY avg_g DESC
```



What does this compute?

Good Luck!