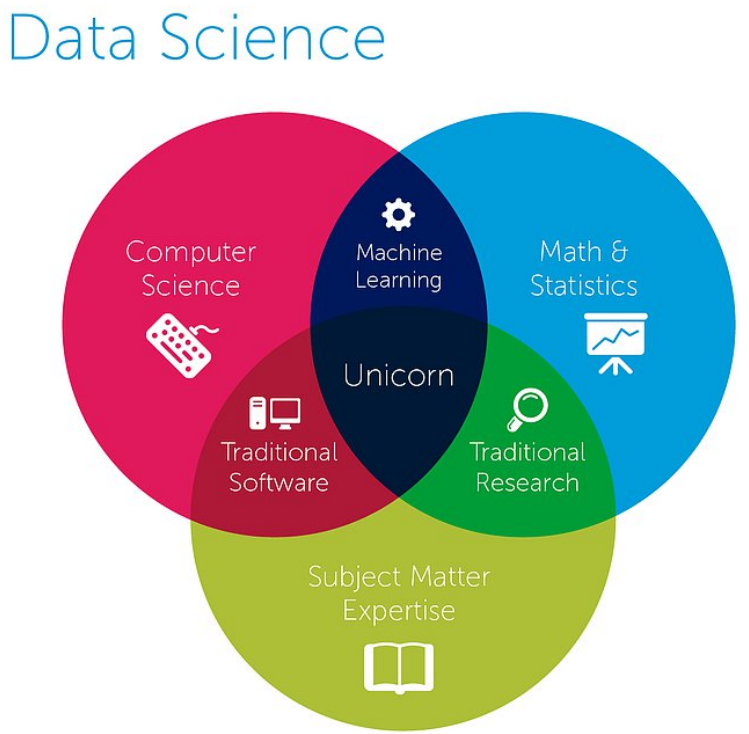
Introduction to Statistical Learning

36-600

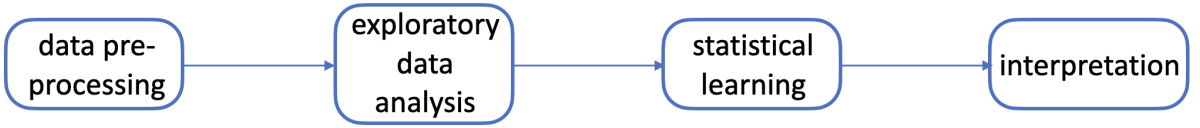
Week 1 Tuesday – Fall 2021

Data Science: The Search for the Unicorn



(Credit: Computer Science Department, Luther College)

The Canonical Analysis Workflow



- *Data Pre-Processing*: the act of extracting analyzable data (e.g., a structured data table) from unstructured sources (e.g., images, audio files, text, etc.), as well as the act of editing the data table to mitigate missing data.
- *Exploratory Data Analysis*: the act of visualizing the observed data--via, e.g., histograms, scatter plots, box plots, etc., etc.--so as to build intuition about them. No statistical modeling is involved. EDA is not a substitute for statistical modeling, due to its implicit dimensional reduction!
- *Statistical Learning*: the attempt to find meaningful structures in the data or to uncover relationships between elements of the dataset. More on this below.
- *Interpretation*: what did you discover through your analysis?

Our primary focus in this course is **statistical learning**. We will touch on EDA, and perhaps pre-processing, but the majority of the time will be spent contextualizing the world of statistical learning.

Some Initial Statistical Learning Terminology

Let's assume you have a table of data.

- The rows of this table represent individual *observations* (e.g., people, objects); your *sample size* n is thus the number of rows.
- The columns of this table represent *measurements* taken for each *observation* (e.g., height, weight, favorite ice cream flavor); let the number of columns be $p + 1$. (Why "+1"? This will make a bit more sense below.)

So, let's try this statistical learning thing again.

- *Unsupervised learning*: every observation is a point in a $p + 1$-dimensional space, and your goal is to segment these points and then ascribe meaning to the segments. Examples of unsupervised learning algorithms include k-means clustering and the expectation-maximization (EM) algorithm.
- *Supervised learning*: here, every observation can be thought of as a point in a p-dimensional space (let's call this $\mathbf{x}$) that is paired with the value in the left-out column (which we'll call y). Then your goal is to uncover a statistically significant functional relationship between $\mathbf{x}$ and y, i.e., $y = f(\mathbf{x})$ + random error. Examples of supervised learning algorithms include (multiple) linear regression and decision trees.

Supervised Learning

"Then your goal is to uncover a statistically significant functional relationship..."

This "functional relationship" is termed a *statistical model*. This is a mathematical model, possibly encompassing a set of assumptions, that describes the data-generating process.

To be clear: linear regression is a statistical model. `lm()` in `R` or `LinearRegression` in `Python` are *not* statistical models; they are functions you use to learn the models, given input data (i.e., to estimate the values of the linear regression coefficients, the intercept and the slopes). For too many, this distinction is, unfortunately, not clear.

Some more terminology:

- The *variables* that comprise the set of measurements $\mathbf{x}$ are dubbed *predictor* variables, *independent* variables, or *explanatory* variables. (I tend to use "predictor variables," but analysts in your domain may use another term. Always be flexible.)
- The variable y is the *response* variable or the *dependent* variable. (I tend towards "response.") In this class, we will assume that there is only one, although depending on the model there can be more than one.

Where is Machine Learning in All of This?

OK, slow down.

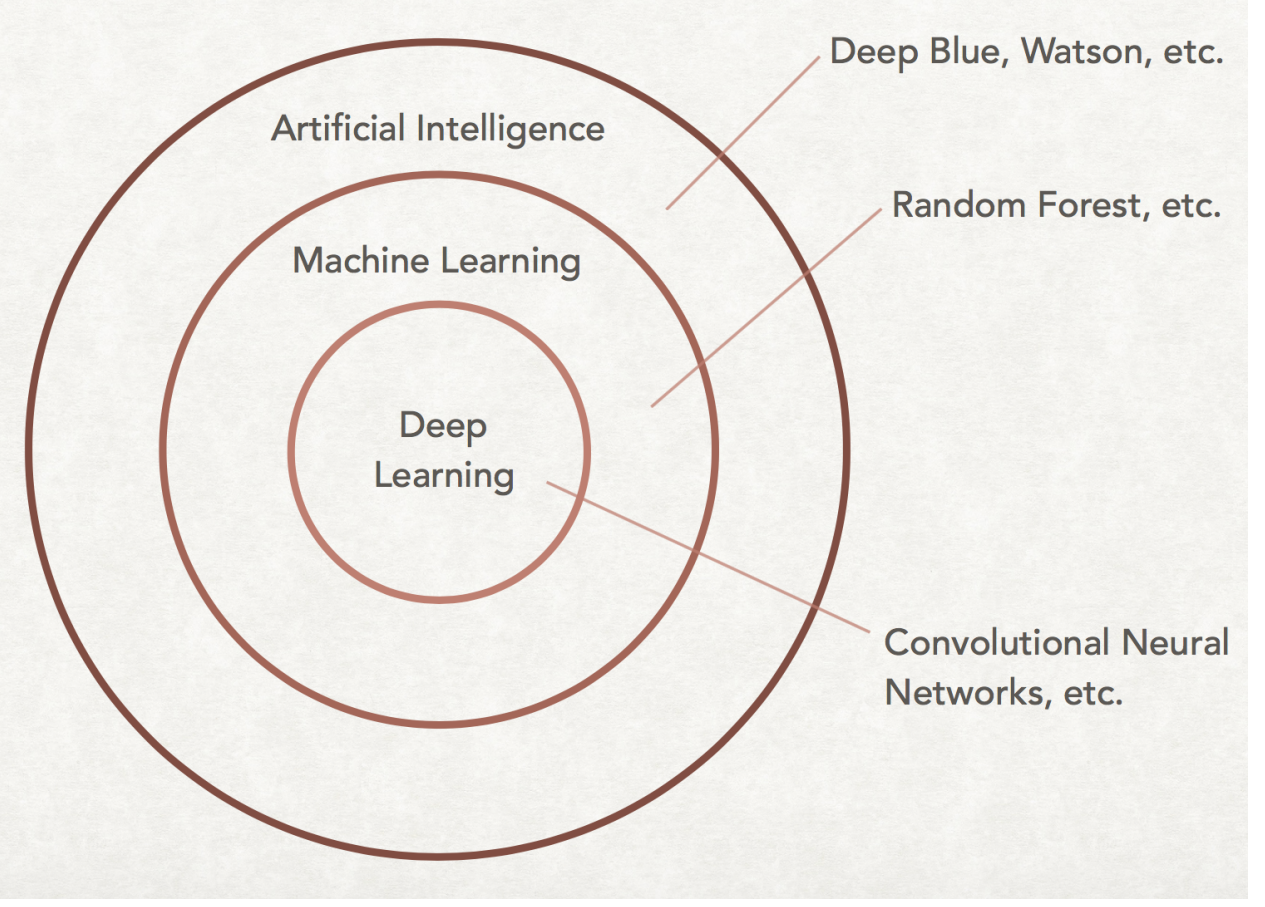
The basic idea is that you can either specify the parametric form of a supervised learning model (such as you can for linear regression or any curve-fitting exercise) or you can let the computer construct the model for you, given the data and some algorithmic rules. Computer-constructed models are *machine learning* models. Examples of machine learning models include decision trees, random forest, extreme-gradient boosting, support vector machine, and the various flavors of deep learning.

Why are there multiple ML models? Because, depending on how your predictor variables are distributed in their native p-dimensional space, some ML models will be better at relating $\mathbf{x}$ and y. Full stop. You just need to figure out which one is best for your particular data. However...

One overarching thing to keep in mind: if the relationship between $\mathbf{x}$ and y is truly linear, then linear regression will be your best model and ML will not help you. **Don't be an ML snob!** Always try linear regression and do not simply start with deep learning because some blog told you to do so.

Are We Gonna Do AI?

Yes.



Words, Words, Words...I Want an Example

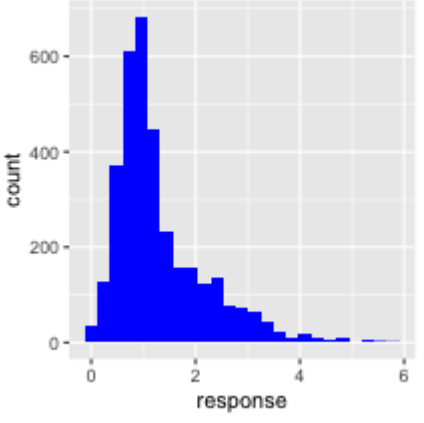
OK.

```
load(url("https://raw.githubusercontent.com/pefreeman/36-290/master/EXAMPLE_DATASETS/PHOTO_MORPH/photo_morph.Rdata"))
dim(predictors)
```

```
## [1] 3419   16
```

These data are comprised of measurements made for 3,419 galaxies: 16 predictor variables and 1 response variable. (The predictors are brightnesses and appearance statistics at different wavelengths. The response is "redshift," which you can think of as "distance from the Sun.")

What is the distribution of the response variable? We can show that with a histogram:



(You will learn how to construct a histogram like this in R in a future lecture or lab.)

Example, Part 2

The raw predictor data table looks something like this:

mag.i	col.Vi	col.iJ	col.JH	V.G	V.M20	V.C	V.size	J.G	J.M20	J.C	J.size	H.G	H.M20	H.C	H.size
-2.688712287	1.6003916	1.0945781	0.43445239	0.4669932	-2.0098914	3.657756	0.8550404	0.5153533	-2.0778946	3.800239	0.9282933	0.5062522	-2.035148	3.680959	0.8597183
0.540569680	1.4697780	1.5372292	0.56113018	0.3884504	-1.4596036	2.782257	0.6581460	0.3984545	-1.7358275	3.522715	0.5853450	0.4446404	-1.814633	3.495193	0.6149408
-1.189388561	1.2384926	1.1782910	0.45021207	0.3955888	-1.4006042	2.968765	0.9121046	0.3323260	-1.2742901	2.473426	0.8298767	0.4071918	-1.524523	3.259704	0.8696576
1.739926810	1.7536618	2.5048038	0.65331841	0.2497647	-1.4437609	2.760286	0.3385138	0.4508824	-1.6229520	3.211855	0.4062165	0.4426040	-1.677055	3.322800	0.3918580
-1.086210681	0.3873798	0.6925946	0.27716911	0.4764610	-1.0978497	2.494967	0.7188935	0.4082313	-1.0023375	2.462863	0.8131379	0.4509829	-1.211500	2.737906	0.9276759
-5.704781087	0.7537113	0.7743299	0.33452462	0.3558670	-1.6328994	3.016005	0.8768748	0.4875368	-2.0352400	3.824191	1.3027283	0.4842033	-1.999343	3.764287	1.2096835

What do we see?

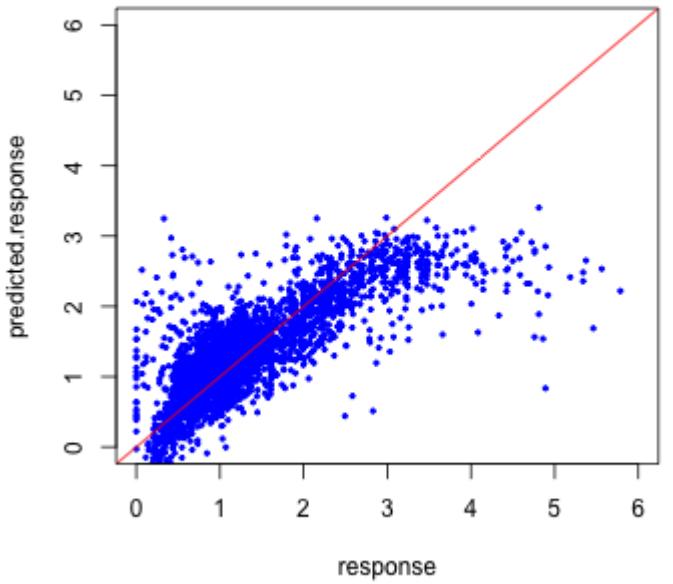
- *Every cell has a value in it*. This is not always the case; one bane of analysts is *missing data*. Statistical learning algorithms cannot run with missing data, so dealing with them (moot in this case) is part of pre-processing.
- *Every cell has a quantitative value in it*. This also would not always be the case in general. Sometimes, a measurement is represented by a so-called *factor* variable that has a number of discrete, generally non-quantitative values. ("What's your favorite ice cream flavor: chocolate or vanilla?")

What would we do with this?

- As we will see in future lectures, we can make histograms, box plots, scatter plots, assess correlations, etc. Or make six-figure summaries or tables or... We'll get back to this.
- We might then try to relate the values in each column to the response variable value...

Example, Part 3

Here is a diagnostic plot showing the result of learning a multiple linear regression model:



The model appears to be "OK," except predictive ability appears to break down for higher-redshift galaxies. Maybe we can do better with an ML model...

But enough for now. There is much for you to learn to get to this point!

What Should I Take Away From the Course?

- The overarching question: given data, can you perform basic analyses, and better yet, *explain these analyses to others*?
- This *is not* a course about the theory behind, and the mathematics of, statistical learning. You may learn this material yourself, if you are so inclined, but knowing it does not really affect your ability to analyze data and to explain your analyses to others.
- This *is* a course about context: how do the elements of the analysis workflow all fit together, on a (mostly) *qualitative* level? Your advisor and your readers don't care about the mathematics of the "kernel trick" underlying the support vector machine algorithm. They do care about *why* you used SVM, and perhaps why it added value in your particular analysis, and how it is different from other learning algorithms, etc. *Can you explain these things in words while standing in front of a poster or in a conference room?*
- Oh, and don't be an ML snob. Deep learning is not the first and last answer to all analysis questions.

OK, I Want to Do This

Let's learn about `R`, then.

Your question: "Why not [insert programming language/platform here]?"

My answer:

- `R` and `RStudio` are both free and, in my experience, extremely easy to install and maintain.
- The curve for learning how to analyze data is not steep, even for those with little programming experience.
- And `R` does everything a statistician wants to do, via functions in its base packages or in its 10,000+ contributed packages.

The only advantage to, e.g., `Python` is computational efficiency for big data. If you have a huge dataset, I would still suggest prototyping your analysis using a subset of your data in `R` and then translating your code. (Note that most undergraduate statistics majors at CMU are bilingual in `R` and `Python`. Other languages such as `Julia` and `Scala` are fine, but are really not necessary at all for your data analysis purposes.)

By the way, if you ask "Why not `Excel`?" you will be removed from the class roster and put on academic probation.

What is R?

From the R FAQ:

2.1 What is R?

R is a system for statistical computation and graphics. It consists of a language plus a run-time environment with graphics, a debugger, access to certain system functions, and the ability to run programs stored in script files.

The design of R has been heavily influenced by two existing languages: Becker, Chambers & Wilks' S (see [What is S?](https://cran.r-project.org/doc/FAQ/R-FAQ.html#What-is-S_003f)) and Sussman's Scheme. Whereas the resulting language is very similar in appearance to S, the underlying implementation and semantics are derived from Scheme...

The core of R is an interpreted computer language which allows branching and looping as well as modular programming using functions. Most of the user-visible functions in R are written in R. It is possible for the user to interface to procedures written in the C, C++, or FORTRAN languages for efficiency. The R distribution contains functionality for a large number of statistical procedures. Among these are: linear and generalized linear models, nonlinear regression models, time series analysis, classical parametric and nonparametric tests, clustering and smoothing. There is also a large set of functions which provide a flexible graphical environment for creating various kinds of data presentations...

...

R has a [home page](https://www.R-project.org/). It is free software distributed under a GNU-style copyleft, and an official part of the GNU project ("GNU S").

The R Language is an Interpreted Language

A key phrase on the previous slide is "[t]he core of R is an interpreted computer language." This means that instructions are executed directly when typed into the console of `RStudio`. For instance:

```
> 666
[1] 666
```

In this particular case, `R` doesn't know necessarily what to do with the input, so it outputs it to the screen (as the first element of an output vector...hence the `[1]`). If you input something of unknown type, like an uninitialized variable, you'd get an error instead:

```
> x
Error: object 'x' not found
```

But for now, let's defer learning more about `R` programming...

Installing Necessary Software

To download `R`, go to a "mirror site" of the Comprehensive R Archive Network, like this one at CMU. Download the version appropriate for your operating system and install it.

Once `R` is installed, download `RStudio` (the Desktop version, with a free open source license) by going to this web page. Again, download the version appropriate for your operating system and install it.

`R` comes bundled with a small subset of the 10,000+ packages that have been written for it. A particular package bundle that we would like to use, collectively dubbed the `tidyverse`, is not part of that subset. To install the `tidyverse`, you would begin by opening `RStudio` and noting how there are different panes. One includes tabs that say "Files," "Plots," and "Packages," etc. Click on "Packages," then click on "Install," then in the window that pops up, type in "tidyverse".

Once the `tidyverse` is successfully installed, we should be ready to go!

But Wait, I Still Don't Know R

That's OK...you'll learn the basics as you go. In the meantime, here is a resource for exploring `R` on your own (as optional homework):

- `RStudio` Primers

But you need not do this if you don't want to: next time, we will go over the basics of vectors and data frames, so that you can start working with tables. We'll build from there.