Data Science Crash-course

A data science quick-start guide by

SHARP SIGHT

Table of Contents

Data science is one of the most valuable skills of the 21st century

So you want to learn data science. That's awesome.

Right now, the world has more data than we know what to do with. Some writers have called it "the data deluge," which sounds hyperbolic, but it's fairly accurate. Corporations, non-profits, scientists, even individuals now have the ability to capture massive amounts of data.

Meanwhile, the world desperately needs insight. The world is overrun with problems waiting to be solved, we just need *insight into how to fix those problems.*

So we have plenty of data. And lots of problems waiting to be solved. What we don't have is enough people to analyze that data. We need insight, but don't have enough people to produce it.

That's where you come in. Data science is one of the biggest opportunities of this century. If you can master the skills of data, you'll not only be able to add massive value to the world by understanding and fixing complex problems, but also capture some of the value you create (in the form of profit).

Want to change the world? Want to create real value and build wealth?

Learn data science.

The first programming language you should learn

When people want to get started with data science, inevitably they ask "Which tools should I use? I don't know where to start!"

More often than not, the answer they receive is a list of every tool that might be occasionally used for data science. I've seen lists of *25+* tools for "getting started."

That's BS. You don't have the time to learn 25 tools. And, most businesses that are hiring data scientists are looking for expertise with only a few core tools (typically R or Python, SQL, Excel and a few auxiliary tools).

All that said, don't listen to people giving you lists of dozens of tools. As with almost any task, learning data science is best performed when you simplify and focus your efforts on the things that have the highest return on

investment. In this case, you want an "in demand" tool that has the greatest functionality and the lowest cost.

That tool is R.

In the long run, as a data professional, you'll want to develop skill in three core areas: data acquisition and data shaping (sometimes called data wrangling, or data munging), data visualization, and machine learning. R has excellent packages and toolsets for all of these skill areas. R is also the tool-of-choice at many top-tier academic institutions: it's used extensively at Stanford, Carnegie Mellon, MIT, to name a few.

To put it simply: if you're getting started with data science, R is the best tool to use.

With that in mind, I want to get you up-and-running with R within about an hour (actually, it could be as little as 10 minutes, if you move fast).

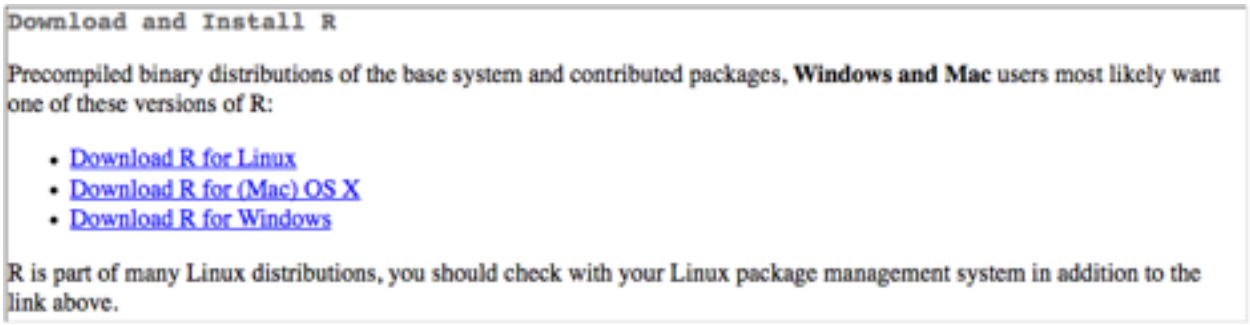
How to set up R

Install R

You first need to install R on your computer.

1. **Go to** http://cran.r-project.org/

(Here's a screenshot of what you'll see there.)


Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- Download R for Linux
- Download R for (Mac) OS X
- Download R for Windows

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

2. **Follow the installation instructions**

When you get to the installation webpage, find the appropriate file, and follow the installation instructions.

Install R Studio

Next you should install RStudio, which is an integrated development environment (IDE) for the R programming language. If you're a veteran programmer, you'll know what an IDE is, but if you're new to programming, think of an IDE as an application that allows you to write, execute, and administer your code, as well as manage your data and the outputs of your analysis all in one place.

1. Go to http://www.rstudio.com/products/rstudio/download/

2. Follow the instructions for installation for your particular platform (i.e., Mac, Windows, Linux)

Install Packages

Next, you want to install several R packages.

Packages are extensions of the R programming language. R has pre-built functions and datasets that come with the stock R installation, but sometimes you want to do things that are outside of R's basic function set. In this case, you can *extend* R with new functions by installing "packages."

The primary packages that we need to install are `ggplot2` and `dplyr`.

`ggplot2` is a graphics package that makes data visualization extremely easy. `dplyr` is a package for data manipulation.

In future chapters of this book, you'll learn a little about both data visualization using `ggplot2`, as well as data manipulation with `dplyr`. Very quickly though we're going to start with data visualization.

Data visualization is the best skill area to start with for a couple of reasons. First, it's easy to get started. It's easy to find data sets that are ready to be visualized. Second, data visualization is a "quick win." If you can learn visualization fairly rapidly. Third, it's ubiquitous in data science at almost

every part of the workflow. Whether you're creating a report, doing exploratory analysis, or applying a machine learning algorithm, you'll need to use visualization as an exploratory or communication tool. Lastly, in the analytics world, few people are actually good at visualization. Truthfully, it's one of the most necessary skills, but few analysts are truly exceptional at it (so if you can master it early, you'll be able to stand out).

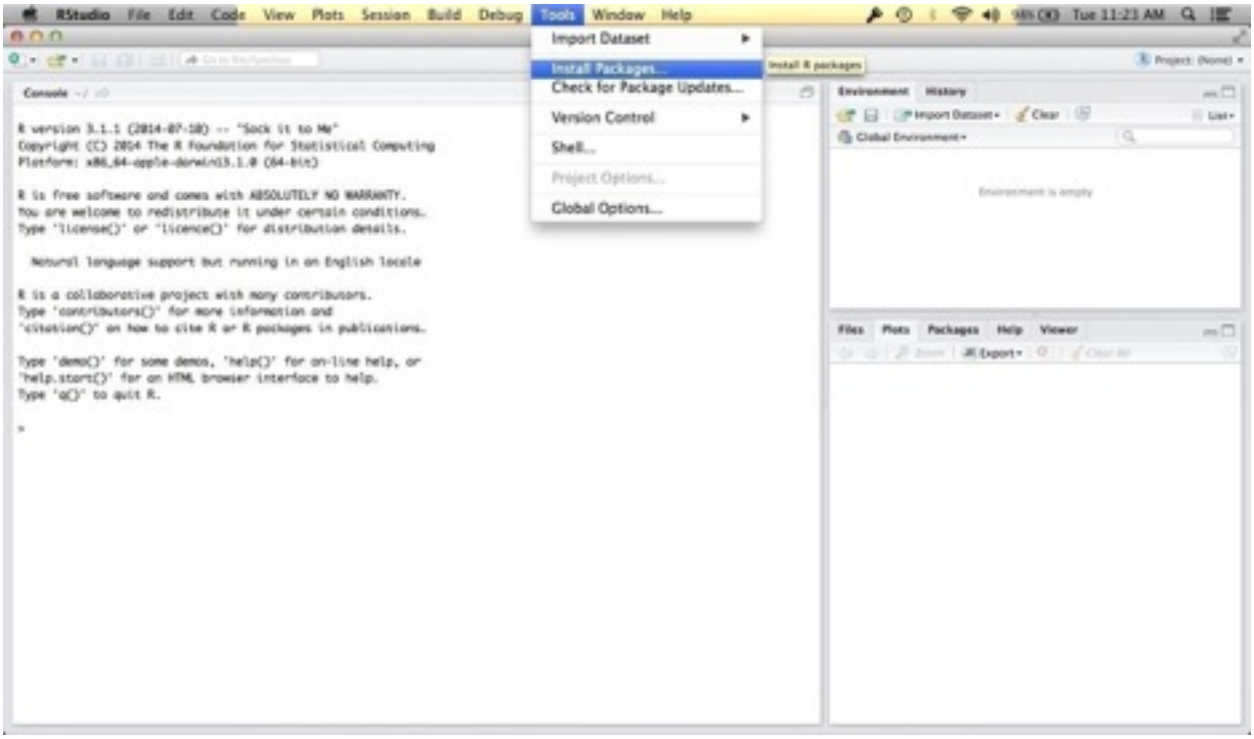
The reason I am going to teach you `ggplot2`, is that it's a best-in-class tool for data visualization. `ggplot2` is excellent because it's highly structured. I won't go into detail here, but I'll note that the structure of `ggplot2` will help you learn how to think about visualizing data. Once you learn the syntax as well as the overall conceptual framework, creating insightful visualizations becomes much, much easier.

(Note: the following shows how this is done on a Mac. The basic procedures should be very similar on Windows or Linux.)

Install `ggplot2` and `dplyr`

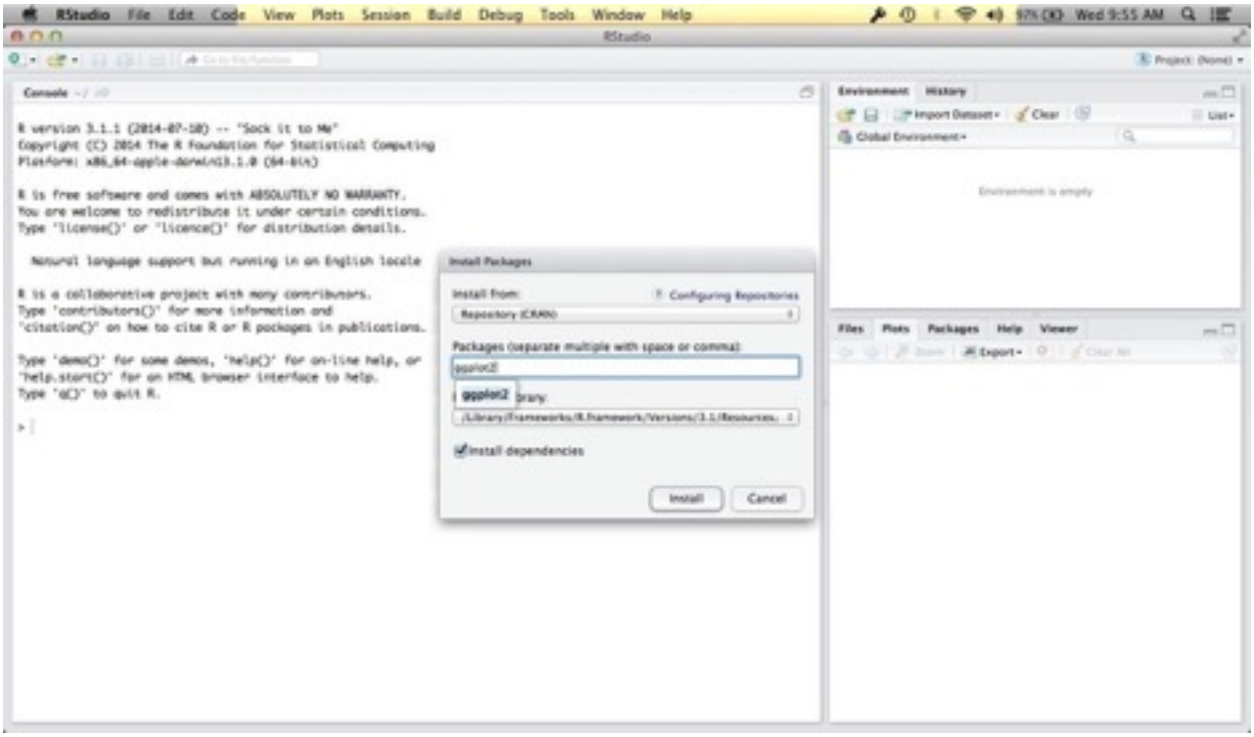
1. Open R Studio

2. Go to: Tools > Install Packages



3. Install "`ggplot2`" package

- Note: make sure the "Install dependencies" check box is checked



4. Repeat instructions 2 and 3 for `dplyr` and `caret`

A STEP-BY-STEP DATA SCIENCE LEARNING PLAN

You've probably read numerous articles telling you how to start learning data science. Collectively, they tell you to dozens of things you need to learn. Learn Python. Learn R. Learn Hadoop. They tell you all the skills you need: learn machine learning, visualization, data wrangling. Little technical skills like manipulating vectors, matrices, loops. More tools like Pig, Hive, d3, and Tableau.

For most students it's overwhelming.

It's like standing at the base of Everest, saying to yourself, "how the hell will I climb that?"

There's a mountain of material out there, and it will be difficult to climb without knowing where to start.

You need a path

Look, there are lots of people out there that aren't actually skilled in data science, telling you how to start learning data science (I'm looking at you, HR professionals).

Then, there are people who are actual data scientists, but are terrible teachers and communicators. I'm sure you know what I'm talking about. The guy with the super-elite PhD who says "oh, it's easy," and then proceeds to talk for 45 minutes about arcane math that nobody can understand.

Let me break this down for you: you can't learn everything at once. Your time is limited.

And, you shouldn't learn everything with equal enthusiasm.

Some skills are more useful than others, and some skills are easier to learn than others.

Some skills are used every day by almost every data scientist, and other skills are “specialty” skills, used either by a select few specialists or used only occasionally by the average analyst.

Early in your learning, you need to be able to distinguish between the essential and the “nice to have.”

You need to be selective, and you need to learn things in the proper order. You need a path for getting started.

A step-by-step data science learning path

You need to focus on learning the skills with the highest return on investment (ROI). Focus on the skills that are easy to learn, easy to

implement, that yield the greatest results. (ahem. Do you know the results clients actually want? Maybe you need to learn what clients want first.)

Learn R

I believe that it's best to focus your efforts. Learn one tool.

Right now, the tool I recommend is R.

There are a few reasons I recommend R:

1. R is the most common programming language among data scientists. O'Reilly Media just released their 2014 Data Science Salary Survey. In that report, they note that R is the most commonly used programming language (if you exclude SQL).

2. R has 2 packages that dramatically streamline the data science workflow:
    - `dplyr` for data manipulation
    - `ggplot2` for data visualization

3. R is common (and increasingly common) at the best tech companies like Google, Twitter, Netflix, LinkedIn, and Apple.

To be fair, I think you could also make a strong case for learning Python (it came in just behind R in O'Reilly's list of data science tools).

However, `dplyr` and `ggplot2` "tip the scales" in favor of R.

As I've noted earlier, `ggplot2` has a deep structure that underlies it's syntax. When you learn that structure, you begin to think about data visualization in a deep way. In some sense, learning `ggplot2` teaches you how to think about visualization.

Similarly, `dplyr`'s syntax is easy to learn, easy to use, and operates in a way that streamlines your workflow. `dplyr` is one of the best data management tools I've ever seen.

Moreover, when you start to combine `dplyr` with `ggplot2` through "chaining" (much like Unix pipes) you can rapidly explore your data with very little effort.

Learn data visualization first

Data visualization is the fastest, easiest means of finding insight when you're first starting out. In your first few months, it's the highest leverage, highest ROI skill. It's also one of the most versatile tools: you can use visualization for data exploration (finding insight) and communication (communicating what you find to business partners and executives).

For learning visualization, I recommend R's `ggplot2`. Again, I recommend `ggplot2` because it has a deep underlying structure to it. When you learn

the structure of the syntax, you are at the same time learning deep principles about visualization.

For starters, I suggest learning the “core” visualization techniques. These core techniques are:

- the bar chart
- the line chart
- the scatterplot
- the small multiples design

These are important for a few reasons:

1. These charts are the basis for more advanced visualizations. For example, the bubble chart is just a variant of the scatterplot. In fact, the dot distribution map is nothing but a modified scatterplot.

2. These charts have a structure to them. When you learn that structure, you will start to learn how to think about visualizing your data.

3. The charts are the essential “tools of insight.” They are foundational tools for finding insights (during data exploration) and communicating insight. And insight is what clients actually want and need.

Learn data manipulation second

Once you learn the foundational tools of data visualization, you can “back into” data manipulation in R.

When you just start learning data science, you can use “dummy data” or very simple data sets that don’t require much data reshaping.

As you advance though, the “shape” of your data will be a problem: you’ll have multiple data files that you need to join together; you’ll need to subset and change variables; you’ll need to do lots of aggregations. When you reach this point – when the shape of your data is a bottleneck – then you should put more time into learning data manipulation. An example of this is

the recent tutorial analyzing 'supercar' data, where the data were found in five separate files.

When you're starting to learn data manipulation, I recommend mastering the 5 basic `dplyr` verbs, as well "chaining" using the `%>%` operator.

Learn machine learning last

I know: this is the opposite of what most other people are telling you.

The fact is, the vast majority of data jobs – particularly the entry level jobs – are not machine learning jobs.

Think of a baseball team. There are core baseball skills (hitting, throwing, and running). The vast majority of people on the team have a mix of skills. Teams are built from individuals with mixes of the core baseball skills. And then there's one guy who is highly specialized in the most arcane of skills: pitching (specialization in throwing).

Machine learning is similar to pitching. It's valuable, technical, a little arcane, and difficult to do well. There are also fewer of those jobs. Strategically, you're much better served learning visualization and data manipulation: they are easier to learn, easier to implement, and the jobs are more plentiful.

Not to mention: you need data visualization and data manipulation for machine learning anyway! If you're implementing a ML algorithm, you still need to put your data into the right shape first. And when you're done, in most cases you'll need to explore the results. Typically you'll perform this data exploration with data visualization.

So to summarize my view on ML: in the beginning it's the skill with the lowest return on investment! It's the hardest to learn, the hardest to implement (both because it's difficult and because it requires a foundation in data manipulation and visualization), and there are fewer jobs requiring machine learning.

Keep in mind: I'm not saying that you should never learn machine learning. It's extremely valuable. I'm just saying that you should learn it *after* you've built solid foundation of data visualization and data manipulation.

TL;DR (How to start learning data science)

Here's the recap of how to start learning data science:

- Choose one tool: the R programming language
- Learn data visualization first (with R's `ggplot2`), using simple data or dummy data. Then find a more complicated dataset
- Learn data manipulation second (with R's `dplyr`), and practice data manipulation on your more complex data
- Learn machine learning last

You need to master the basics first

Recently, an acquaintance told me that he was interested in getting started with machine learning.

He's a web developer who primarily works in Ruby and Python, but also has a small amount of experience with R. Day-to-day, he works as a run-of-the-mill web developer, but he's confessed to me that he's a bit bored and looking for something new and exciting.

When he told me he wants to get into machine learning, we started talking. The conversation went like this:

Sharp Sight: "Do you know data visualization? Do you know data wrangling techniques?"

Acquaintance: "I don't want to do data visualization."

Sharp Sight: “You don’t understand. Data visualization is a prerequisite for machine learning. You need to learn how to dive into a dataset and analyze it before you can make machine learning algorithms work. You need to be able to analyze data first.”

We proceeded to talk for about 10 minutes. He asked a few questions, and I gave him solid advice. I gave him the advice I told you earlier in this ebook: learn basic plots, master basic syntax. Focus on foundations. Be really systematic about learning data visualization and data wrangling, and your progress with ML will be much, much faster.

His response?

“Well, I’ll just figure it out.”

“Jump in and figure it out” is a losing strategy

You have to know this guy. He’s young. He’s cocky. He doesn’t know what he doesn’t know yet.

“I’ll just figure it out” is code for, “I don’t want to do all that stuff you recommended, so I’m just going to jump in, without the prerequisites, and see how far I get.”

His plan is to jump in *without* a plan, and to overcome the challenges in front of him with feigned confidence and a bold attitude.

You need to understand: this is a losing strategy.

I’m fairly certain that if I grill him in 6 months, he won’t know many of the core ML concepts that he needs to understand, if any. I’m confident that if I gave him a dataset, and asked him to write me some code (from scratch

and by memory) to implement a logistic regression, he wouldn't be able to do it.

It's not because he's unintelligent (he's a fairly smart guy). It's because his approach is wrong and likely to lead him to failure.

To be clear, there are absolutely people who will be able to "figure it out." People who use the jump-in-and-figure-it-out strategy sometimes get to their goal. But the odds aren't good, and it's terribly inefficient (you'll work harder, for fewer gains).

Average performers demand to learn the "sexy" stuff first

It's understandable … everyone wants to learn the sexiest stuff first. This isn't just limited to data science.

People who begin learning a musical instrument do the same thing. They say, "I want to play guitar" but they want to jump right into playing

advanced guitar solos, instead of meticulously and intensely mastering the basics. And because they don't want to master the basics, they fail to learn critical skills and ultimately miss their target. They never learn the basics, and they never learn to shred.

Don't be that guy.

Top performers are disciplined and systematic

Top performers are different.

If you look at top performers of all stripes, they are extremely methodical in how they approach learning and skill acquisition.

Navy SEALs, olympians, top performing students, elite musicians (violinists, cellists, guitarists) …. the best people are patient, disciplined, and strategic.

They don't demand to start with the cool stuff. Top performers diligently learn and master the foundations.

To be the best, you need to learn the right way

I get it. The cool stuff is why you want to get into data science in the first place. For example, machine learning is very exciting right now. It's powering self driving cars, intelligent IoT objects, and a variety of other cutting-edge technology. Of course you want to do machine learning.

But ask yourself: do you want to be in the bottom 95% who fail to really learn? The bottom 95% that under-perform? The bottom 95% who make less money? The bottom that say "I'll just figure it out" but then fail, and complain about how hard data science is?

Or do you want to be in the top 5%? … the top 5% who earn most of the money, get the best perks, and work on the coolest projects.

You can choose which group you fall into – the top 5% or the bottom 95%.

You choose by how you learn.

How to create 3 essential data visualizations

Ok, by now, you should have R installed and set up.

We've also talked about what you should learn first: data visualization.

And I've emphasized why: you need to master the basics *first* before you jump into more advanced skills.

Having said that, I'm going to show you 3 "core" data visualizations that you need to know. If you want to be a data scientist, you need to eventually be able to do these "with your eyes closed." Getting there will take practice. In the beginning though, you'll be able to take this code, copy it, and run it.

3 core data visualizations

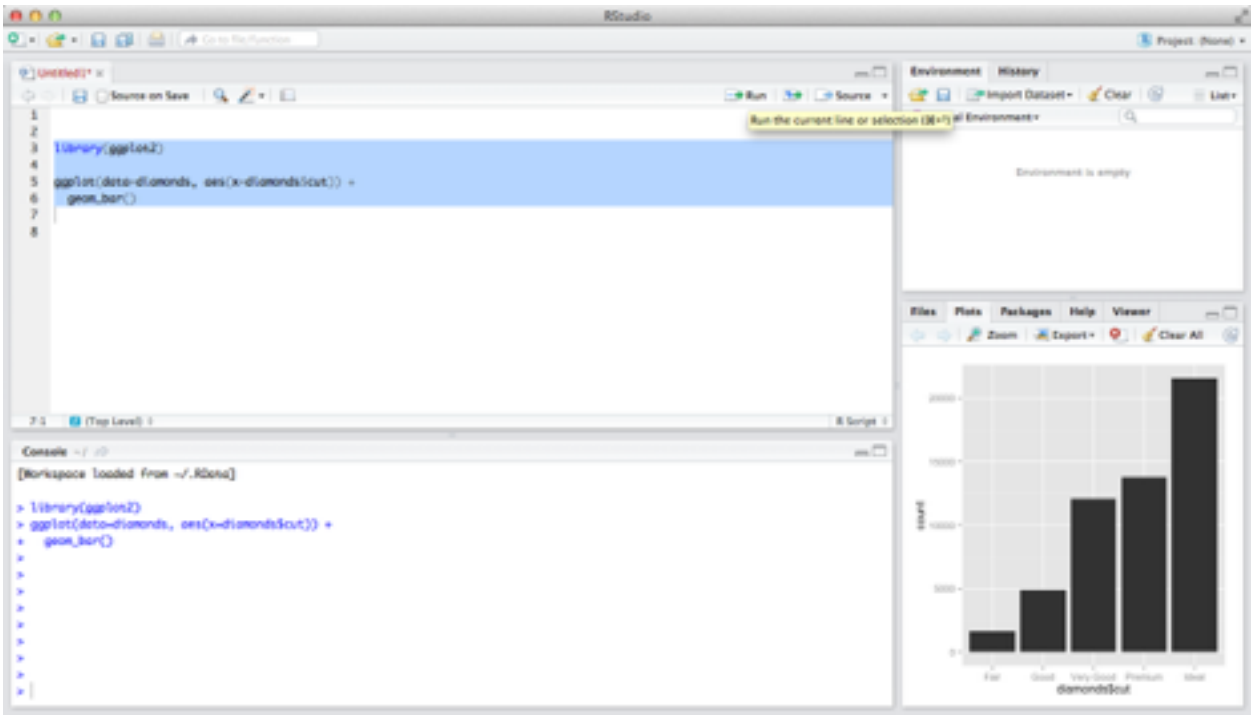
Here are 3 essential visualizations that you should master:

1. Bar chart
2. Scatter plot
3. Line chart

I want to point out that while these are fairly basic visualizations, when we take a closer look, they show something important. Each one is made up of basic components - namely bars, lines, and points respectively. These components (which we will later refer to as "geoms") are *building blocks of more complex visualizations.* So, don't be deceived. Yes, these are somewhat basic. But if you can master these charts and a few other basic building blocks, you'll later be able to create much more complex, insightful, and valuable visualizations.

To run this code, you need to open up RStudio (by this point, you should already have installed the R programming language and R Studio. If you haven't installed them already, go back to the chapter with the installation instructions and install them both before proceeding.

In RStudio, you need to type in the code in the following sections, highlight the code, and click the "Run" button (see the screen shot).

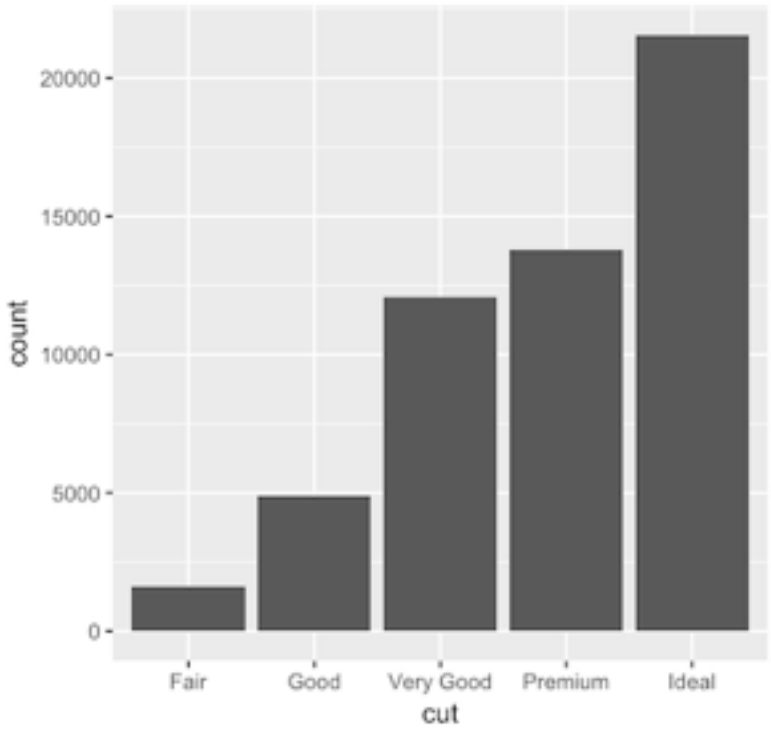


Bar Chart Code

First, the bar chart. For the time being, you can type this code in. (You could also copy-and-paste it, but I don't recommend copy-and-paste as a learning strategy.)

```
library(ggplot2)

ggplot(data = diamonds, aes(x = cut)) +
  geom_bar()
```



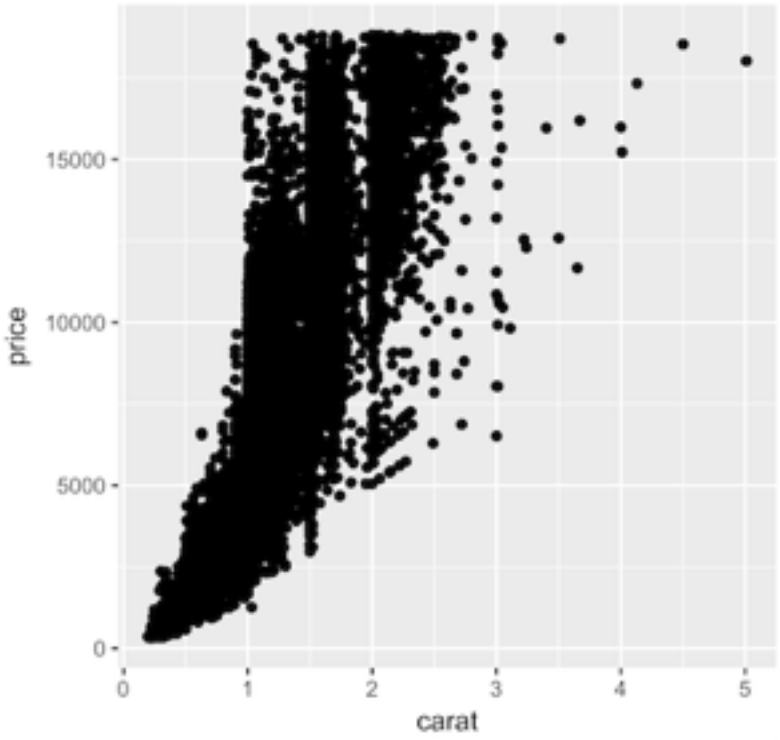
Scatter Plot Code

Next, the scatter plot. The scatter plot is very, very common. We use it for reporting, exploratory data exploration …. a variety of applications.

Here's the code:

```
library(ggplot2)

ggplot(data = diamonds, aes(x = carat, y = price)) +
  geom_point()
```



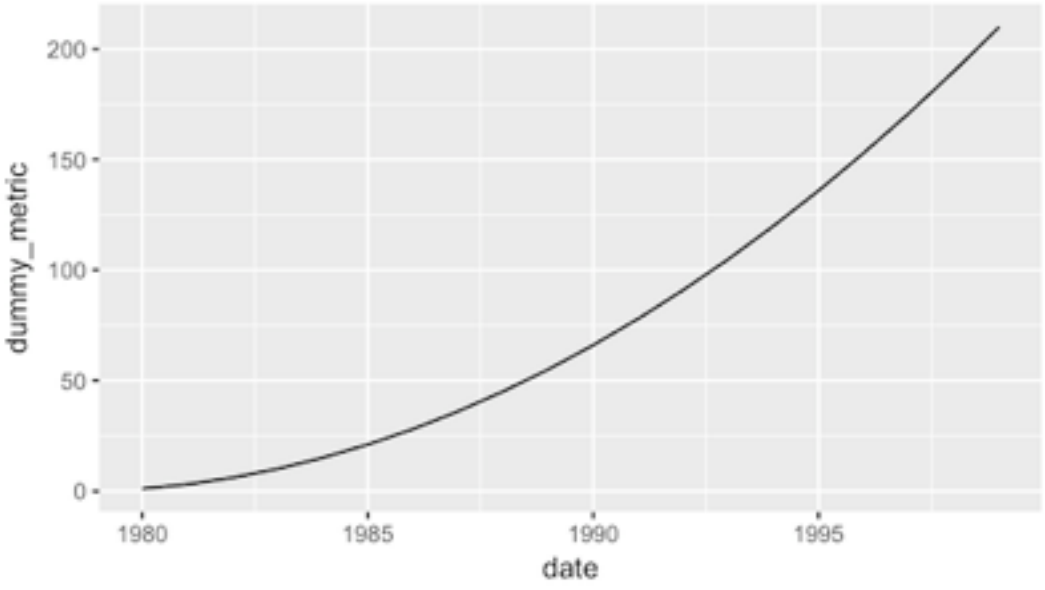
Line Chart Code

For the line chart, we're going to create a simple dummy dataset, then plot it using `ggplot2`.

```
# Load ggplot2 graphics package
library(ggplot2)


# Create dummy dataset
df.dummy_data <- data.frame(
  dummy_metric = cumsum(1:20),
  date = seq.Date(as.Date("1980-01-01"), by="1 year", length.out=20)
  )

# Plot the data using ggplot2 package
ggplot(data = df.dummy_data, aes(x = date, y = dummy_metric)) +
  geom_line()
```



dplyr: How to do data manipulation with R

Ok. Here's an ugly secret of that data world: lots of your work will be prep work.

Of course, any maker, artist, or craftsman has the same issue: chefs have their *mise en place*. Carpenters spend a hell of a lot of time measuring vs. cutting. Etcetera.

So, you just need to be prepared that once you become a data scientist, 80% of your work will be data manipulation.

Getting data, aggregating data, subsetting data, cleaning it, and merging various datasets together: this constitutes a large percent of the day-to-day work of a data scientist. Data wrangling is an essential skill.

When you're just starting out with analytics and data science, you can get away with doing only minimal data manipulation. In the beginning, your datasets are likely to be txt, csv, or simple Excel files.

And if you do need to do some basic formatting, you can use use Excel if it's a small and simple dataset. (actually, Excel is a good tool in your workflow for basic data manipulation tasks.)

As you progress though, you'll eventually reach a bottleneck. You'll start doing more sophisticated data visualizations or machine learning techniques, and you will need to put your data in the right format. And you'll need a better toolset.

By most accounts, the best toolset for data manipulation with R is `dplyr`.

`dplyr`: the essential data manipulation toolkit

In data wrangling, what are the main tasks?

- Filtering rows (to create a subset)
- Selecting columns of data (i.e., selecting variables)
- Adding new variables
- Sorting
- Aggregating

`dplyr` gives you tools to do these tasks, and it does so in a way that streamlines your workflow. It's not an exaggeration to say that `dplyr` is almost perfectly suited to real data science work. (Data science as it's actually practiced … not data science theory.)

To be clear, these aren't just the "basics." They are the essentials. These are tasks that you'll be doing every. single. day. You really need to master these.

Again though, `dplyr` makes them extremely easy. It's the toolset that I wish I had years ago.

Moreover, once you combine `dplyr` verbs with "chaining," it becomes even more streamlined and more powerful. Not to mention, chaining together the data wrangling tools of `dplyr` with the data visualization tools of `ggplot2`. Once you start combining these together, you will have a powerful toolset for rapid data exploration and analysis.

`dplyr` verbs

`dplyr` has 5 main "verbs" (think of "verbs" as commands).

`filter()`

`filter()` performs row selection from your data.

`filter()` subsets your data by keeping rows that meet specified conditions.

Let's look at an example.

```
library(dplyr)
library(ggplot2)

head(diamonds)

#--------
# FILTER
#--------

df.diamonds_ideal <- filter(diamonds, cut=="Ideal")
```

Here, we're subsetting (i.e., filtering) the diamonds dataset and keeping only the rows where `cut=="Ideal"`.

select()

`select()` allows you to select specific columns from your data.

In the following code, we're going modify our data frame, selecting only the columns we want.

```
#---------------------------------------
# SELECT
# - select specific columns from your data
#---------------------------------------

df.diamonds_ideal <- select(df.diamonds_ideal, carat, cut
                            , color, price, clarity)

head(df.diamonds_ideal)
# carat   cut     color price clarity
# 0.23    Ideal     E   326     SI2
# 0.23    Ideal     J   340     VS1
# 0.31    Ideal     J   344     SI2
# 0.30    Ideal     I   348     SI2
# 0.33    Ideal     I   403     SI2
# 0.33    Ideal     I   403     SI2
```

Notice that now `df.diamonds_ideal` only has our selected columns:

`carat`, `cut`, `color`, `price`, and `clarity`.

`mutate()`

`mutate()` allows you to add variables to your dataset. Obviously, this is a really common task in any programming environment.

```
#-------------------------------------
# MUTATE:
# - Add variables to your dataset
#-------------------------------------

df.diamonds_ideal <- mutate(df.diamonds_ideal
                            , price_per_carat = price/carat)

head(df.diamonds_ideal)
# carat   cut     color price clarity   price_per_carat
# 0.23    Ideal     E   326     SI2        1417.391
# 0.23    Ideal     J   340     VS1        1478.261
# 0.31    Ideal     J   344     SI2        1109.677
# 0.30    Ideal     I   348     SI2        1160.000
# 0.33    Ideal     I   403     SI2        1221.212
# 0.33    Ideal     I   403     SI2        1221.212
```

Using `mutate()`, we've successfully added a new variable:

`price_per_carat`.

`arrange()`

`arrange()` sorts your data. To be clear, there are other sorting functions that you can use from base R, but the syntax is a bit of a pain.

Ok, in the following example, I'll show you how to use `arrange()`. Having said that, we're not going to use the `diamonds` dataset, because it's too large to properly see order at work.

So, we'll create a simple data frame with a numeric variable. This numeric variable has the numbers out of order and we'll use arrange to reorder the numbers.

```
#------------------------
# ARRANGE: sort your data
#------------------------

# create simple data frame
#  containing disordered numeric variable
df.disordered_data <- data.frame(num_var = c(2,3,5,1,4))

head(df.disordered_data)

# these are out of order

# 2
# 3
# 5
# 1
# 4

# now we'll order them with arrange()
arrange(df.disordered_data, num_var)

# 1
# 2
# 3
# 4
# 5


# we can also put them in descending order
arrange(df.disordered_data, desc(num_var))

# 5
# 4
# 3
# 2
# 1
```

Sorting your data may not seem useful, but it's used more often than you might think. Quite a bit of data-inspection involves sorting and ordering your data. This is more useful than it might seem at first blush.

summarize()

`summarize()` allows you to compute summary statistics.

Again: the following (simple) example might not seem useful, but `summarize()` becomes extremely useful when combined with `group_by()`.

```
#-------------------------------
# SUMMARIZE:
#  aggregate your data
#-------------------------------

summarize(df.diamonds_ideal
          , avg_price = mean(price, na.rm = TRUE) )

#   avg_price
#   3457.542
```

`dplyr`: one of the data science essentials

As I've already mentioned, to quickly master data science, you need to master the *basics* first. That includes these 5 `dplyr` verbs. Master these data manipulation tools, along with a few critical data visualization tools (like the line chart, the bar chart, and the scatterplot) and you'll have a solid foundation that you can build on. You'll have many of the tools you need to move on to more advanced topics like machine learning.

What is machine learning and why is it so important?

On Google's recent Q3 earnings call, Google's CEO, Sundar Pichai said that one "transformative" technology is causing Google to rethink "how we're doing everything."

Read that again. There's a single technology that's causing Google to rethink they way it does everything.

And it's not just Google.

The same technology is in the process of transforming many of the biggest names in tech — Facebook, Amazon, Netflix, UBER, Twitter — not to mention smaller, up-and-coming startups.

Entrepreneur and thought leader Peter Diamandis say that this technology will “do more to improve healthcare than all the biological sciences combined” and will generate large amounts of wealth and abundance.

Billionaire venture capitalist Vinod Khosla agrees, saying that over the next 50 years, it will drive abundance, transform industries, and impact almost every part of society.

What is it?

Machine learning.

What is machine learning

You’ve probably started to read about about machine learning in the popular press and technology press. Media outlets like TechCrunch, New York Times, and Forbes have been writing about it recently.

That said, it's not always called "machine learning." In some outlets, you'll hear about "statistical learning." (This is less common in the popular news. "Statistical learning is more commonly used in academic circles.) To be fair, there are minor differences between statistical learning and machine learning, but as far as the common ML practitioner is concerned, they are basically the same thing.

Machine learning is also discussed indirectly. You might read articles about closely related fields, like machine intelligence, artificial intelligence, predictive analytics, and automation. To be clear, these are not all identical to machine learning, but they strongly related. Machine learning is a key component of all of these areas.

This is important to keep in mind: machine learning is an area of study in and of itself, but in an applied setting, it has wide-ranging applications that cut across various parts of business and software.

Machine learning, a quick and dirty definition

So what is machine learning?

Machine learning is using computation and algorithms to enable computers to learn from data and make predictions (source: wikipedia article on machine learning).

From the standpoint of a practitioner, machine learning is is a set of tools for writing programs that input data and output predictions or decisions.

Software that learns: the key to unlocking the value of Big Data

Currently, most software does not adapt. It does not learn. It has no intelligence and only limited ability to change how it works to meet the needs of the user.

In contrast, programs that incorporate machine learning can adapt. They can learn over time, and improve.

The process that machine learning algorithms use to “learn” is analogous to how a human learns through examples. To be clear, this is a simplification, but an instructive one.

When a human learns by example, they observe the examples, make generalizations about the examples, and then apply those inferences and generalizations to new observed data.

What’s critical in the process is the data. Data is required to provide training examples from which to learn.

Although this is a large simplification, machine learning (and specifically, supervised learning) works in a very similar way: observe training examples, make generalizations, and then apply those generalizations to new observed data.

The main takeaway here, is that the key component in getting machine learning algorithms to learn and operate properly is *data*. In fact, some of the most powerful machine learning techniques not only require data, but very large amounts of data.

This is critical for understanding why Google is "rethinking everything."

Software that predicts and adapts

Once trained, the machine learning algorithm (and again, specifically, a "supervised learning" algorithm) can make predictions.

We may not recognize this, but much of what human beings do involves a *prediction* of some sort. Simple, everyday tasks like picking up an object involve a variety of predictions, like "how far away is the object" and "how much force do I need to apply" to pick it up. These predictions happen below our level of conscious awareness, but they do occur.

Similarly, in the task of driving a car, humans make predictions about terrain conditions and about the outcomes of various directional changes.

In more complicated cognitive tasks, even decisions can be boiled down predictions; when we make decisions we're choosing the best of several options, where each option has it's own predicted outcome. For example, when we decide to watch one movie over another, we are, in some sense, implicitly making predictions about our satisfaction with the available options, and choosing the best one.

And it's certainly not just inconsequential decisions like choosing the best movie. Effectively, every important decision we make is underpinned by some type of prediction. We make decisions about how to spend money, how to invest, and how to allocate our time, and how to select people as friends and team-members. Good decisions (and therefore, good predictions) are required for a variety of tasks with important consequences.

Good predictions are valuable

If you look closely, you can see that in a wide variety of areas, from driving a car, to choosing how to allocate resources, good predictions are extraordinarily valuable.

And increasingly, where there's a prediction to be made, there's a machine learning algorithm that can automate it.

To the extent that these predictions can be automated and optimized, machine learning algorithms – powered by data – can produce tremendous value.

So, to recap where we're at, machine learning is powered by data. Good data is required for training machine learning algorithms. But at the same time, the machine learning algorithms, provide tremendous value by make and automate decisions.

In some sense, machine learning unlocks value from data.

What that also means, is that machine learning will become more valuable as data volumes increase.

Again, this helps us understand what's going on with Google. It helps us understand why Google is "rethinking everything" because of machine learning.

Google is sitting on more data (and better data), than almost anyone in the world. Data is necessary for training machine learning algorithms. In turn, these machine learning algorithms enable them to create adaptive software. And this adaptive, intelligent software can produce massive value.

Why machine learning will become even more valuable in the next decade

Having said that, for companies like Google, it's not only about the data that they currently have, but the data that they will have in the future.

The estimates vary, but common estimates state that the amount of data that humanity generates is doubling about every two years.

Given this exponential increase in data, by 2020, it's estimated that the world will have 44 Trillion gigabytes of data.

Unfortunately, our human intuition breaks down when trying to conceptualize volumes this large (and rates of change this fast). What we can say, in very simplistic terms, is that data is growing very, very fast and reaching huge volumes.

Sensors, exponential data, and connecting the physical & digital worlds

This explosion of data is being driven not only by the internet itself, but also the emerging connection between the virtual and physical worlds.

That sounds a little abstract, so let me unpack it.

Sensors are becoming smaller and less expensive. As they become smaller and less expensive, companies are adding a wide variety of sensors to physical objects. The current example of this is mobile phones. Mobile phones are instrumented with a variety of sensors that enable the phones to collect data.

Just like in mobile phones, as sensors get smaller we will be able to add more sensors to everyday objects. We'll be able to instrument more and more of the physical world. Effectively, this is what people mean when they say "the Internet of Things." The IoT isn't so much an internet of things, but an internet of sensors, data, actuators, and intelligent software.

Setting aside the large topic of the IoT (that's a different article) the point here is that data is exploding, and will continue to explode as sensors miniaturize and we begin to connect the physical world with the digital world.

And again, the value of this data (and really, the value of the IoT) will largely come from machine learning and intelligent software.

Software is eating the world, and machine learning is eating software

In some sense, this is the ultimate meaning of Marc Andreessen's thesis that "software is eating the world."

Back in 2011, the famous entrepreneur and venture capitalist wrote an article for the Wall Street Journal making the bold claim that "Software is Eating the World."

Looking backwards, and seeing the rise of companies like AirBnB, Uber, not to mention the continued success of Amazon and Netflix (as they dominate their industries and outcompete their former brick-and-mortar competitors), it seems that Andreessen was absolutely correct. Software is being built into the DNA of the most successful companies of our time. Software is critical to the success of the companies who are disrupting old industries, and winning.

But a large part of what makes the software at these companies so valuable is machine learning. In many of the most successful software-driven companies, machine learning is part of the "secret" that drives their success.

Take Amazon. Like nearly all businesses, what drives revenue at Amazon is very simple: increased customers, increased order sizes, and repeat buys. At every step of this process, Amazon uses machine learning to optimize. For example, to increase order sizes, Amazon uses machine learning to recommend "similar items" as you browse and make your

purchase. It also uses similar techniques to target past customers with new “recommended” offers.

If we look closely, Amazon is really just a data-driven marketing machine that sells books. (well, to be clear, Amazon sells everything.)

Similarly, Netflix (who uses very similar machine learning techniques) is a data driven marketing machine that sells movie rentals.

In both cases, data is the fuel, and machine learning is the critical part of the “engine” of those machines. And software sort of wires it all together.

Yes, these are software-driven businesses. And we can say that they are data-driven businesses. But what entrepreneurs and software engineers alike need to understand, is that in a very critical way, these are machine learning businesses.

And that’s only the initial part of the story.

As we instrument the world, and collect larger amounts of data from everything, we can build software for the physical world that's powered by that data; software that interacts with the world (i.e., robotics) and also software that digitizes and optimizes formerly physical services (like Uber and AirBnB).

As we collect more data from the world, we'll be able to use machine learning to create better software that predicts, decides, optimizes and adapts. We'll be able to build more valuable software.

Yes: software is eating the world, but machine learning is eating software.

Why software engineers and entrepreneurs need machine learning

This then is why Google is transforming everything it does because of machine learning. Machine learning will allow them to unlock the value of big data.

In any industry that Google (or Alphabet) is in – search, mobile, the IoT, robotics, and even healthcare – it will allow them to create software that unlocks the value of their massive datasets.

Machine learning will likely create massive amounts of wealth for companies like Google and will have a large impact on almost every part of society: transportation, healthcare, communications, marketing, to name a few.

To quote billionaire investor Vinod Khosla (one of the heavyweights of the venture capital world): “I think the impact of machine learning on society will be larger than the impact of mobile … Almost any area I look at, machine learning will have a large impact.”

Ultimately, the pervasive impact of machine learning on business (not to mention, society itself) is causing investors like Khosla to invest.

So Google is rethinking everything because of machine learning. Investors like Khosla are investing heavily, claiming that machine learning will have a large impact on almost everything.

Here's my claim: if you're a budding data scientist, a software developer, or an entrepreneur, you need to learn and leverage machine learning.

It's not just a transformative technology that is causing Google to rethink everything. It's a transformative technology that will cause software developers and entrepreneurs to rethink everything (whether they want to or not).

Software businesses will increasingly disrupt traditional industries. And intelligent software will increasingly win out over "dumb" software.

Applications that learn, adapt, optimize, and automate will increasingly win.

Moreover, data-driven businesses that leverage machine learning and develop systems to optimize and automate marketing, hiring, and operations are going to out-compete those that don't.

Google understands this. Investors like Khosla understand this. You need to understand it too.

So, once you've mastered the essential tools of data science (i.e., data visualization and data manipulation), I strongly encourage you to dive into machine learning.

A QUICK INTRODUCTION TO MACHINE LEARNING IN R WITH `caret`

If you've learned the essential data visualization and data exploration techniques, the next logical step is to start learning some machine learning.

To help you begin learning about machine learning in R, I'm going to introduce you to an R package: the `caret` package. We'll build a very simple machine learning model as a way to learn some of `caret`'s basic syntax and functionality.

But before diving into `caret`, let's quickly discuss what machine learning is and why we use it.

What is machine learning?

Machine learning is the study of data-driven, computational methods for making inferences and predictions.

Without going into extreme depth here, let's unpack that by looking at an example.

A simple example

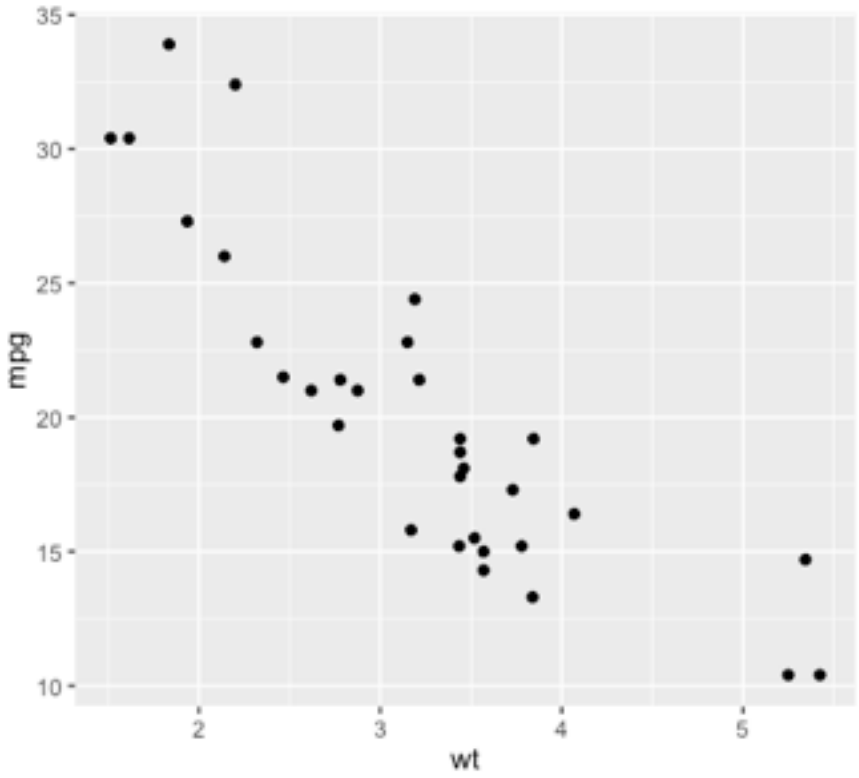
Imagine that you want to understand the relationship between car weight and car fuel efficiency (I.e., miles per gallon); how is fuel efficiency effected by a car's weight?

To answer this question, you could obtain a dataset with several different cars, and attempt to identify a relationship between weight (which we'll call `wt`) and miles per gallon (which we'll call `mpg`).

A good starting point would be to simply plot the data, so first, we'll create a scatterplot using `ggplot2`:

```
library(ggplot2)

ggplot(data = mtcars, aes(x = wt, y = mpg)) +
  geom_point()
```



Just examining the data visually, it's pretty clear that there's *some* relationship. But if we want to make more precise claims about the relationship between `wt` and `mpg`, we need to specify this relationship mathematically. So, as we press forward in our analysis, we'll make the assumption that this relationship can be described mathematically; more precisely, we'll assume that this relationship can be described by some mathematical function, $f(x)$. In the case of the above example, we'll be making the assumption that "miles per gallon" can be described as a function of "car weight".

Assuming this type of mathematical relationship, machine learning provides a set of methods for identifying that relationship. Said differently, machine learning provides a set of computational methods that accept data observations as inputs, and subsequently estimate that mathematical function, $f(x)$; machine learning methods learn the relationship by being trained with an input dataset.

Ultimately, once we have this mathematical function (a model), we can use that model to make predictions and inferences.

How much math you <u>really</u> need to know to do machine learning

What I just wrote in the last few paragraphs about "estimating functions" and "mathematical relationships" might cause you to ask a question: "how much math do I need to know to do machine learning?"

Ok, here is some good news: to implement basic machine learning techniques, you don't need to know much math.

To be clear, there is quite a bit of math involved in machine learning, but most of that math is taken care of for you. What I mean, is that for the most part, R libraries and functions perform the mathematical calculations *for you*. You just need to know which functions to use, and when to use them.

Here's an analogy: if you were a carpenter, you wouldn't need to build your own power tools. You wouldn't need to build your own drill and power saw. Therefore, you wouldn't need to understand the mathematics, physics, and electrical engineering principles that would be required to construct those tools from scratch. You could just go and buy a drill and saw "off the shelf." To be clear, you'd still need to learn how to *use* those tools, but you wouldn't need a deep understanding of math and electrical engineering to operate them.

When you're first getting started with machine learning, the situation is very similar: you can learn to use some of the tools, without knowing the deep mathematics that makes those tools work.

Having said that, the above analogy is somewhat imperfect. At some point, as you progress to more advanced topics, it will be very beneficial to know the underlying mathematics. My argument, however, is that in the beginning, you can still be productive without a deep understanding of calculus, linear algebra, etc.

Ok, so you don't need to know that much math to get stared, but you're not entirely off the hook. As I noted above, you still need to know how to use the tools properly.

In some sense, this is one of the challenges of using machine learning tools in R: many of them be difficult to use.

R has many packages for implementing various machine learning methods, but unfortunately many of these tools were designed separately, and they are not always consistent in how they work. The syntax for some of the machine learning tools is very awkward, and syntax from one tool to the next is not always the same. If you don't know where to start, machine learning in R can become very confusing.

This is why I recommend using the `caret` package to do machine learning in R.

A quick introduction to `caret`

For starters, let's discuss what `caret` is.

The `caret` package is a set of tools for building machine learning models in R. The name "`caret`" stands for Classification And REgression Training. As the name implies, the `caret` package gives you a toolkit for building classification models and regression models. Moreover, `caret` provides you with essential tools for:

- Data preparation, including: imputation, centering/scaling data, removing correlated predictors, reducing skewness
- Data splitting
- Model evaluation
- Variable selection

`caret` simplifies machine learning in R

While `caret` has broad functionality, the real reason to use caret is that it's simple and easy to use.

As noted above, one of the major problems with machine learning in R is that most of R's different machine learning tools have different interfaces. They almost all "work" a little differently from one another: the syntax is slightly different from one modeling toolkit to the next; tools for different parts of the machine learning workflow don't always "work well" together; tools for fine tuning models or performing critical functions may be awkward or difficult to work with. Said succinctly, R has many machine learning tools, but they can be extremely clumsy to work with.

`caret` solves this problem. To simplify the process, `caret` provides tools for almost every part of the model building process, and moreover, provides a common interface to these different machine learning methods.

For example, `caret` provides a simple, common interface to almost every machine learning algorithm in R. When using `caret`, different learning methods like linear regression, neural networks, and support vector machines, all share a common syntax (the syntax is basically identical, except for a few minor changes).

Moreover, additional parts of the machine learning workflow – like cross validation and parameter tuning – are built directly into this common interface.

To say that more simply, `caret` provides you with an easy-to-use toolkit for building many different model types and executing critical parts of the ML workflow. This simple interface enables rapid, iterative modeling. In turn, this iterative workflow will allow you to develop good models faster, with less effort, and with less frustration.

`caret`'s syntax

Now that you've been introduced to `caret`, let's return to the example above (of `mpg` vs `wt`) and see how `caret` works.

Again, imagine you want to learn the relationship between `mpg` and `wt`. As noted above, in mathematical terms, this means identifying a function, $f(x)$, that describes the relationship between `wt` and `mpg`.

Here in this example, we're going to make an additional assumption that will simplify the process somewhat: we're going to assume that the relationship is linear; we'll assume that it can be described by a straight line of the form $f(x) = \beta_0 + \beta_1 x$.

In terms of our modeling effort, this means that we'll be using linear regression to build our machine learning model.

Without going into the deep details of linear regression, let's look at how we implement linear regression with `caret`.

The `train()` function

The core of `caret`'s functionality is the `train()` function.

`train()` is the function that we use to "train" the model. That is, `train()` is the function that will "learn" the relationship between `mpg` and `wt`.

Let's take a look at this syntactically.

Here is the syntax for a linear regression model, regressing `mpg` on `wt`.

```
#~~~~~~~~~~~~~~~~~~~~~~~~~
# Build model using train()
#~~~~~~~~~~~~~~~~~~~~~~~~~

library(caret)

model.mtcars_lm <- train(mpg ~ wt
                         ,data = mtcars
                         ,method = "lm"
                         )
```

That's it. The syntax for building a linear regression is extremely simple with `caret`.

Now that we have a simple model, let's quickly extract the regression coefficients and plot the model (i.e., plot the linear function that describes the relationship between `mpg` and `wt`).

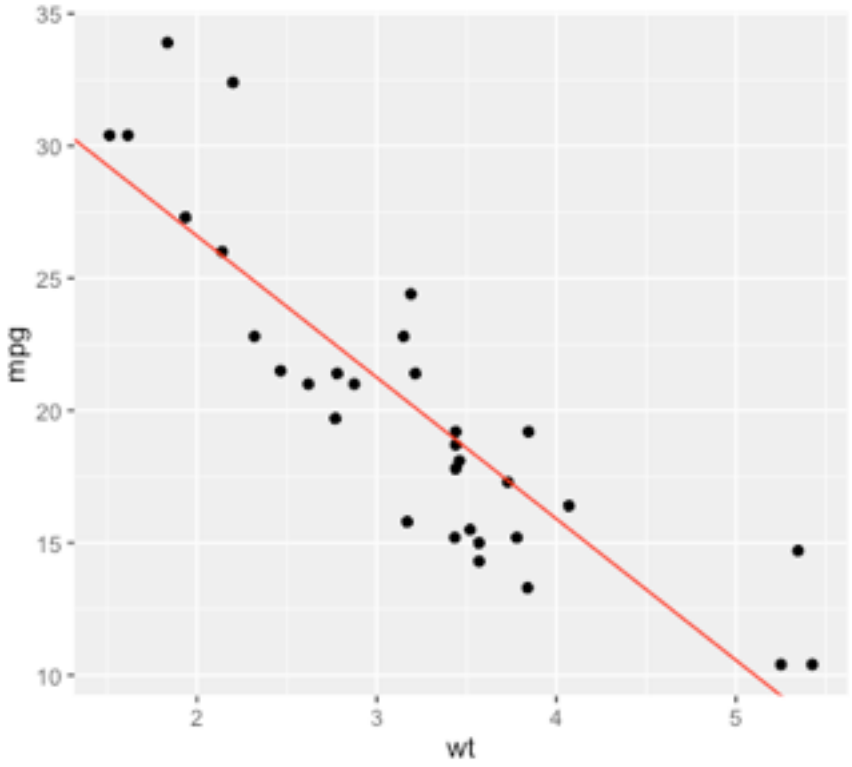
Retrieve regression coefficients and plot the model

```
#~~~~~~~~~~~~~~~~~~~~~~~~~
# Retrieve coefficients for
#  - slope
#  - intercept
#~~~~~~~~~~~~~~~~~~~~~~~~~

coef.icept <- coef(model.mtcars_lm$finalModel)[1]
coef.slope <- coef(model.mtcars_lm$finalModel)[2]


#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Plot scatterplot and regression line
#  using ggplot()
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

ggplot(data = mtcars, aes(x = wt, y = mpg)) +
  geom_point() +
  geom_abline(slope = coef.slope, intercept = coef.icept, color = "red")
```



How caret's syntax works

Now, let's look more closely at the syntax and how it works.

When training a model using `train()`, you only need to tell it a few things:

- The dataset you're working with
- The target variable you're trying to predict (e.g., the `mpg` variable)
- The input variable (e.g., the `wt` variable)
- The machine learning method you want to use (in this case "linear regression")

Formula notation

In `caret`'s syntax, you identify the target variable and input variables using the "formula notation."

The basic syntax for formula notation is y ~ x, where y is your target variable, and x is your predictor.

Effectively, y ~ x tells `caret` "I want to predict y on the basis of a single input, x."

Now, with this knowledge about `caret`'s formula syntax, let's reexamine the above code. Because we want to predict `mpg` on the basis of `wt`, we use the formula `mpg ~ wt`. Again, this line of code is the "formula" that tells `train()` our target variable and our input variable. If we translate this line of code into English, we're effectively telling `train()`, "build a model that predicts `mpg` (miles per gallon) on the basis of `wt` (car weight)."

The `data =` parameter

The `train()` function also has a `data =` parameter. This basically tells the `train()` function what dataset we're using to build the model. Said differently, if we're using the formula `mpg ~ wt` to indicate the target and predictor variables, then we're using the `data =` parameter to tell `caret` where to find those variables.

So basically, `data = mtcars` tells the `caret` function that the data and the relevant variables can be found in the `mtcars` dataset.

The `method =` parameter

Finally, we see the `method =` parameter. This parameter indicates what machine learning method we want to use to predict y. In this case, we're building a linear regression model, so we are using the argument `"lm"`.

Keep in mind, however, we could select a different learning method. Although it's beyond the scope of this book to discuss all of the possible learning methods that we could use here, there are many different methods we could use. For example, if we wanted to use the k-nearest neighbor technique, we could use the `"knn"` argument instead. If we did that, `train()` would still predict `mpg` on the basis of `wt`, but would use a different statistical technique to make that prediction. This would yield a different model; a model that makes different predictions.

Again, it's beyond the scope of this book to discuss all of the different model types. However, as you learn more about machine learning, and want to try out more advanced machine learning techniques, this is how you can implement them. You simply change the learning method by changing the argument of the `method =` parameter.

This is a good place to reiterate one of `caret`'s primary advantages: switching between model types is extremely easy when we use `caret`'s `train()` function. Again, if you want to use linear regression to model your data, you just type in `"lm"` for the argument to `method =`; if you want to change the learning method to k-nearest neighbor, you just replace `"lm"` with `"knn"`.

`caret`'s syntax allows you to very easily change the learning method. In turn, this allows you to "try out" and evaluate many different learning methods rapidly and iteratively. You can just re-run your code with different

values for the method parameter, and compare the results for each method.

Next steps

Now that you have a high-level understanding of `caret`, you're ready to dive deeper into machine learning in R.

Keep in mind though, if you're new to machine learning, there's still lots more to learn. Machine learning is intricate, complex, and fascinating. Moreover, `caret` has a variety of additional tools for model building. We've just scratched the surface here.

What's the difference between machine learning, statistics, and data mining?

In the last few chapters, you've learned a little bit about machine learning, why it's valuable, and how to get started with machine learning in R.

Throughout those chapters, I've been using the following definition of machine learning: creating computational systems that learn from data in order to make predictions and inferences.

However, machine learning isn't the only subject in which we use data for prediction and inference. Anyone who's taken an introductory statistics class has heard a similar definition about statistics itself. And if you talk to someone who works in data-mining, you'll hear the same thing: data mining is about using data to make predictions and draw conclusions from data.

This raises the question: what is the difference between machine learning, statistics, and data mining?

The long answer has a bit of nuance (which we'll discuss soon), but the short answer answer is very simple: machine learning, statistical learning, and data mining are almost exactly the same.

An expert opinion: there is no difference

Larry Wasserman wrote a blog post about this a few years ago. If you're not familiar with him, Wasserman is a professor in both the Department of Statistics and in the Machine Learning Department at Carnegie Mellon, one of the premier universities for stats and ML. Moreover, I'll point out that Carnegie Mellon is one of the few universities that has a department devoted exclusively to machine learning. Given that Wasserman is officially a professor in ML and statistics at one of the best STEM

universities in the world, I'd say that he's uniquely positioned to answer this question.

According to Wasserman, what's the difference?

According to Wasserman, "the short answer is: None. They are … concerned with the same question: how do we learn from data?"

To be clear, Wasserman was specifically addressing the difference between "machine learning" and "statistics" (he didn't mention data mining). However, I'll add that his answer applies equally well to "data mining".

So to echo what Wasserman wrote, and re-state the point: machine learning, statistics, and data mining are mostly the same.

Having said that, Wasserman notes that if you look at some of the details, there is a "more nuanced" answer that reveals minor differences.

With that in mind, let's review the major similarities as well as the differences, both to prove the point that they really are extremely similar, and also to give you the more nuanced view.

The core similarities between ML, stats, and data mining

First, let's examine the what makes machine learning, statistics, and data mining fundamentally similar.

Nearly identical subject-matter and toolkits

The primary reason that these three subjects are effectively the same is that they cover almost exactly the same material and use almost exactly the same techniques.

Let me give you an example. If you examine the table of contents of the book *An Introduction to Statistical Learning*, you'll find chapters on regression, classification, resampling, and non-linear methods. If you

examine the contents a bit more closely, you'll see sections concerning linear regression, logistic regression, the bias-variance tradeoff, neural networks, and support vector machines.

Next, if you look at the syllabus for Andrew Ng's *machine learning* course on Coursera, and you'll see all of the topics I just listed. The material covered in Ng's machine learning course is almost exactly the same as the material covered in ISL; said differently, the material covered by the "statistical learning" experts is the same as that covered by the "machine learning" experts. With only minor exceptions, the material is exactly the same.

If you perform the same exercise and look at the table of contents for the book *Data Mining* by Witten, Frank, and Hall, you'll find almost exactly the same material. Again: regression, classification, neural networks, etc.

So to summarize, the material covered in machine learning, statistics (and statistical learning in particular), and data mining are so similar that they're nearly identical.

The three cultures: why there are three identical subjects with different names

So what's going on here? They're nearly identical?! Why are there three different subjects that cover the exact same material?

The best answer is that even though they use the same methods, they evolved as different cultures, so they have different histories, nomenclature, notation, and philosophical perspectives.

Let me give you an analogy: "machine learning" and "statistics" are like identical twins (triplets, if you want to include "data mining" in the analogy). They are quite nearly identical. And in fact, at a deep level, they are identical: they "share the same DNA" as it were.

However, even though identical twins are identical in some sense, they might still dress differently and hang out with different people.

To relate this to the topic of ML, statistics, and data mining, what I mean by "dressed differently" is that they use different words and terms, and have different notation. Much like human twins that have different groups of friends, people in ML, stats, and data mining also have different and separate social groups: they exist in different academic departments (in most universities), typically publish in different journals, and have different conferences and events.

The core differences between ML, stats, and data mining

Let's examine these differences a little more closely.

They emphasize different things

Perhaps the biggest difference between these three fields is their emphasis. Although they use almost the exact same methods, they tend to

emphasize different things. A different way of saying this, is that although they use almost exactly the same methods, tools, and techniques, these three fields have philosophical differences concerning how and when those methods should be applied.

In his blog post, Wasserman mentioned these different emphases, noting that “statistics emphasizes formal statistical inference (confidence intervals, hypothesis tests, optimal estimators) in low dimensional problems.” (To be clear, in this quote, Wasserman seems to be talking about statistics broadly, and not statistical learning in particular.)

Wasserman went on to note that machine learning is more focused on making accurate predictions – a sentiment echoed by Professor Andrew Gelman of Columbia University. Because of this purported emphasis on prediction above all, some people characterize ML as being less “strict” about testing assumptions; that in ML, making good predictions trumps more formal considerations like testing assumptions, etc.

Machine learning is focused on software and systems

Actually, I'll go a step further and state that machine learning isn't just more focused on making predictions, but is more focused on building software systems that make predictions. That is, among ML practitioners (as opposed to statisticians), there is a much stronger emphasis on software engineering. This makes sense considering the historical growth of machine learning as a sub-discipline of computer science; historically, machine learning developed because computer scientists needed a way to create computer programs that learn from data.

This greater emphasis on systems (i.e., computer programs that learn from data) leads some people to argue that ML is more of an "engineering discipline" whereas statistics is more of a "mathematical" discipline. Again though, I'll repeat my caveat that these are very broad generalizations.

The purpose of data mining is finding patterns in databases

How about data mining?

As I already noted, the tools that data miners use are almost exactly the same as the tools used by statisticians and machine learning experts. The major difference is how and why these tools are used. So how do data miners apply these techniques, and to what end?

Here's its quite helpful to look at the name "data mining."

As the name implies, "data mining" is akin to physical mining. In a physical mining operation – for example a gold mining operation – large piles of dirt and material are extracted from the mine and then the miners sift through the dirt to find nuggets of gold. Here you can think of a data warehouse as the mine: it contains mostly useless data. This useless data is like the "dirt and rubble" in a mine. Then after being extracted from the database, this "useless" data needs to be "mined" for valuable insight.

Sometimes, finding these insights requires simple exploratory data analysis. Sometimes it requires more formal statistical techniques like hypothesis testing. And sometimes it requires building models – basically the same types of models used in machine learning and statistical learning.

So again, we find that the tools are almost identical, but the purpose is slightly different: finding valuable insights in large databases.

To summarize: that although ML, stats, and data mining use the same methods, they have slightly different philosophies about how, when, and why to apply those methods.

They use different words and terminology

Not only do they have slightly different emphases and purposes, they also use different terminology.

Professor Rob Tibshirani – one of the authors of the excellent book *An Introduction to Statistical Learning* – created a glossary comparing several major terms in machine learning vs statistics.

Machine learning	Statistics
network, graphs	model
weights	parameters
learning	fitting
generalization	test set performance
supervised learning	regression/classification
unsupervised learning	density estimation, clustering
large grant = $1,000,000	large grant = $50,000
nice place to have a meeting: Snowbird, Utah, French Alps	nice place to have a meeting: Las Vegas in August

While a few of these comparisons are a bit tongue-in-cheek, the list is instructive. Most specifically, the chart drives home the point I've been making in this post: machine learning and statistics are quite nearly identical. They are so similar that they discuss almost all of the same

topics, methods, and concepts, but just have different names for many of them.

Having said that, I want to add that frequently these terms are used interchangeably by people in stats and machine learning (as well as data mining).

For example, if you examine Andrew Ng's machine learning course, you'll find that he frequently uses the term "parameter" instead of "weight". Although most people would consider Ng a member of the "machine learning culture", but he readily uses the terms attributed to the "statistical learning culture." So, it appears that even though there are slight cultural differences, even those differences are insubstantial; members of the "different cultures", basically use terminology interchangeably.

ML, stats, and data mining tend to favor different tools

Next, let's talk about tools. There are some very rough generalizations we can make about tool choices between statisticians, data miners, and machine learning practitioners (but as I've pointed out several times, these are sort of hasty generalizations).

In the machine learning camp, you're more likely to see Python and Matlab. If you do a quick review of the major courses and texts on machine learning, many of them use one of these two languages. For example, Andrew Ng's course teaches ML labs in Matlab, as do several other excellent books by authors such as Barber or Murphy.

Outside of academia, and particularly in San Francisco/Silicon Valley, you'll often see Python as a primary ML tool of choice.

On the statistics side, you're much more likely to see R. You're very likely to see R used as the primary programming language for probability and

statistics classes at colleges and universities. Also, several of the best books on probability, statistics, Bayesian statistics, and statistical learning use R for their exercises and labs.

I'll repeat though that these are very hasty generalizations. You'll see statisticians using Matlab, and you'll certainly see machine learning practitioners using R. Moreover, as ML, stats, and data mining practitioners begin to cooperate and these fields begin to converge, you're seeing people learn several tools. Many of the best people in stats and ML know python and R, as well as other tools.

And of course, the list certainly isn't limited to Python, R, and Matlab. In both academia and industry, you'll find statisticians, ML experts, and data miners using a broad array of other technologies like SAS, SPSS, c++, Java and others.

So while I'll suggest that Python and Matlab are more popular among the ML crowd and that R is more popular among statisticians, again, this is really just a quick-and-dirty generalization.

ML and data mining typically work on "bigger" data than statistics

Finally, let's talk briefly about the size and scale of the problems these different groups work on.

The general consensus among several of the prominent professors mentioned above is that machine learning tends to emphasize "larger scale" problems than statistics.

Wasserman noted in his blog post that in contrast to statistics, machine learning emphasizes "high dimension" prediction problems (which presumably means that machine learning emphasizes problems with a larger number of predictor variables).

Andrew Gelman – who is also a very well respected professor of statistics – similarly implies that statistics emphasizes smaller scale problems, He said, “My impression is that computer scientists work on more complicated problems and bigger datasets, in general, than statisticians do.”

And finally, data mining also emphasizes large scale data. While I won’t offer any quotes here, I can say from personal experience that people who identify as “data miners” commonly work with databases that have millions, even hundreds of millions or billions of observations (although, it’s quite common to subset these large datasets down, to take samples, etc).

Again: there are far more similarities than differences

It should seem clear by now that machine learning, statistics, and data mining are all fundamentally similar. And to the extent that they have differences, those differences are – in the words of Revolution Analytics' David Smith – more superficial than substantive.

Moreover, even though there are differences – particularly philosophical differences – you need to realize that these fields are beginning to converge. As Wasserman noted in his blog article, it's clear that members of these communities are beginning to communicate with one another, and the lines between these fields are becoming increasingly blurred. So even if we can identify some differences, those differences may become much less pronounced over time.

What this means for you, if you're getting started with data science, is that you can safely treat machine learning, statistics, and data mining as "the same." Learn the "surface level" differences so you can communicate with people from the "different cultures," but ultimately, treat machine learning,

statistics, and data mining each as subjects that you can learn from as you work to master data science.

To master data science, you need to practice

There you have it. In this book, you've seen almost everything that you need to get started with data science.

We've covered all of the essentials: data visualization, data manipulation, and machine learning.

Having said that though, you can't learn data science in a day.

If you want to master data science, you need to practice.

Do you want to know how to practice? Do you want to discover how to learn the basics so that you *never forget them*?

That's what you need. In order to get a data science job, you need to master the basic charts and graphs. You need to master data manipulation.

You need to be able to write the code for these things "with your eyes closed."

So, review the material in this book. Run the code. Continue reading the blog posts at sharpsightlabs.com.

But when you're ready to get to the next level, you'll need a system for *practicing* data science. You'll need a system to develop intuition for the essential concepts. You'll need a system for practicing the syntax.

Sharp Sight has just such a system. Several times per year, we open the doors to a course that will show you how to practice. A course that will help you absolutely master data visualization and data manipulation.

Interested? Watch your inbox …

Postscript: Send us your questions and problems

What other questions do you have about data science?

What are you struggling with?

What courses have you taken, and why aren't they working for you?

Email your questions and problems directly to: josh@sharpsightlabs.com