Lecture 3: Introduction to Data - Part 2

LSE ME314: Introduction to Data Science and Machine Learning (https://github.com/me314-lse)

2025-07-16

Daniel de Kadt

Slides last updated: 15 July 2025

Where Do Data Come From?

Data Sources

Typically we think of data as coming from a ‘source:’

- API
- Database
- Users
- Surveys
- Experiments

Data Sources

Typically we think of data as coming from a 'source:'

- API
- Database
- Users
- Surveys
- Experiments

But where do data *really* come from?

Adventure Awaits!

Source: Wizards of the Coast, DND Beginners Guide

Adventure Awaits!

Gather your party, and venture forth to find out...

```r
# View our characters:
head(dnd_characters)
```

```
             Name     Class       Race     Gender Strength Constitution
1          Thinos Barbarian   Tiefling     Female        9           14
2           Theax Barbarian      Dwarf       Male        8           13
3         Thuldor   Fighter      Human       Male       10           14
4 Nyax Stormborn Barbarian   Halfling     Female       10           13
5         Droador     Rogue        Elf       Male       15           12
6         Korinar Barbarian Dragonborn Non-binary       15           14
  Dexterity Wisdom Charisma Intelligence Hitpoints InitiativeMod
1        15     13       12           14        14             2
2         9     10       11            8        13            -1
3        14     10        7           10        12             2
4        10     13       13            9        13             0
5        16     14       11            9         9             3
6        13     14       18            8        14             1
```

Probability

What is a Probability?

Random variable: A variable that is subject to probability.

What is a Probability?

Random variable: A variable that is subject to probability.

Sample space: The set of all possible outcomes.

Event: A subset of the sample space - something that could occur.

Outcome: A realization of the random variable - what happens.

What is a Probability?

Random variable: A variable that is subject to probability.

Sample space: The set of all possible outcomes.

Event: A subset of the sample space - something that could occur.

Outcome: A realization of the random variable - what happens.

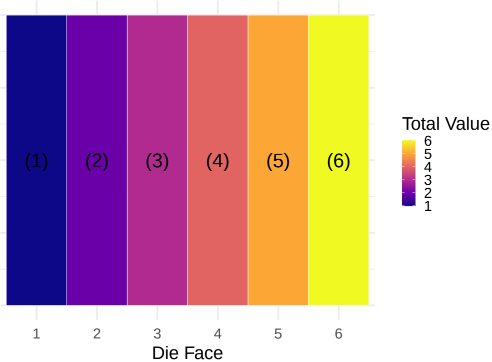
Probability: The chance that the outcome will be a particular event in the sample space.

Sample Space

A **sample space** is denoted by Ω:

```
# Create sample space of a single roll of a D6:
df_single <- data.frame(
  rollvalue = factor(1:6),
  row = 1
  )
```

Here $\Omega = \{1, 2, 3, 4, 5, 6\}$ and each value is an event. Visually:



Events, Probabilities, and the Probability Axioms

Consider one event A in the sample space Ω.

This could be one outcome of a roll, e.g. A is defined as $roll1 = 3$

The probability of A, denoted $P(A)$ (or $P(roll1 = 3)$)

Events, Probabilities, and the Probability Axioms

Consider one event A in the sample space Ω.

This could be one outcome of a roll, e.g. A is defined as $roll1 = 3$

The probability of A, denoted $P(A)$ (or $P(roll1 = 3)$)

Ω and all events therein (including A) are constrained by three axioms (Kolmogarov):

1. **Non-negativity**: $P(A) \geq 0$.
2. **Normalization**: $P(\Omega) = 1$.
3. **Additivity**: If $A_1, A_2, \ldots, A_n$ are mutually exclusive events, then $P(A_1 \cup A_2 \cup \ldots \cup A_n) = P(A_1) + P(A_2) + \ldots + P(A_n)$.

Joint Sample Spaces

What if we have two probabilistic processes?

Joint Sample Spaces

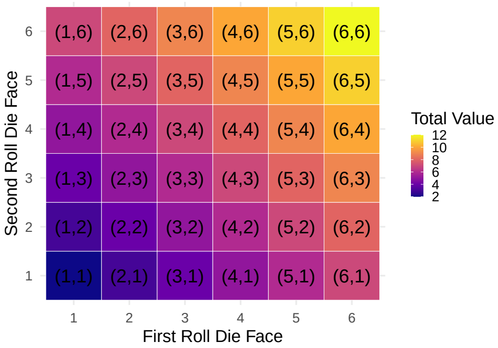
What if we have two probabilistic processes?

```r
# Create sample space for two rolls of a D6:
df_double <- expand.grid(
  roll1value = 1:6,
  roll2value = 1:6
)
```

Joint Sample Spaces

What if we have two probabilistic processes?

```
# Create sample space for two rolls of a D6:
df_double <- expand.grid(
  roll1value = 1:6,
  roll2value = 1:6
)
```



Joint Sample Spaces

With multiple probabilistic processes occuring together, we are describing a joint sample space.

We can reason about $P(\text{roll1} = x1)$ or $P(\text{roll2} = x2)$ or:

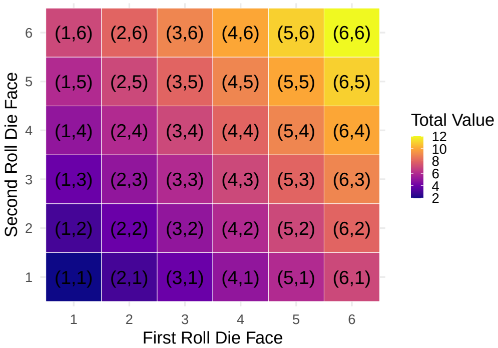
- Union (either or): $P(\text{roll1} = x1 \cup \text{roll2} = x2)$
- Intersection (both): $P(\text{roll1} = x1 \cap \text{roll2} = x2)$

Joint Sample Spaces

With multiple probabilistic processes occuring together, we are describing a joint sample space.

We can reason about $P(\text{roll1} = x1)$ or $P(\text{roll2} = x2)$ or:

- Union (either or): $P(\text{roll1} = x1 \cup \text{roll2} = x2)$
- Intersection (both): $P(\text{roll1} = x1 \cap \text{roll2} = x2)$



Probability Distributions

A random variable is drawn from a probability distribution.

Probability distributions are defined by:

1. **Support**: The set of values that the variable can take
2. **Function**:
   - **Probability Mass Function (PMF)**: For discrete variables, the PMF gives the exact probability of each possible outcome
   - **Probability Density Function (PDF)**: For continuous variables, gives a relative likelihood of possible outcomes in an interval (not a probability!)
   - **Parameters**: : Values that determine the shape and characteristics of the PMF or PDF
3. **Cumulative Distribution Function (CDF)**: The CDF gives the probability that the variable is less than or equal to a given value ($P(X \leq x)$).

Probability Distributions: Uniform (Discrete)

For random variable X:

We denote $X \sim \mathcal{U}(a, b)$

Probability Distributions: Uniform (Discrete)

For random variable X:

We denote $X \sim \mathcal{U}(a, b)$

This means "the random variable X follows a uniform distribution between a and b"

Or "the random variable X is drawn from a uniform distribution with support $[a, b]$"

(a and b are the lower and upper boundaries of support)

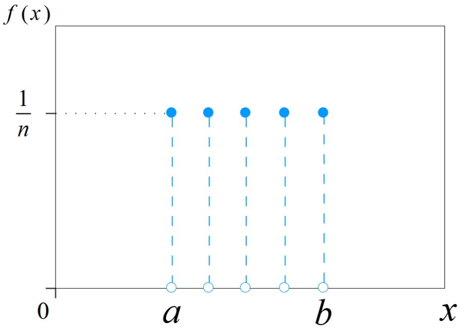
Probability Distributions: Uniform (Discrete)

For random variable X: $X \sim \mathcal{U}(a, b)$:

Probability Distributions: Uniform (Discrete)

For random variable X: $X \sim \mathcal{U}(a, b)$:

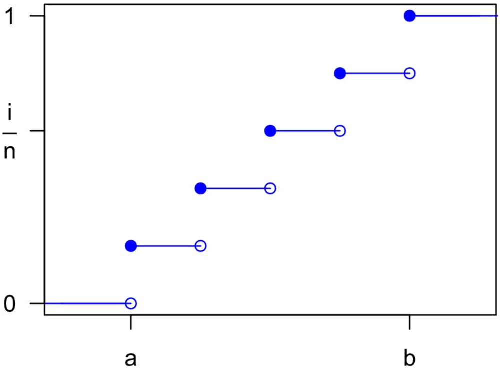
The PMF is written as: $P(X = x) = \frac{1}{n}$ where n is the number of unique values in the support.



Source: Wikipedia, Discrete Uniform Distribution

Probability Distributions: Uniform (Discrete)

The CDF looks like this:



Source: Wikipedia, Discrete Uniform Distribution

Probability Distributions: Binomial

The binomial distribution models the outcome of a series of independent Bernoulli trials (coin flips)

Each trial has two possible outcomes: heads (with probability p) or tails (with probability $1 - p$)

Probability Distributions: Binomial

The binomial distribution models the outcome of a series of independent Bernoulli trials (coin flips)

Each trial has two possible outcomes: heads (with probability p) or tails (with probability $1 - p$)

The total number of trials, n, is fixed, and the trials are independent of each other (what you get in one coin flip has no effect on any other coin flip)

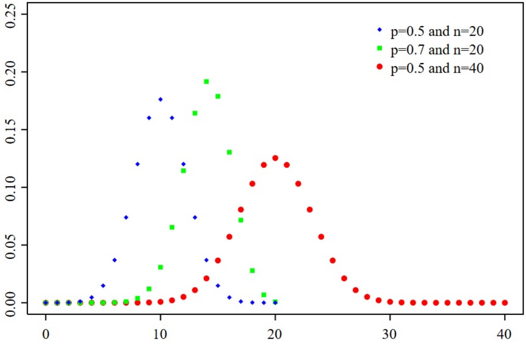
For random variable X: $X \sim \mathcal{B}(n, p)$.

Probability Distributions: Binomial

The PMF is given by:

$$P(X = k) = \binom{n}{k} p^k (1-p)^{n-k}$$

where $\binom{n}{k}$ is the binomial coefficient, which counts the number of ways to choose k successes in n trials.



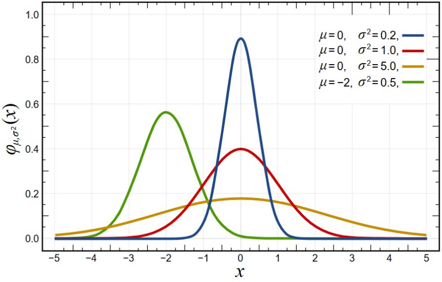
Source: Wikipedia, Binomial Distribution

Probability Distributions: Normal or Gaussian

A normal distribution is a continuous probability distribution.

For random variable X: $X \sim \mathcal{N}(\mu, \sigma^2)$, where μ is the mean, and σ^2 is the variance.

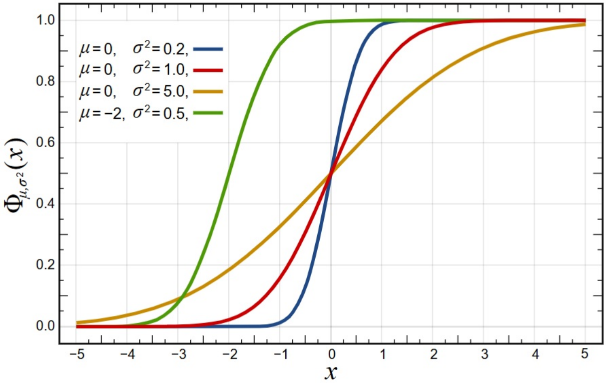
The PDF is given by $\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$:



Source: Wikipedia, Normal Distribution

Probability Distributions: Normal or Gaussian

And the CDF of the normal:



Source: Wikipedia, Normal Distribution

Probability Distributions in R

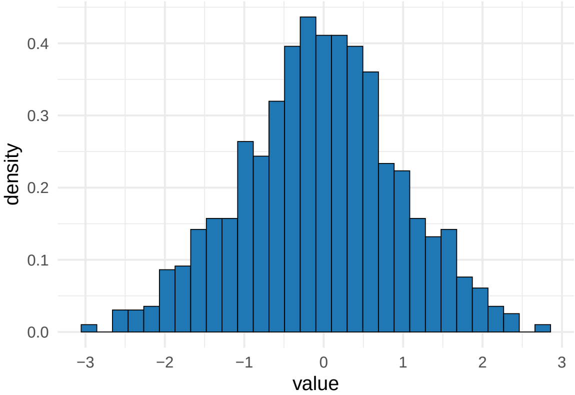
Let's sample from a normal distribution in R using `rnorm()`, setting the mean and standard deviation:

```
library(ggplot2); library(dplyr)

normal_sample <- data.frame(value = rnorm(1000, mean = 0, sd = 1))

normal_hist <- normal_sample %>%
  ggplot(aes(x = value)) +
  geom_histogram(aes(y = ..density..), bins = 30,
                 fill = "#1f77b4", color = "black") +
  theme_minimal(base_size=24)
```

What's going on here? Each value in `normal_sample$value` is a single draw from a normal distribution with mean 0 and standard deviation 1.

We then plot a histogram of the drawn values.

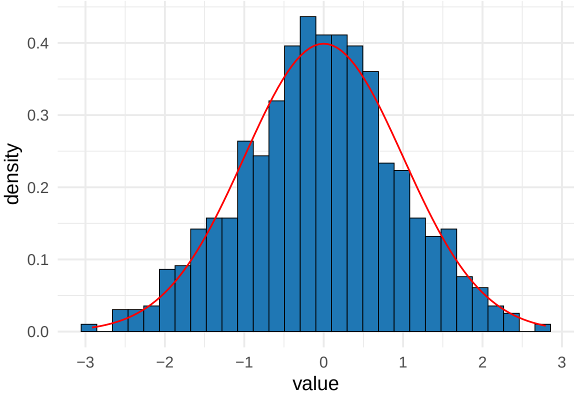
Probability Distributions in R



Probability Distributions in R

Let's overlay the actual probability density function for the normal distribution ($\mu = 0, \sigma^2 = 1$):

```
normal_hist + # this is the previous ggplot2 object, now we add a layer
    stat_function(fun = dnorm, args = list(mean = 0, sd = 1),
                  color = "red", size = 1)
```



Probability Distributions in R

You can do similar things for other distributions:

- `runif()`: randomly draw from a uniform distribution
- `dunif()`: density for a uniform distribution
- `rbinom()`: randomly draw from a binomial distribution
- `dbinom()`: density for a binomial distribution
- `rpoisson()`: randomly draw from a Poisson distribution
- etc.

Each function takes different arguments depending on the parameters required.

Some Theory: Expected Values

A random variable is said to have an **expected value** of $\mathbb{E}(X)$, which is the average value of the random variable over very many trials.

Intuitively you can think of this as the mean of all the possible values X could take, weighted by the probability of each value being realized.

Some Theory: Expected Values

A random variable is said to have an **expected value** of $\mathbb{E}(X)$, which is the average value of the random variable over very many trials.

Intuitively you can think of this as the mean of all the possible values X could take, weighted by the probability of each value being realized.

The expected value of:

- a uniform distribution: $\mathbb{E}(X) = \frac{a+b}{2}$.
- a normal distribution: $\mathbb{E}(X) = \mu$.
- a binomial distribution: $\mathbb{E}(X) = n \cdot p$.

Some Theory: Expected Values

Let's demonstrate this in `R`:

Some Theory: Expected Values

Let's demonstrate this in R:

```r
# Sample 1000 times from a uniform dist with a = 0, b = 1
draw <- runif(1000, min = 0, max = 10)
# Calculate the average (should be close to `(a+b)/2`)
mean(draw)
```

```
[1] 4.88331
```

Some Theory: Expected Values

Let's demonstrate this in R:

```
# Sample 1000 times from a uniform dist with a = 0, b = 1
draw <- runif(1000, min = 0, max = 10)
# Calculate the average (should be close to `(a+b)/2`)
mean(draw)
```

```
[1] 4.88331
```

```
# Sample 1000 times from a binomial dist with n = 1, p = 0.5
draw <- rbinom(1000, size = 1, prob = 0.5)
# Calculate the average (should be close to `n`*`p`)
mean(draw)
```

```
[1] 0.48
```

Some Theory: Expected Values

Let's demonstrate this in R:

```
# Sample 1000 times from a uniform dist with a = 0, b = 1
draw <- runif(1000, min = 0, max = 10)
# Calculate the average (should be close to `(a+b)/2`)
mean(draw)
```

```
[1] 4.88331
```

```
# Sample 1000 times from a binomial dist with n = 1, p = 0.5
draw <- rbinom(1000, size = 1, prob = 0.5)
# Calculate the average (should be close to `n`*`p`)
mean(draw)
```

```
[1] 0.48
```

```
# Sample 1000 times from a normal dist with mean = 5, sd = 5
draw <- rnorm(1000, mean = 5, sd = 5)
# Calculate the average (should be close to `mean`)
mean(draw)
```

```
[1] 5.059436
```

Some Theory: Law of Large Numbers

What we just saw is an example of the **Law of Large Numbers** (LLN).

As the number of i.i.d* observations in our sample increases, the sample mean converges to the true expected value:

$\bar{X}_n \longrightarrow \mathbb{E}(X)$ as $n \to \infty$

where $\bar{X}_n$ is the sample mean of n realisations of the random variable X.

*i.i.d = independent and identically distributed

Some Theory: Central Limit Theorem

Let's connect the LLN to a deeper result: the **Central Limit Theorem** (CLT).

Over K repeated samples each of size n, the *distribution* of the sample mean $\bar{X}_k$ approaches a normal distribution as $K \to \infty$.

This is true regardless of the underlying probability distribution from which X is drawn.

This will be the backbone of asymptotic statistical inference (tomorrow).

Some Theory: Central Limit Theorem

Let's do the following simulation for K repeated samples:

1. Pick a probability distribution to generate X
2. K times:
   2.1 Draw one sample of size n: $X_{1k}, X_{2k}, \ldots, X_{nk}$
   2.2 Calculate and store the sample mean $\bar{X}_k$
3. Plot the distribution of all K sample means.

Theory in Practice

We'll do the simulation with our favourite dice, the D6:

```r
d6_sampler <- function(K, n){

  sample_mean <- c()

  for(k in 1:K){
    # roll a D6 `n` times to create an i.i.d sample
    sample <- sample(1:6, n, replace = TRUE)
    # calculate the sample mean
    sample_mean[k] <- mean(sample)
  }

  return(sample_mean)
}
```

Theory in Practice

We'll do the simulation with our favourite dice, the D6:

```
d6_sampler <- function(K, n){

  sample_mean <- c()

  for(k in 1:K){
    # roll a D6 `n` times to create an i.i.d sample
    sample <- sample(1:6, n, replace = TRUE)
    # calculate the sample mean
    sample_mean[k] <- mean(sample)
  }

  return(sample_mean)
}
```

Start with just a single draw of 100 rolls – this is the LLN in action:

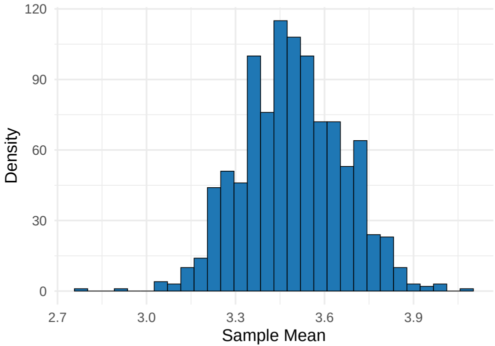
```
d6_sampler(K = 1, n = 100)
```

```
[1] 3.29
```

Theory in Practice

Now draw 1000 samples each of size 100 – this is the CLT in action:

```
many_d6_samples <- d6_sampler(K = 1000, n = 100)
many_d6_samples %>%
  as.data.frame() %>%
  ggplot(aes(x = many_d6_samples)) +
  geom_histogram(bins = 30, fill = "#1f77b4", color = "black") +
  labs(x = "Sample Mean", y = "Density") +
  theme_minimal(base_size=24)
```



So, Where Do Data *Really* Come From?

Data Generating Processes

All data we encounter comes from an underlying **Data Generating Processes** (DGPs).

We (usually) cannot see these processes, and typically only observe *one* draw (sample).

Data Generating Processes

All data we encounter comes from an underlying **Data Generating Processes** (DGPs).

We (usually) cannot see these processes, and typically only observe *one* draw (sample).

We often want to understand how our outcome variable Y responds to changes in some set of features $X_1, X_2, \ldots, X_n$.

Building a model of Y is us making some *assumptions* (often parametric) about the DGP that generated Y.

These may or may not be good assumptions.

Data Generating Processes

Consider this **Conditional Expectation Function** (CEF):

$$\mathbb{E}(Y \mid X_1, X_2, \ldots, X_n) = f(X_1, X_2, \ldots, X_n)$$

where $f(\cdot)$ is some function that describes the relationship between the features and the outcome.

Evaluated for any particular $X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n$, this returns the expected value of Y.

Data Generating Processes

Consider this **Conditional Expectation Function** (CEF):

$$\mathbb{E}(Y \mid X_1, X_2, \ldots, X_n) = f(X_1, X_2, \ldots, X_n)$$

where $f(\cdot)$ is some function that describes the relationship between the features and the outcome.

Evaluated for any particular $X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n$, this returns the expected value of Y.

What should $f(\cdot)$ look like?

Data Generating Processes

Consider this **Conditional Expectation Function** (CEF):

$$\mathbb{E}(Y \mid X_1, X_2, \ldots, X_n) = f(X_1, X_2, \ldots, X_n)$$

where $f(\cdot)$ is some function that describes the relationship between the features and the outcome.

Evaluated for any particular $X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n$, this returns the expected value of Y.

What should $f(\cdot)$ look like? Well, that is the million dollar question.

Data Generating Processes

We will think about DGPs as a function with different possible components:

- **Deterministic**:
  - Typically 'fixed' or 'given'
- **Probabilistic**:
  - Product of a probabilistic process

These probabilistic components can themselves be of different types:

- *Systematic*:
  - Product of other variables, of which at least one is probabilistic
- *Stochastic*:
  - Random processes independent of other features

Data Generating Processes

We will think about DGPs as a function with different possible components:

- **Deterministic**:
  - Typically 'fixed' or 'given'
- **Probabilistic**:
  - Product of a probabilistic process

These probabilistic components can themselves be of different types:

- *Systematic*:
  - Product of other variables, of which at least one is probabilistic
- *Stochastic*:
  - Random processes independent of other features

Deep philosophical point: Is anything ever actually random?

(Probably not)

Data Generating Processes

Let's return to the CEF of Y:

- Assume each of $X_1, X_2, \ldots, X_3$ is drawn from a probability distribution
- If Y is determined by these variables, then Y is a probabilistic function of these underlying distributions

This can become extremely complicated – we could allows $X_1, X_2, \ldots, X_n$ to be correlated or jointly drawn from a multivariate probability distribution.

Data Generating Processes

Let's return to the CEF of Y:

- Assume each of $X_1, X_2, \ldots, X_3$ is drawn from a probability distribution
- If Y is determined by these variables, then Y is a probabilistic function of these underlying distributions

This can become extremely complicated – we could allows $X_1, X_2, \ldots, X_n$ to be correlated or jointly drawn from a multivariate probability distribution.

Key insight: One DGP could produce very many unique datasets. But we only observe one.

DGPs in Action

Let's return to our D&D characters.

What features did our data have:

- **Deterministic**:
  - Name
  - Class
  - Race
- **Probabilistic**:
  - *Stochastic*:
    - Strength, Constitution, Dexterity, Intelligence, Wisdom, Charisma
  - *Systematic*:
    - Hitpoints (based on Constitution)
    - Initiative Mod (based on Dexterity)

Our D&D characters are a sample from a DGP!

DGPs in Action

The stochastic components (the 'stats' or attributes):

DGPs in Action

The stochastic components (the ‘stats’ or attributes):

```
roll_4d6 <- function() {
    rolls <- sample(1:6, 4, replace = TRUE)
    sum(sort(rolls, decreasing = TRUE)[1:3])
}
```

DGPs in Action

The stochastic components (the 'stats' or attributes):

```
roll_4d6 <- function() {
    rolls <- sample(1:6, 4, replace = TRUE)
    sum(sort(rolls, decreasing = TRUE)[1:3])
}
```

The systematic components are derived from the above (and a few other bits):

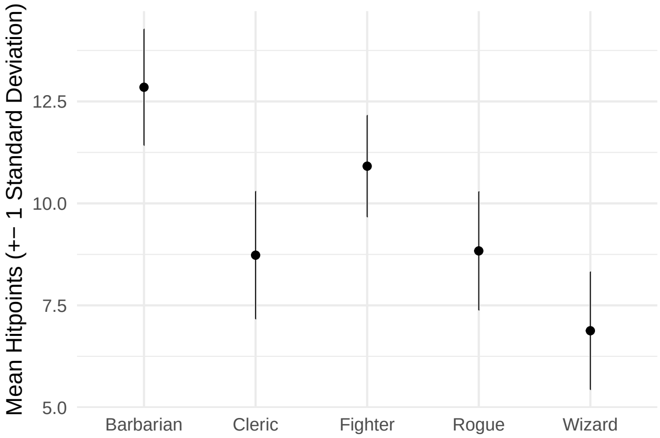
```
# Note: The below is pseudo-code!
# Assigning class (this was actually random!)
class <- sample(c("Fighter","Wizard","Rogue","Cleric","Barbarian"), 1)
# A function to calculate 'modifiers':
modifier <- function(score) floor((score - 10) / 2)
# Calculating the constitution and dexterity modifiers from a stats object:
con_mod <- modifier(stats[["Constitution"]])
dex_mod <- modifier(stats[["Dexterity"]])
# A dictionary of hit dice by class
hit_dice <- c(Fighter = 10, Wizard = 6, Rogue = 8, Cleric = 8, Barbarian = 12)
# Calculating the hitpoints and initiative modifier:
Hitpoints = hit_dice[[class]] + con_mod
InitiativeMod = dex_mod
```

DGPs in Action

Given that we know the DGP, we can model (abstract) things very effectively, e.g.:

```
dnd_characters %>%
  group_by(Class) %>%
  summarise(mean_hitpoints = mean(Hitpoints),
            sd_hitpoints = sd(Hitpoints)) %>%
  ggplot(aes(x = Class, y = mean_hitpoints)) +
  geom_point(size = 4) +
  geom_errorbar(aes(ymin = mean_hitpoints - sd_hitpoints,
                    ymax = mean_hitpoints + sd_hitpoints), width = 0) +
  labs(x = "", y = "Mean Hitpoints (+- 1 Standard Deviation)") +
  theme_minimal(base_size = 24)
```

DGPs in Action



This shows the CEF of hitpoints as a function of class!

Visualising Data

What is Visualisation?

Visualisation: Presenting something visually (wow, insightful).

What is Visualisation?

Visualisation: Presenting something visually (wow, insightful).

This could mean a:

- histogram or density plot of a variable
- two-way (or three-way) scatterplot of points
- map of something
- photo or screenshot of an experimental intervention
- coefficient plot
- trend-line over time
- table (!)

What is Visualisation?

Visualisation: Presenting something visually (wow, insightful).

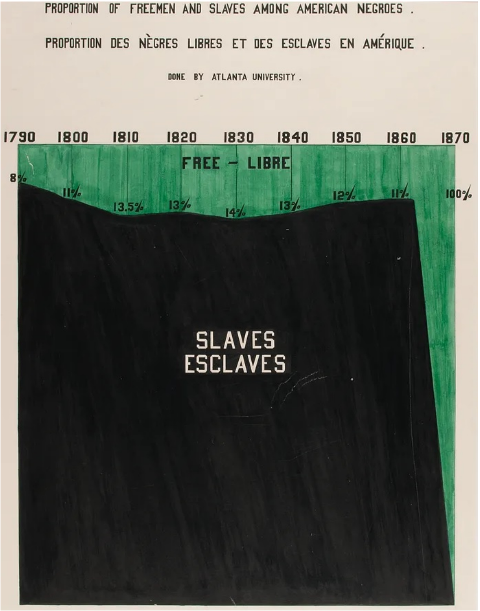
This could mean a:

- ➜ histogram or density plot of a variable
- ➜ two-way (or three-way) scatterplot of points
- ➜ map of something
- ➜ photo or screenshot of an experimental intervention
- ➜ coefficient plot
- ➜ trend-line over time
- ➜ table (!)

Visualisation is important through the life-cycle of a project:

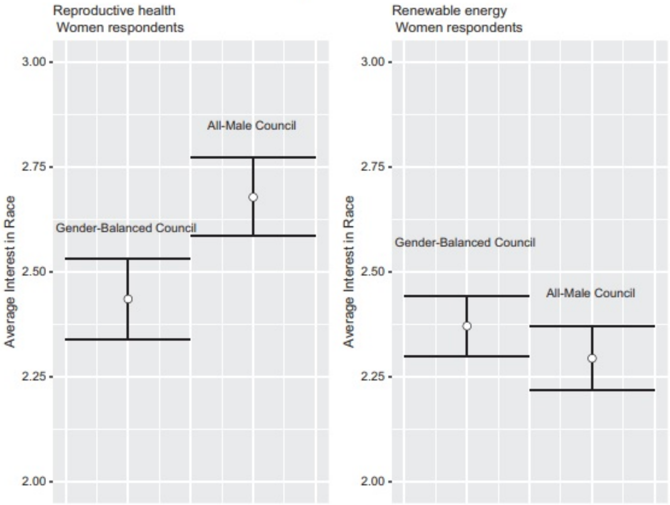
- ➜ **Early**: looking at data to assess quality, properties, face validity
- ➜ **Mid**: monitoring what you are doing along the way
- ➜ **Late**: what you include in presentations/papers

Good Visualisation, Bad Visualisation



Good Visualisation, Bad Visualisation

FIGURE 2. Treatment Effects for Women Respondents on Interest in Hypothetical Race



Note: Error Bars at 95% confidence intervals. See also Model 1 of Table 1.

Good Visualisation, Bad Visualisation

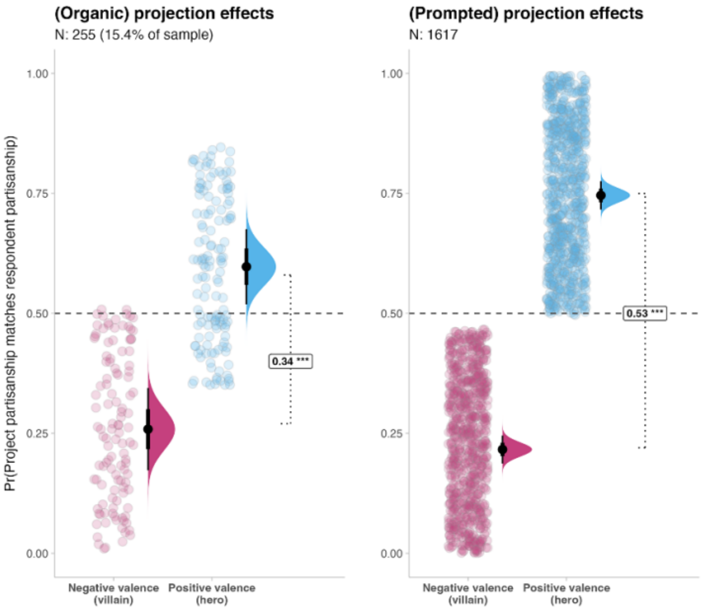
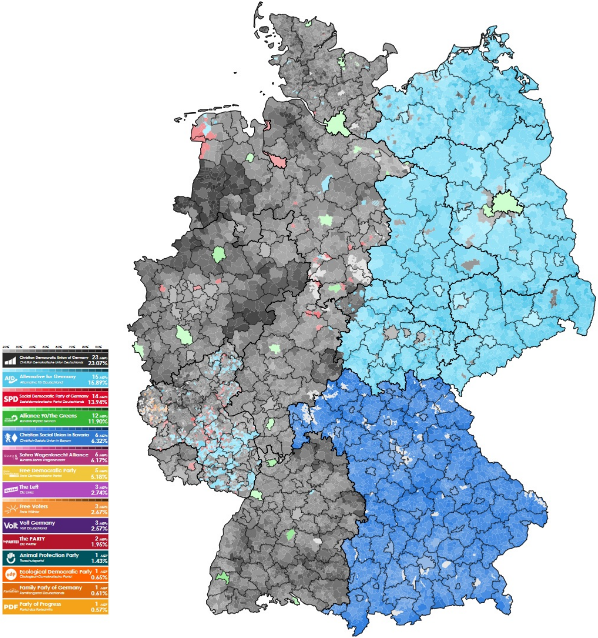
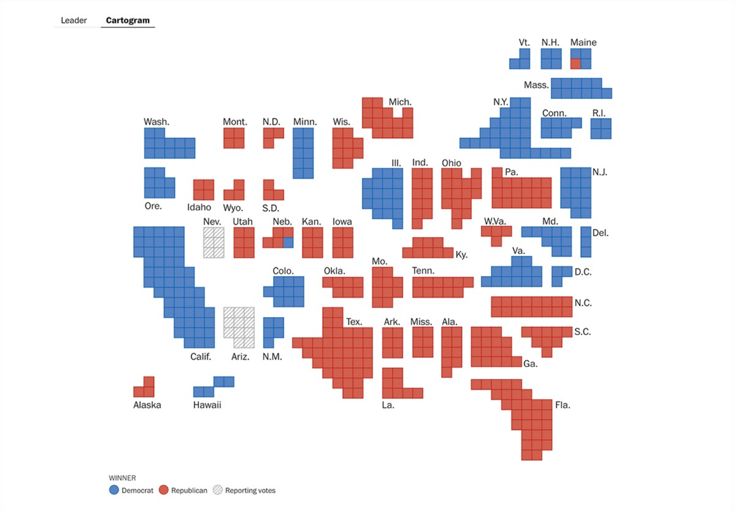


Figure 5. Modeling projecting via false recall (Study 2).

Good Visualisation, Bad Visualisation



Good Visualisation, Bad Visualisation



The Language of Visualisation: Some Anatomy

A visualisation is some visual representation drawn on a plane, with four necessary components.

The Language of Visualisation: Some Anatomy

A visualisation is some visual representation drawn on a plane, with four necessary components.

- **Data**: Representation of the information we care about

The Language of Visualisation: Some Anatomy

A visualisation is some visual representation drawn on a plane, with four necessary components.

- **Data**: Representation of the information we care about
- **Aesthetics**: Symbolic features, e.g.
  - Coordinate [x,y]
  - Colour
  - Shape

The Language of Visualisation: Some Anatomy

A visualisation is some visual representation drawn on a plane, with four necessary components.

- **Data**: Representation of the information we care about
- **Aesthetics**: Symbolic features, e.g.
  - Coordinate [x,y]
  - Colour
  - Shape
- **Mappings**: Correspondence between data and aesthetics

The Language of Visualisation: Some Anatomy

A visualisation is some visual representation drawn on a plane, with four necessary components.

- **Data**: Representation of the information we care about
- **Aesthetics**: Symbolic features, e.g.
  - Coordinate [x,y]
  - Colour
  - Shape
- **Mappings**: Correspondence between data and aesthetics
- **Geometries**: Objects that are drawn on the plane, e.g.
  - Lines
  - Points
  - Bars

The Language of Visualisation: Some Anatomy

A visualisation is some visual representation drawn on a plane, with four necessary components.

- **Data**: Representation of the information we care about
- **Aesthetics**: Symbolic features, e.g.
  - Coordinate [x,y]
  - Colour
  - Shape
- **Mappings**: Correspondence between data and aesthetics
- **Geometries**: Objects that are drawn on the plane, e.g.
  - Lines
  - Points
  - Bars

A visualisation is a **mapping** of **data** to **aesthetics** via visual **geometries** to produce meaning. Formally this can be represented as a "**grammar of graphics**" (Wilkinson, 2005).

The Language of Visualisation: Some Anatomy

Mappings of data to aesthetics and geometries give visualisations **meaning** (or 'content'). Note what, strictly speaking, doesn't matter...

The Language of Visualisation: Some Anatomy

Mappings of data to aesthetics and geometries give visualisations **meaning** (or 'content'). Note what, strictly speaking, doesn't matter. . .

- **Explanations**: A language-based statement of what the plot represents
  - A note at the bottom of the plot
  - Titles and subtitles
  - Axis labels
  - Legends

The Language of Visualisation: Some Anatomy

Mappings of data to aesthetics and geometries give visualisations **meaning** (or ‘content’). Note what, strictly speaking, doesn’t matter. . .

- **Explanations**: A language-based statement of what the plot represents
  - A note at the bottom of the plot
  - Titles and subtitles
  - Axis labels
  - Legends

- **Design**: Things you do to make the plot look good
  - Typeface
  - Font
  - Themes

The Language of Visualisation: Some Anatomy

Mappings of data to aesthetics and geometries give visualisations **meaning** (or 'content'). Note what, strictly speaking, doesn't matter...

- **Explanations**: A language-based statement of what the plot represents
  - A note at the bottom of the plot
  - Titles and subtitles
  - Axis labels
  - Legends

- **Design**: Things you do to make the plot look good
  - Typeface
  - Font
  - Themes

Of course this stuff **does matter**, but not in the way we imagine. Let's return to it later.

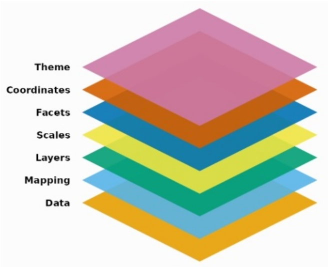
Grammar of Graphics ⟷ {ggplot2}

In {ggplot2}, three key pieces:

1. **Data**: The information
2. **Mapping**: Correspondence between data and aesthetics
3. **Layers**: Geometry and transformation

Then we get:

- ➜ Scales: Paired to an aesthetic
- ➜ Facets: Subsets of data
- ➜ Coordinates: The x-y system of the canvas
- ➜ Theme: Design choices



Source: Introduction to ggplot2

The Good, The Bad, and the Ugly

Misconception: Good data visusalisation = good visual taste.

The Good, The Bad, and the Ugly

Misconception: Good data visusalisation = good visual taste.

No! Good (and bad) data visualisation is **not** primarily about how nice (ugly) your plot looks.

It is about **communication** with the **reader/viewer**.

Making your plot look nice is the **last step** in data/model visualisation, and only relevant once your visualisation meets its "communication goal."

The Good, The Bad, and the Ugly

Misconception: Good data visusalisation = good visual taste.

No! Good (and bad) data visualisation is **not** primarily about how nice (ugly) your plot looks.

It is about **communication** with the **reader/viewer**.

Making your plot look nice is the **last step** in data/model visualisation, and only relevant once your visualisation meets its "communication goal."

If your visualisation is misleading, confusing, or fails to communicate, then from a scientific perspective it **does not matter** how pretty it is.

Design and explanation only matter inasmuch as they contribute to the communication goal.

Goals of Visualisation

What are useful **communication goals**?

- ➜ Delivering a (set of) fact(s)
- ➜ Telling a story
- ➜ Imparting meaning

Goals of Visualisation

What are useful **communication goals**?

- Delivering a (set of) fact(s)
- Telling a story
- Imparting meaning

So, what makes for a good (bad) visualisation:

- **Thoughtful**: There is a clear communication goal, and the visualisation is built with that goal in mind.
- **Honest**: The reader is not misled about the data or the meaning.

Good vs. Bad: General Principles

Emphasis: **thoughtful** and **honest** visualisation.

Thoughtful:

- Communication goal you can articulate in words
- One aesthetic mapping per variable (some exceptions apply)
- No unmapped aesthetics
- Visually interpretable mappings
- Balance information and meaning – raw information is rarely meaningful
- Avoid clutter ("chartjunk" in Tufte's (1983) words)

Good vs. Bad: General Principles

Emphasis: **thoughtful** and **honest** visualisation.

Thoughtful:

- Communication goal you can articulate in words
- One aesthetic mapping per variable (some exceptions apply)
- No unmapped aesthetics
- Visually interpretable mappings
- Balance information and meaning – raw information is rarely meaningful
- Avoid clutter ("chartjunk" in Tufte's (1983) words)

Honest: - Avoid distortions of the data or statistics - Avoid axis compression where possible/necessary - Avoid "made up" data

Good vs. Bad: The Lie Factor

Tufte (1983) proposes the Lie Factor:

$$\text{Lie Factor} = \frac{\text{Size of Comparison in Visual}}{\text{Size of Comparison in Data}}$$

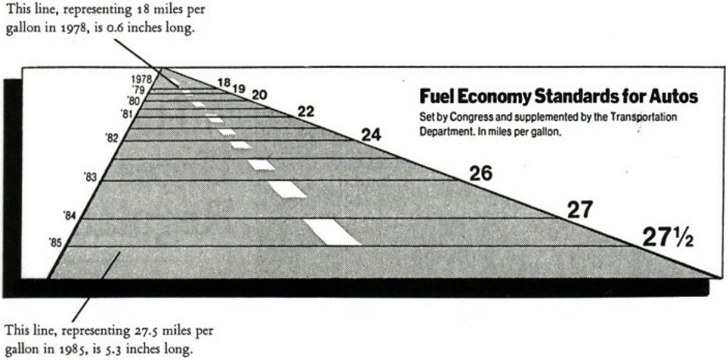
Where:

$$\text{Size of Comparison} = \frac{\text{Second Value - First Value}}{\text{First Value}}$$

Essentially, any Lie Factor other than 1 implies a visual distortion of the data.

Good vs. Bad: The Lie Factor

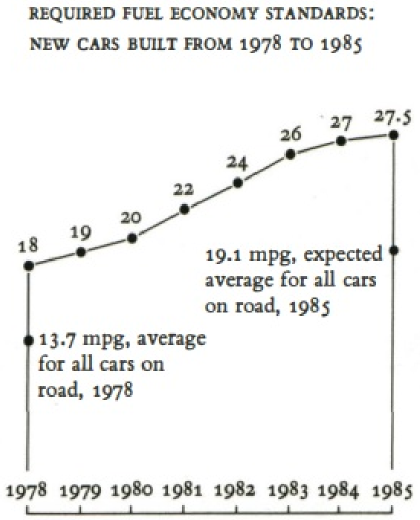
Classic case from the New York Times:



Tufte calculates the Lie Factor of this as $14.8 = \frac{783\%}{53\%}$

Good vs. Bad: The Lie Factor

Undoing the lie, mostly (Tufte, 1983):



From DGP To Visualisation

Finishing Our Adventure

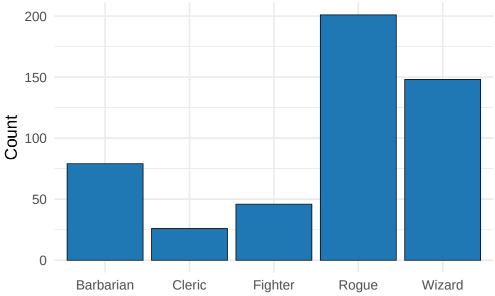
Let's wrap up our D&D adventure by using our visualisation skills to answer some data science questions about our D&D characters.

Finishing Our Adventure

Let's wrap up our D&D adventure by using our visualisation skills to answer some data science questions about our D&D characters.

Question: Which classes are the most popular?

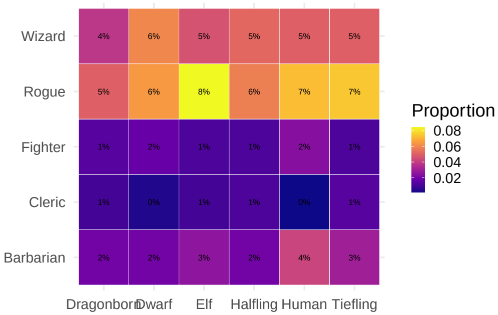
```
dnd_characters %>%
  ggplot(aes(x = Class)) +
  geom_bar(fill="#1f77b4", color = "black") +
  labs(x = "", y = "Count", title = "") +
  theme_minimal(base_size = 24)
```



Finishing Our Adventure

Question: Do character classes and races cluster?

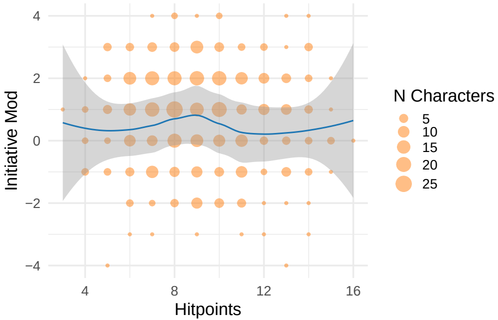
```r
dnd_characters %>%
  count(Class, Race) %>%
  mutate(share = n / sum(n)) %>%
  ggplot(aes(x = Race, y = Class, fill = share)) +
  geom_tile(color = "white") +
  geom_text(aes(label = scales::percent(share, accuracy = 1)), color = "black") +
  scale_fill_viridis_c(option = "C") +
  theme_minimal(base_size = 24) +
  labs(title = "",
       x = "",
       y = "",
       fill = "Proportion")
```



Finishing Our Adventure

Question: Is there a trade-off between initiative and hitpoints?

```
dnd_characters %>%
  count(Hitpoints, InitiativeMod) %>%
  ggplot(aes(x = Hitpoints, y = InitiativeMod)) +
  geom_point(aes(size = n), alpha = 0.5, color = "#ff7f0e") +
  geom_smooth(method = "loess", color = "#1f77b4", se = TRUE) +
  scale_size_continuous(name = "N Characters", range = c(2, 10)) +
  labs(
    x = "Hitpoints",
    y = "Initiative Mod",
    title = ""
  ) +
  theme_minimal(base_size = 24)
```



Capstone: Data Science Practice

Programming: Legibility and Interpretability

We're done with the fundamentals now. Before we wade into deeper waters, let's talk a bit about the practice of data science.

Going forward, you are going to write a lot of your own code. You want to focus on:

- **Legibility**: Is the code easy to read and understand?
- **Interpretability**: Can someone else (or you in the future) understand what the code is doing?

Some (good) ideas:

- Comment your code.
- Use meaningful variable/object names
- Use notebooks ('literate programming')

Programming: Modularity and Reusability

We also want to pay attention to:

- **Modularity**: Can we break the code into smaller, reusable functions or modules?
- **Reusability**: Can we design code such that we can reuse it in different contexts or projects?

There are trade-offs here though, and you should pay attention to them.

Programming: Modularity and Reusability

We also want to pay attention to:

- **Modularity**: Can we break the code into smaller, reusable functions or modules?
- **Reusability**: Can we design code such that we can reuse it in different contexts or projects?

There are trade-offs here though, and you should pay attention to them.

A rule you want to follow: Write **DRY** code, not **WET** code.

- **DRY**: **D**on't **R**epeat **Y**ourself.
- **WET**: **W**rite **E**verything **T**wice (or more).

Honesty, Transparency, and Reproducibility

We have not yet talked about the 'ethics' of data science.

- ➜ **Honesty**: Be honest about what you have done, and what you have not done
- ➜ **Transparency**: Be transparent about the assumptions you are making, and the limitations of your models
- ➜ **Responsibility**: You are responsible for the data you use, the code you write, and the results you product

A good (but not sufficient!) way to practice 'ethical' data science:

- ➜ **Reproducibility**: Make sure that your code can be run by others, and that they can reproduce your results