Week 9: Other Data Types

LSE MY472: Data for Data Scientists
https://lse-my472.github.io/

Autumn Term 2024

Ryan Hübert

Slides last updated: 27 November 2024

Introduction

The standard social science dataset is tabular data

But lots of interesting "data" in the world isn't tabular

➜ For example: audio, photos, maps

All these types of data present two challenges when representing in quantitative format that can be used for analysis

1. **Dimensionality**: how to represent more than 2 dimensions in 2-dimensional spaces
2. **Discretisation**: converting **analog** formats to computer-readable **digital** formats ("digitisation")

Introduction

In this lecture, we'll cover: audio data, digital graphics/photos, maps

We'll talk about how to:

- ➜ conceptualise these data types in quantitative format,
- ➜ ingest these data types into R
- ➜ do some simple manipulations/visualisations

There is lots of *analysis* you can do with these kinds of data

- ➜ Recall: this class isn't about that!
- ➜ Our goal here is to orient you to the basics of these data types
- ➜ Some promising applications use machine learning/AI to analyse data at scale (e.g. Francisco Cantú's 2019 article)

Warning: some of following R packages aren't well developed

Audio data

The basics of sound

Sound is produced by waves, creating **pitch** and **loudness**

- **Frequency** (**wavelength**) → pitch
- **Amplitude** → loudness

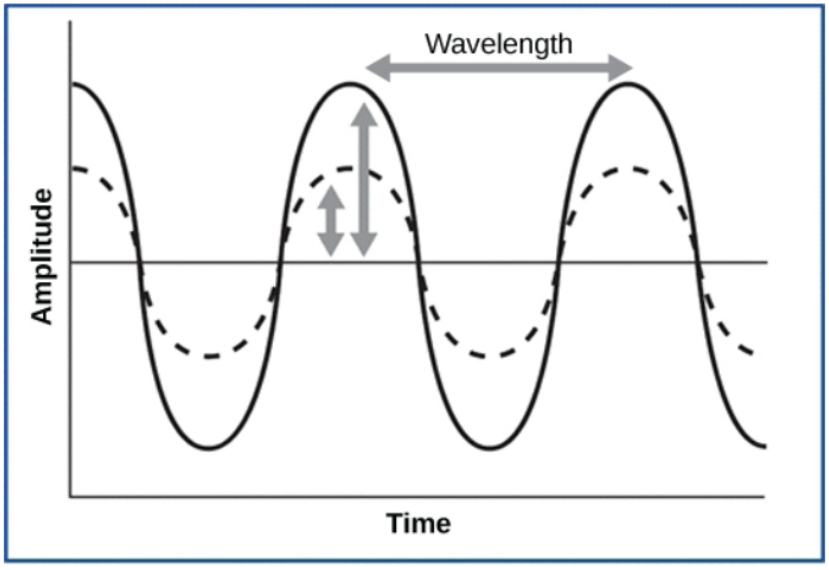


Figure 1. For sound waves, wavelength corresponds to pitch. Amplitude of the wave corresponds to volume. The sound wave shown with a dashed line is softer in volume than the sound wave shown with a solid line. (credit: NIH)

Source: https://books.psychstat.org/rdata/audio-data.html

The basics of sound

The link between sound waves and human hearing is complicated

Focusing on pitch and loudness:

- Sound pitch can be measured using frequency in Hertz (Hz)
  - If a sound wave completes one cycle in one second = 1 Hz
  - Humans can hear sounds from 20 Hz to 20,000 Hz
- Loudness can be measured in decibels (dB)
  - A conversation is around ~60 dB
  - Hearing loss can occur with prolonged exposure above ~80 dB

Sound digitisation

Sound is **time-continuous** but (computer-readable) data is represented discretely

➜ Digitising sound involves discretising it

We have to **sample** (not in statistical sense)

➜ Every X period of time, measure amplitude & "round" it

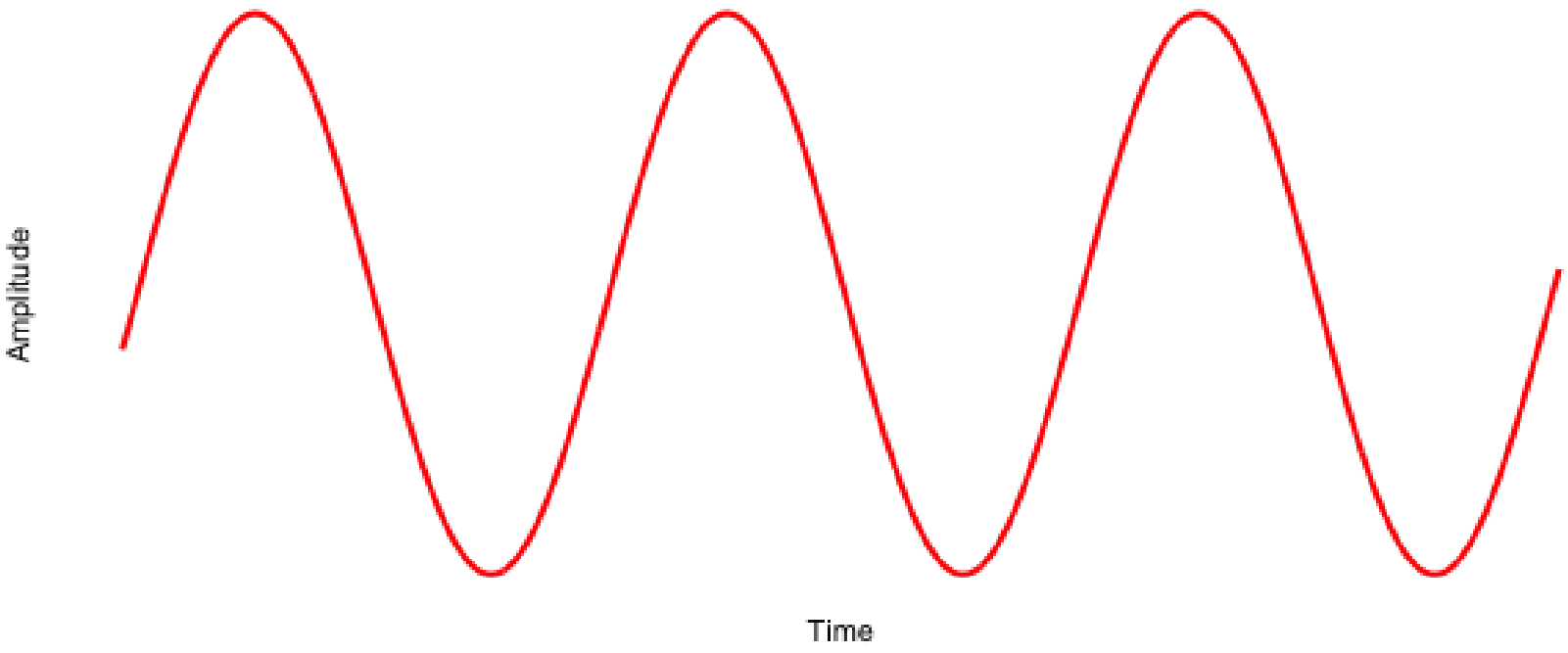
Sampling frequency: measure of how many samples are taken per second, measured in Hz – commonly 44,100 Hz (44.1 kHz)

Quantising: how amplitude readings are "rounded" when stored

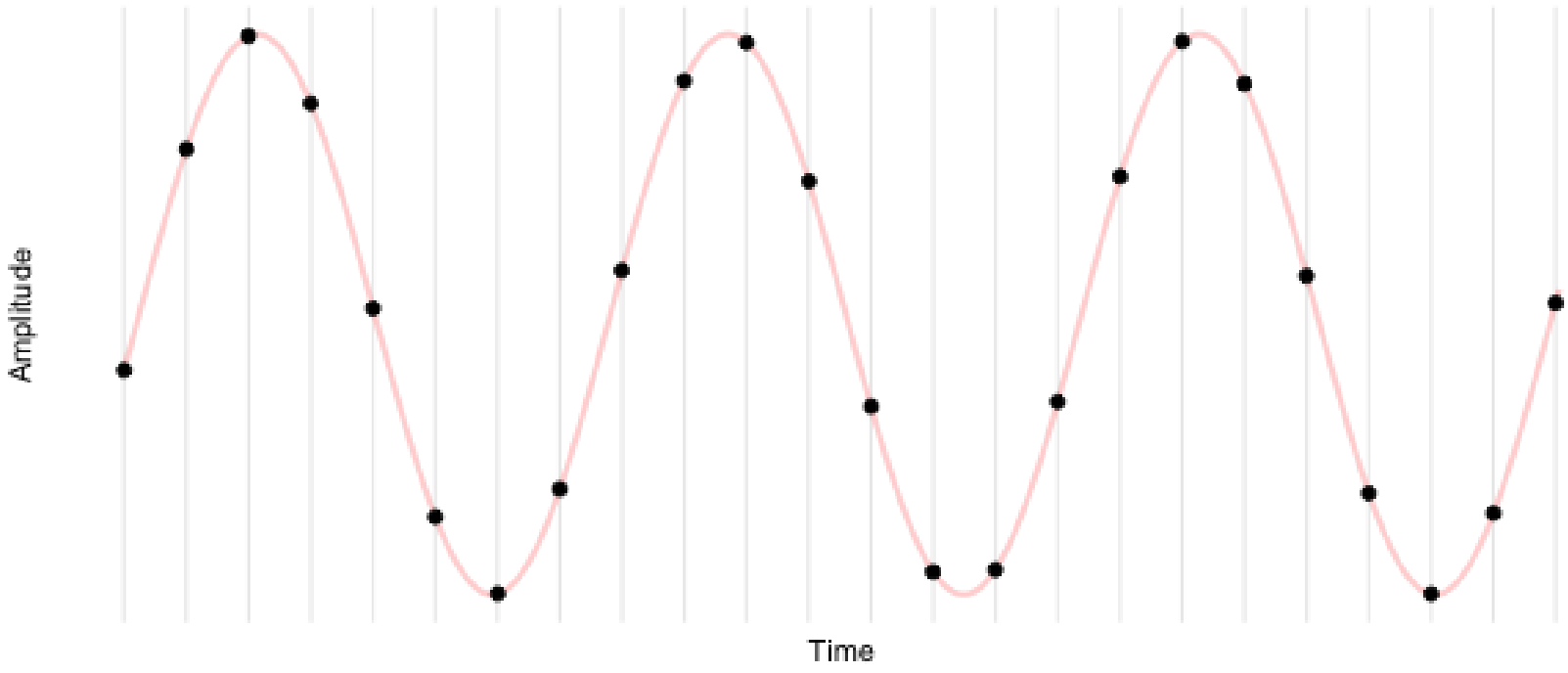
➜ Digitised sound is quantised in bits, so 4-bit digitised sound measures amplitude in $2^4 = 16$ intervals

Sampling + rounding causes information loss and lowers quality

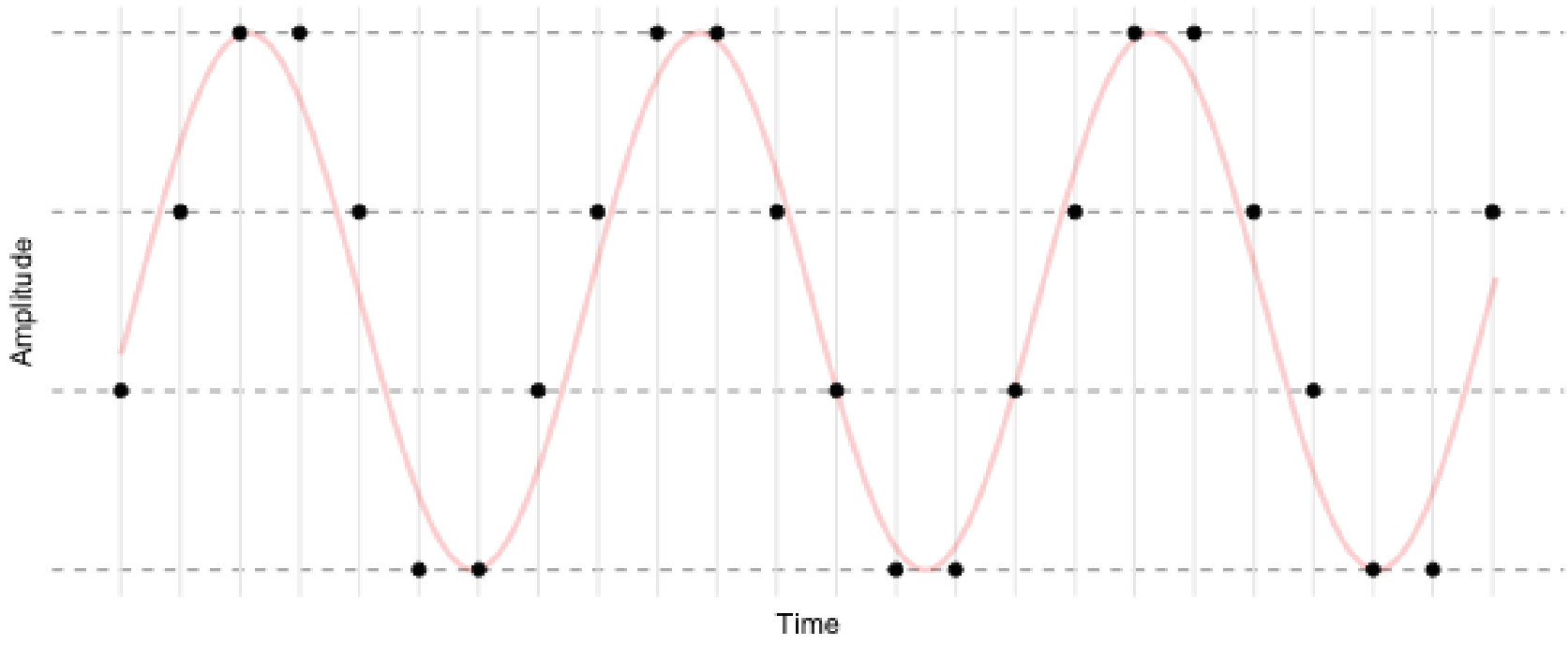
Digital audio sampling



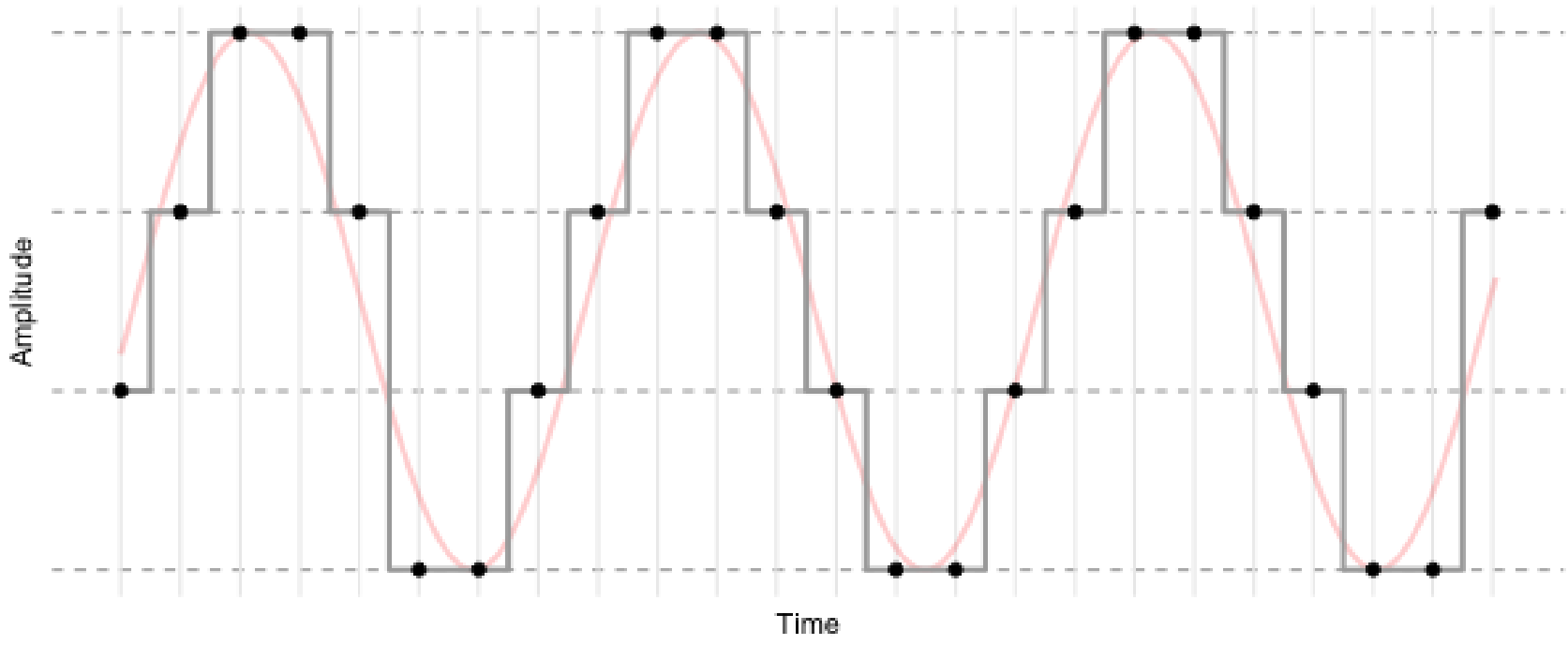
Digital audio sampling



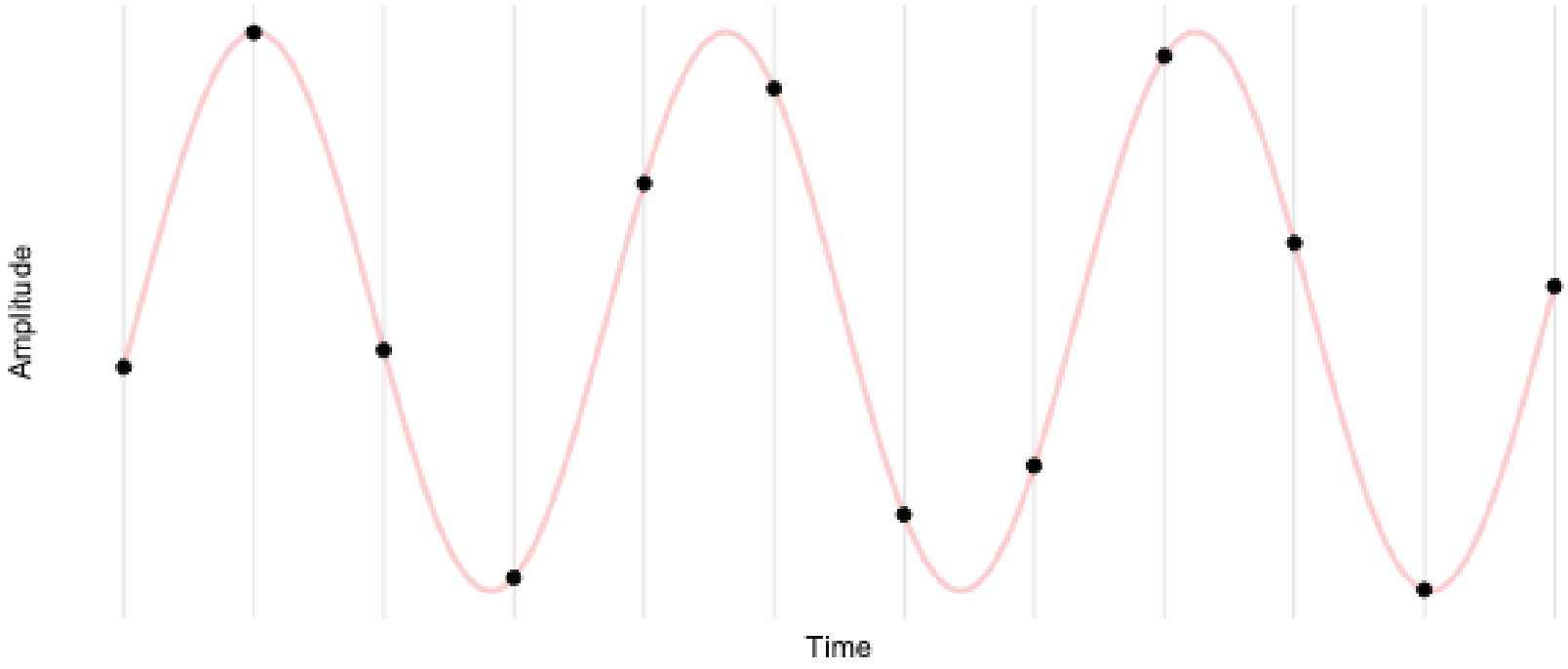
Digital audio sampling



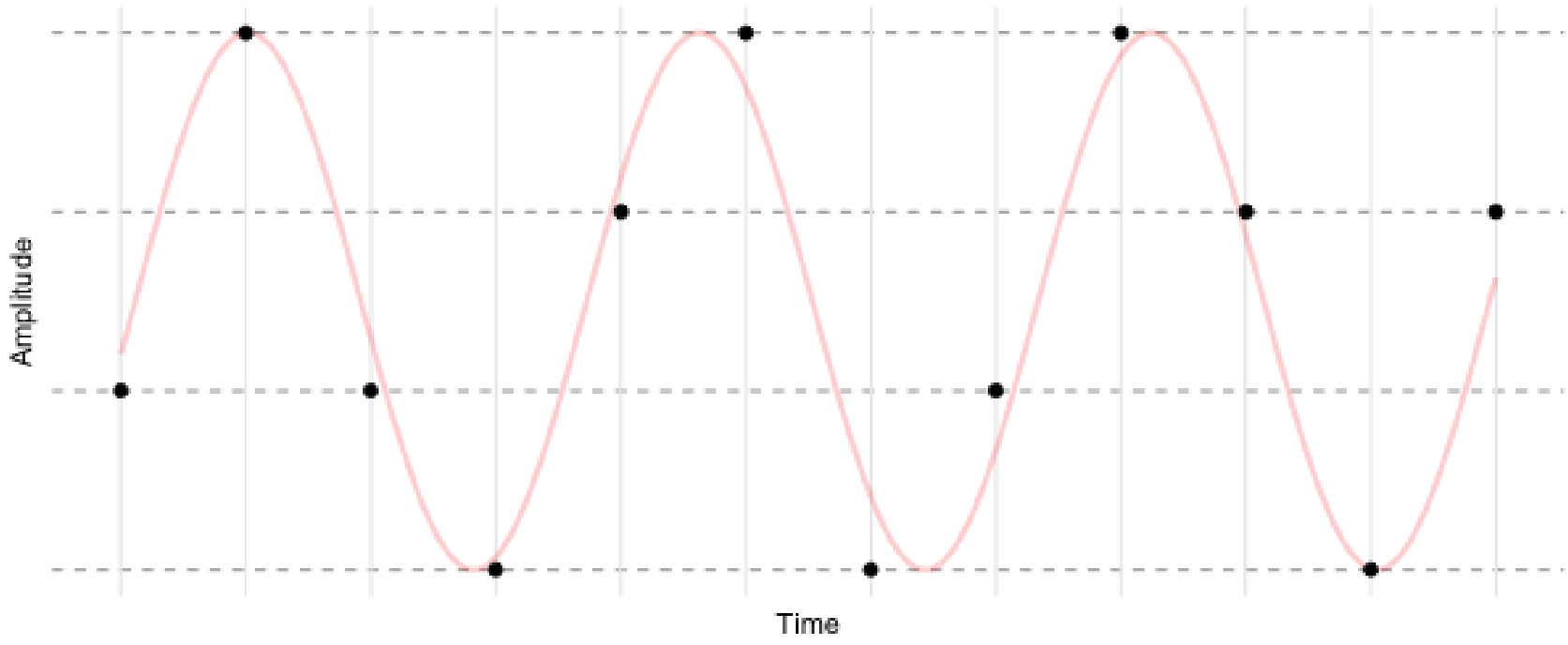
Digital audio sampling



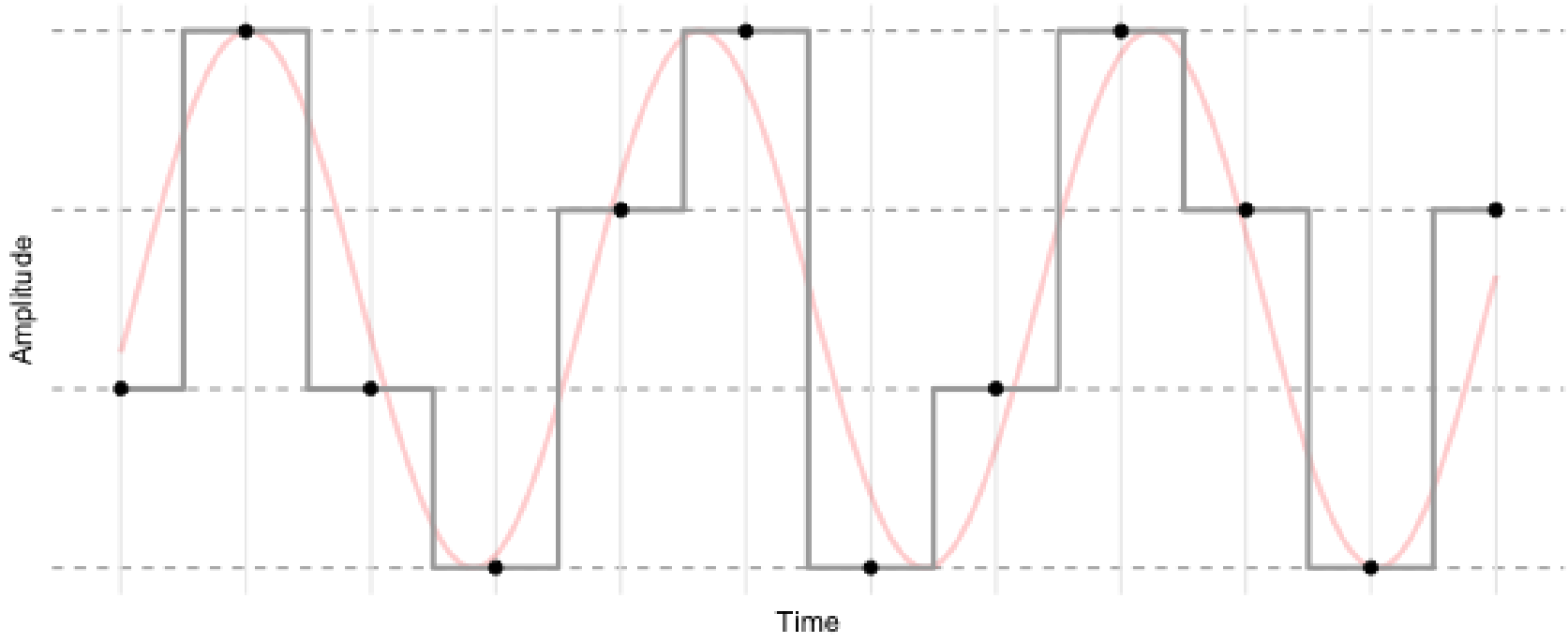
Digital audio sampling



Digital audio sampling



Digital audio sampling



Audio data in R

For audio data in R: `tuneR` and `seewave`

- Audio data is a sequence of recorded amplitude readings plus sampling information (sampling frequency and bits)
- In `tuneR` data is loaded as a `Wave` object
  - Access sampling frequency via `@samp.rate` and bits via `@bits`
  - Access amplitude readings via `@left` and `@right`
  - If mono, there will only be readings in `@left`

Side note: **mono** versus **stereo** refers to how many "channels" are in digitised audio—mono is 1 channel, stereo is 2 channels

- Many stereo speakers/headphones will play mono audio through both speakers

Visualising audio data

Since sound is generated by waves, the "data" of sound allows you to (approximately) reconstruct the waves

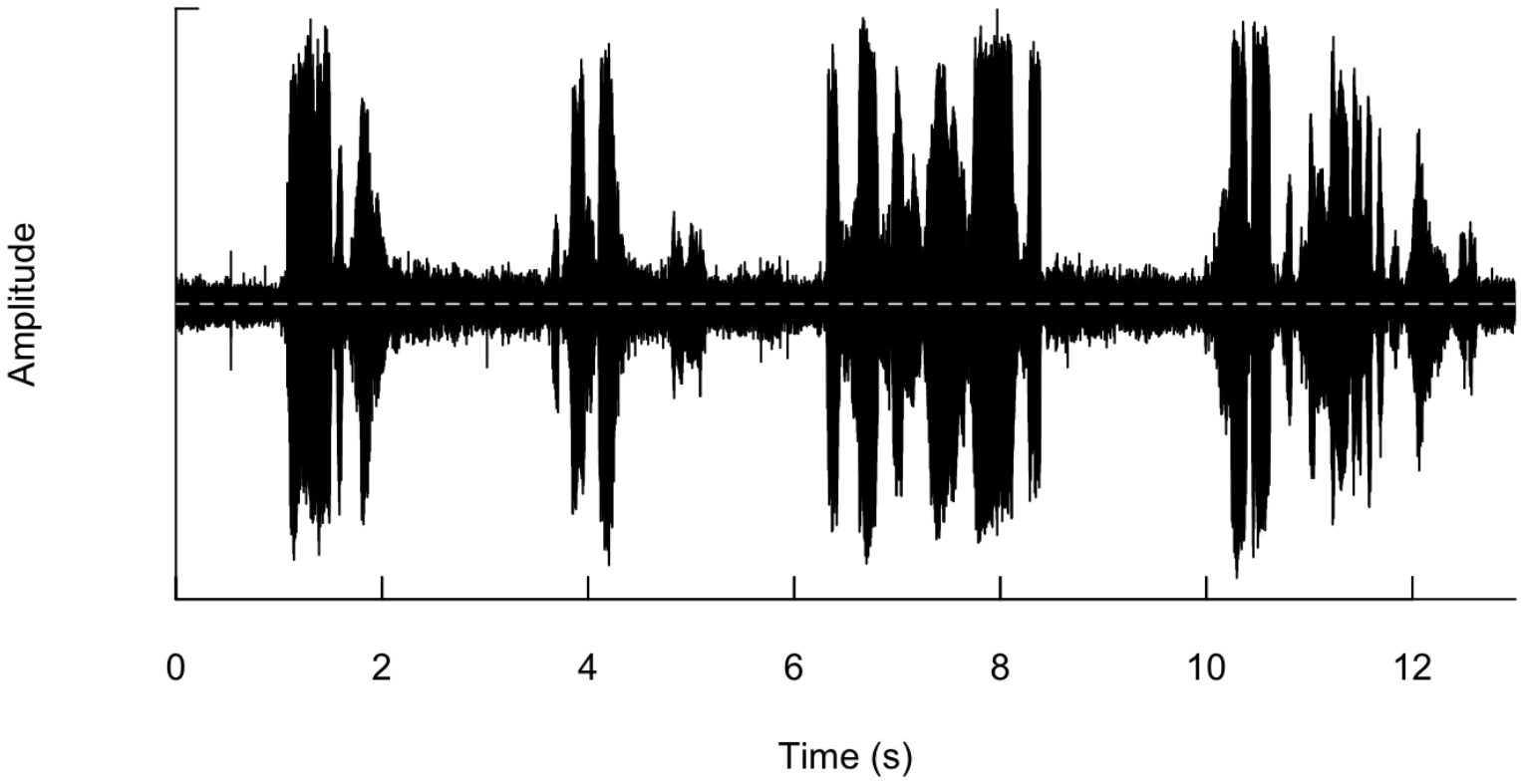
But we have a dimensionality problem:

- ➜ time
- ➜ amplitude
- ➜ frequency

Three basic types of visualisations are useful: **oscillogram**, **periodogram** and a **spectrogram**

Visualising audio data

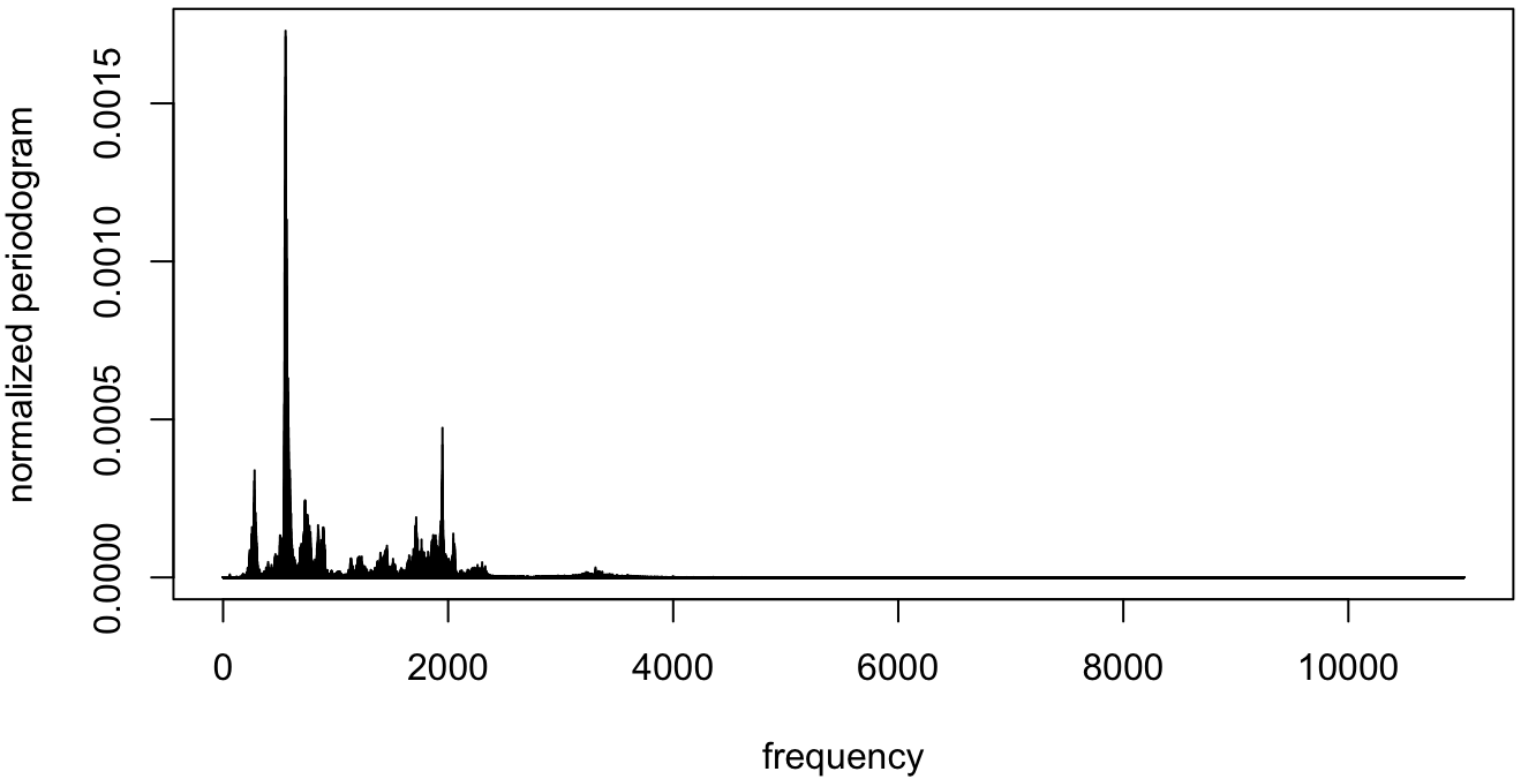
1. An **oscillogram** is simply a plot of a digitised sound wave
   - ➜ But: a little difficult to "see" the frequency



Visualising audio data

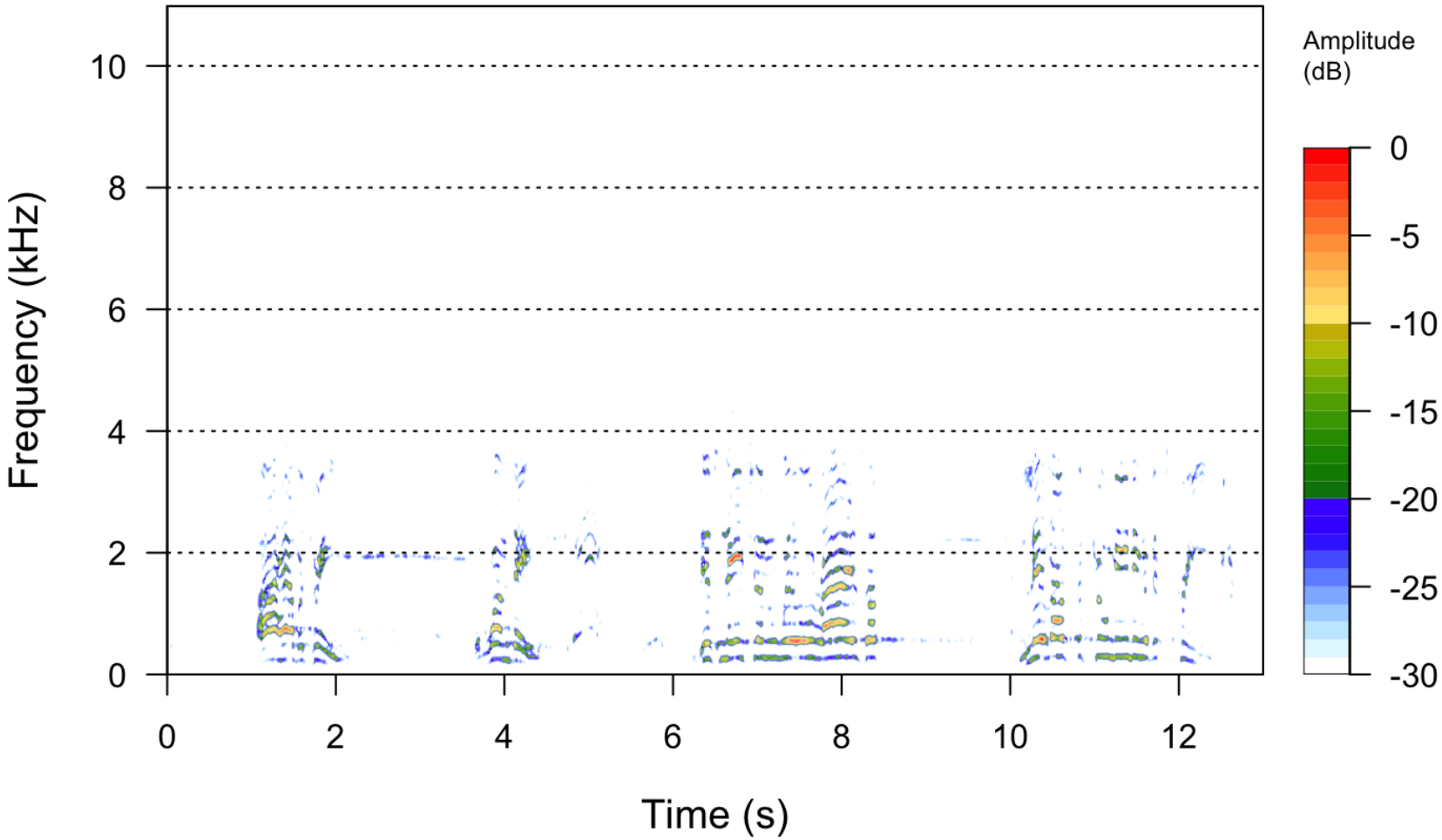
2. A **periodogram** shows the "dominant" frequencies in a recording, i.e. frequencies that contribute most to amplitude

   ➜ But: it doesn't show time



Visualising audio data

3. A **spectrogram** is a "heat map" that depicts variation in amplitude *and* frequency over time

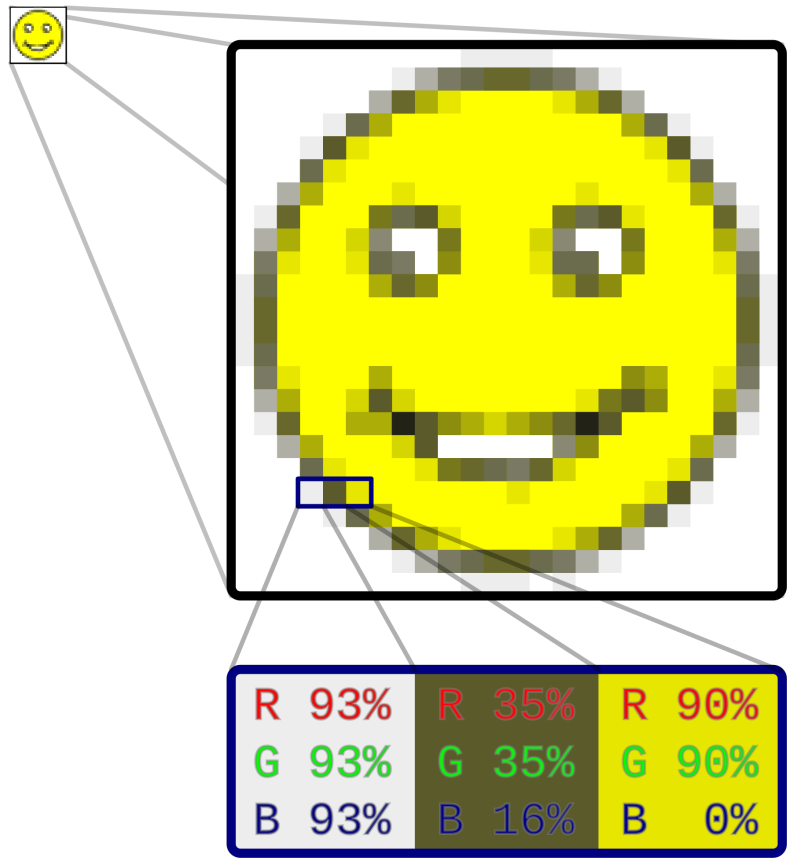


Basics of digital graphics

Raster image data

Raster (or bitmap) graphics store images as a grid

➜ Many digital images: `.gif`, `.jpg`/`.jpeg`, and `.png`



Source: https://en.wikipedia.org/wiki/Raster_graphics

Raster image data

Raster graphics have three important components:

1. A grid (or **tessellation**) of equal size cells
2. A single value in each cell of the grid
3. A **lookup table** that maps cell values to colors (similar to character set from week 4)

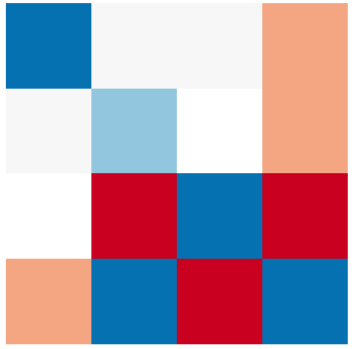
A. Cell IDs

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

B. Cell values

92	55	48	21
58	70	NA	37
NA	12	94	11
36	83	4	88

C. Colored values

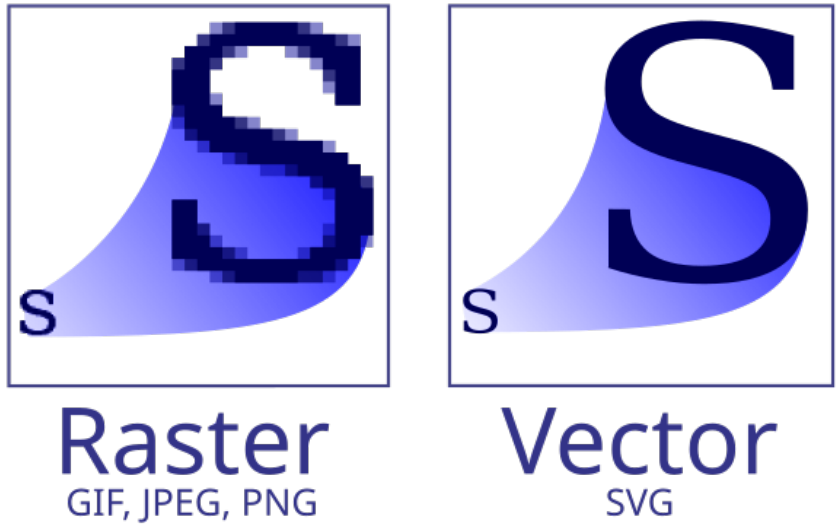


Source: https://r.geocompx.org/spatial-class

Vector image data

Vector graphics store images as geometric objects, such as points, line segments, polygons, etc.

➜ Common file types: `.svg`, `.pdf`, and `.eps`



Source: https://en.wikipedia.org/wiki/Vector_graphics

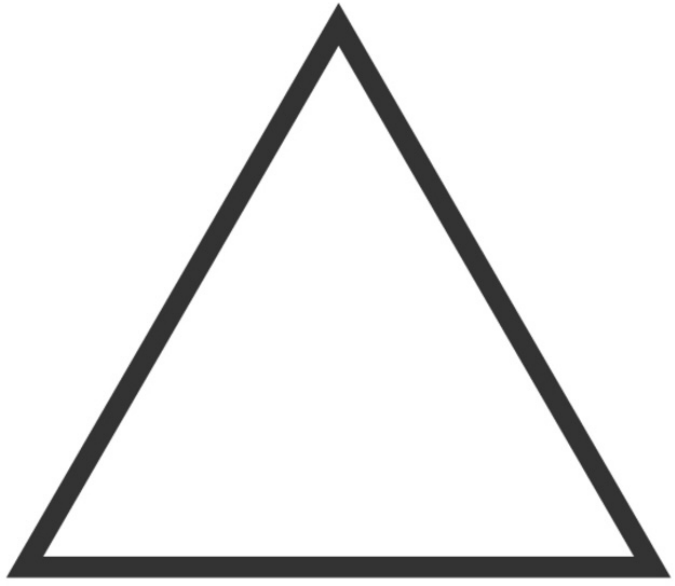
Vector image data

Vector graphics are defined by **geometries**: points, line segments, polygons, curves, etc.

Roughly speaking: vector graphics "trace" images with geometries

Polygons are a common way to make shapes, defined by:

- ➜ **Edges**: line segments
- ➜ **Nodes**/**vertices**: where line segments meet



Use cases

Vector graphics: images with geometric shapes and no continuous tones/colors, and/or when rescaling is important

- ➜ Maps of borders and other geographical shapes
- ➜ Cartoons, clip art, logos, etc.
- ➜ Typography (e.g., PDFs)

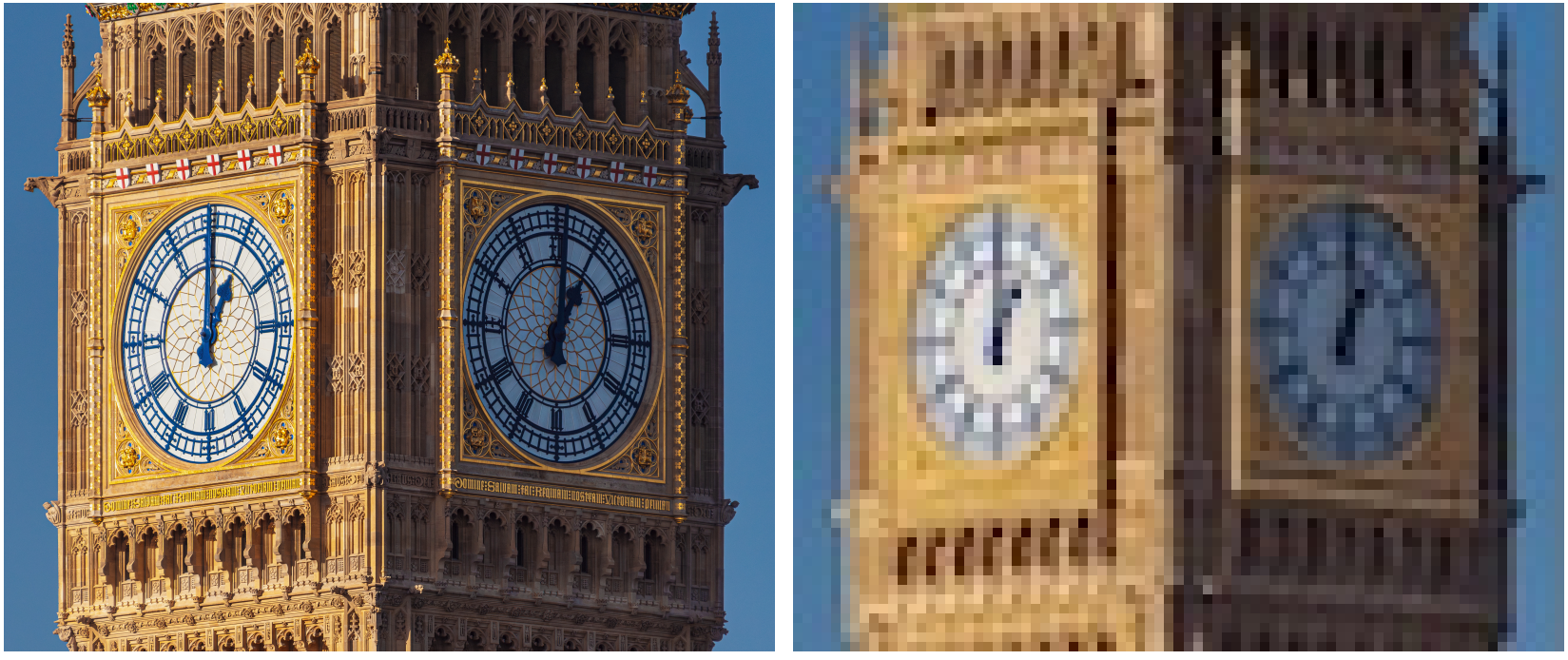
Raster graphics: images with continuous tones/colors, and/or when efficiency/size is important (e.g., websites)

- ➜ Maps with aerial and sensor imaging
- ➜ Topographic maps
- ➜ "Real-life" photos and videos

Notions of image quality

Low quality raster graphics are **pixelated**

➜ "too few" cells in the grid to represent image well

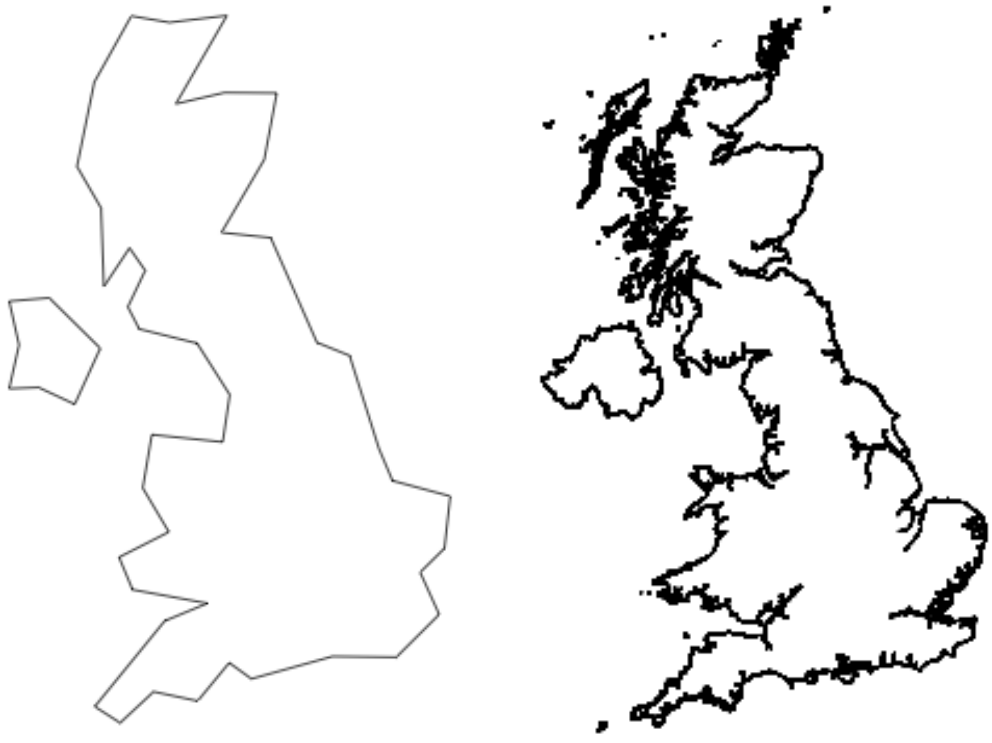


Source: https://en.wikipedia.org/wiki/Big_Ben

Notions of image quality

Low quality vector graphics have **insufficient point density** or **low node resolution**

➜ e.g., “too few” nodes/vertices in a polygon



Digital photos

Photo data in R

Digital photos are typically stored as raster data since they have continuous tones

Most digital photos use the **RGB colour space**

➜ Each raster cell's display colour is determined by a mixture of the three primary colours: red, green and blue

The most common file type for digital photos is `.jpeg`/`.jpg`, although there are others

We won't delve into different file types or how colours are rendered on monitors, etc.

Photo data in R

There are multiple ways to import `.jpg` photo data into R, but we will use `jpeg` and `exifr`

- `exifr`: extract a photo's metadata
- `jpeg`: import a `.jpg` photo and work with its raster data

Reading a `.jpg` into R with `jpeg` yields an **array** object

- It is a collection of three matrices
- Each matrix is a raster grid corresponding to one of the three **colour channels** (RBG)
- Each cell in each matrix is a value from 0 to 1—the proportions of red, green and blue, respectively

When image data is "plotted", each pixel's colour is determined by mixing red, blue and green in the specified proportions

Spatial data

What is spatial data?

Spatial data is data relating to physical space

Physical space is inherently three dimensional, but our data representations are two dimensional

For maps specifically:

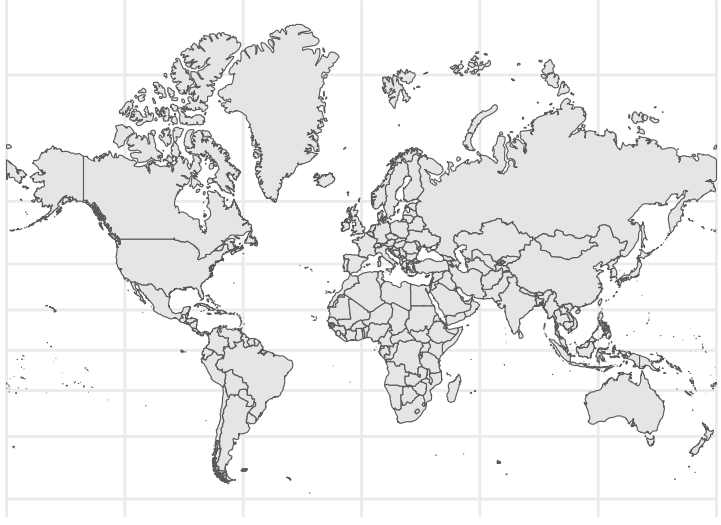
- ➜ the Earth is a 3D sphere (I hope you agree)
- ➜ but maps are depicted on 2D planes

So, mapping requires:

- ➜ a (Cartesian) plane (i.e., x and y axis) and
- ➜ a **projection** that "translates" 3D space to 2D space

Map projections

Mercator Projection
(without Antarctica)



Robinson Projection
(without Antarctica)

Map projection

Coordinate reference system (CRS)

- A way to link points in a 2D geometry with real places
- Set an origin, locations are defined as coordinates (e.g., `x` and `y`) giving distances from origin
- CRS is based on the chosen projection

You need to *choose* a projection that fits your use case

There are tradeoffs, since projections usually distort cardinal directions (north/south/east/west), distances and/or areas

Some commonly used CRS:

- WGS 84: a Mercator projection used in most web applications (e.g, Google Maps)
- British National Grid (BNG), used in the UK

Two types of spatial data: vector and raster

Spatial data in R

For vector data in R: `sf`

- Defines spatial data with **simple features** that represent geometric data types + projection
- Spatial data is loaded into an "`sf` data frame" with:
  - rows: geographic entities (countries, regions, etc)
  - columns: relevant "metadata" about each geographic entity plus one column with geometries
  - the geometry column contains vector data for plotting
- The coordinates used for plotting are "scaled" by the projection

Many ways to store vector data, but commonly used one is "shapefiles" (`.shp`), which is what we will use

Spatial data in R

For raster data in R: `terra`

- Defines spatial data using a raster grid + projection
- Data is loaded as an `SpatRaster` object that is its own class:
  - dimensions: how many cells in the grid (the "resolution")
  - matrix: stores values of cells in raster grid
- The values in the raster grid are "scaled" by the projection

Raster spatial data is stored as raster graphics, usually in `.tif` file format

Code

- `01-audio-visual.Rmd`
- `02-spatial.Rmd`