

Digital humanities

3. Práce s digitálními daty

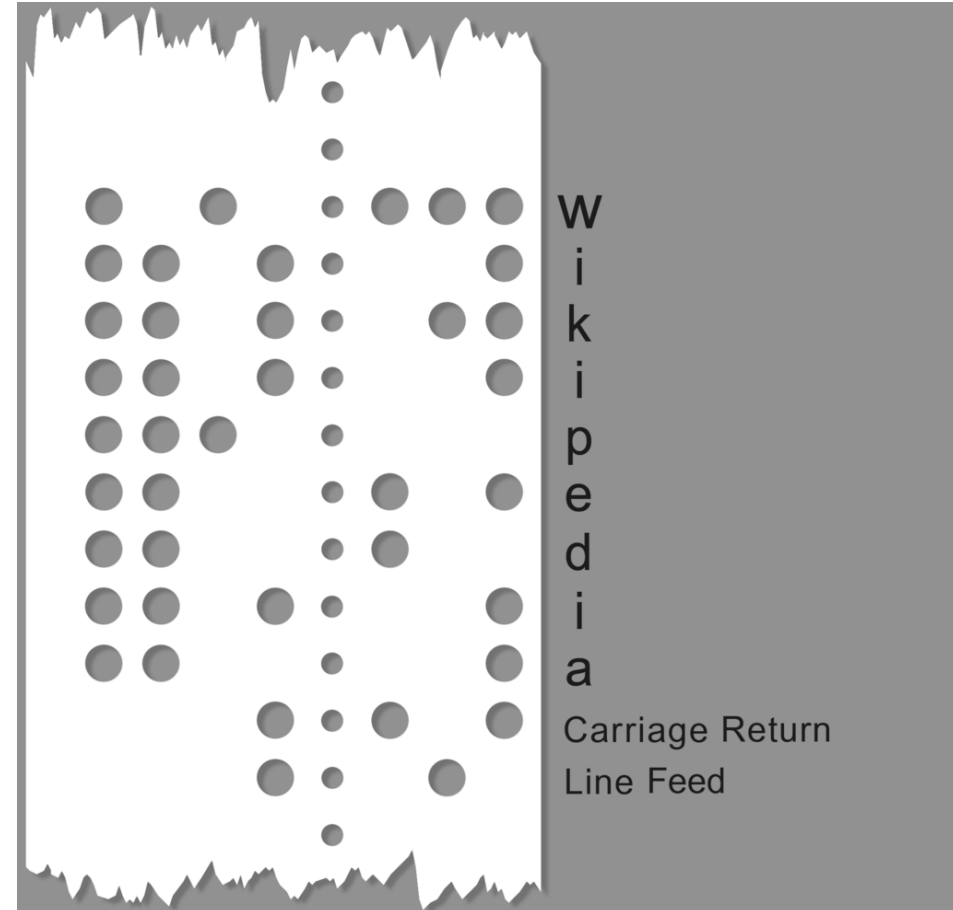
Jindřich Marek

Digitální data: datové typy

- čísla
 - celá čísla
 - desetinná čísla
- texty
- logická data
 - ano/ne
- datum a čas
- tabulární data
- multimédia
 - obrazy (bitové mapy, vektory)
 - video
 - audio
- prostorová data
 - mapy
- síťová data
- ...

Text: kódování znaků

- reprezentace znaků v digitálním prostředí
 - jedničky a nuly
 - ASCII: W = 1010111
 - v desítkové soustavě 87
 - v šestnáctkové soustavě 57







1
:

2
(

3
ß

4
ñ

5
ch

6
ck

7
g

8
s

q

w

e

r

t

z

u

i

Fest-
steller

a

ſ

d

f

g

h

j

k

Um-
schalter

q

r

c

v

b

n

m

?,
;



Ein unentbehrlicher Reise-
begleiter ist die kleine

Erika

Aktiengesellschaft vorm.
Seidel & Naumann, Dresden

ASCII (1963)

- *American Standard Code for Information Interchange*
- 7 bitů na 1 byte
- 128 znaků (pozice 0 až 127)
 - z toho 95 tisknutelných znaků
 - také kontrolní znaky (telekomunikace)

ASCII (1977/1986)																
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1x	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2x	SP	!	"	#	\$	%	&	'	()	*	+	,	-	.	/
3x	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4x	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5x	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_
6x	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7x	p	q	r	s	t	u	v	w	x	y	z	{		}	~	DEL

Legend:

- Changed or added in 1963 version
- Changed in both 1963 version and 1965 draft

ISO/IEC 8859 (1987)

- 8 bitů na 1 byte
- 15 částí (tabulek) pro různé jazykové oblasti
- až 256 znaků (pozice 0 až 255)
- původně ECMA-94, 4 tabulky (1985)
- později podobné Windows code pages

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x																
1x																
2x	SP	!	"	#	\$	%	&	'	()	*	+	,	-	.	/
3x	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4x	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5x	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_
6x	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7x	p	q	r	s	t	u	v	w	x	y	z	{		}	~	
8x																
9x																
Ax	NBSP	À	Á	Â	Ã	Ä	Å	Ā	Š	Ś	Œ	Š	Ť	Ž	SHY	Ž
Bx	°	à	á	â	ã	ä	å	ā	ś	ś	œ	š	ť	ž	ˆ	ž
Cx	Ř	Á	Â	Ã	Ä	Å	Ā	Ć	Ç	Č	É	Ê	Ë	Ě	Í	Ī
Dx	Đ	Ń	Ň	Ó	Ô	Õ	Ö	×	Ř	Ů	Ú	Û	Ü	Ý	Ț	ß
Ex	í	á	â	ã	ä	å	ā	ć	ç	č	é	ê	ë	ě	í	î
Fx	đ	ñ	ň	ó	ô	õ	ö	×	ř	ů	ú	û	ü	ý	ț	·

Unicode

- různé počty bitů na byte
- prvních 256 znaků odpovídá ISO/IEC 8859-1
- verze 1.0 (1992)
 - všeob. přijetí mnohem později
- verze 15.1 (2023)
 - 149 813 znaků
 - 161 písem

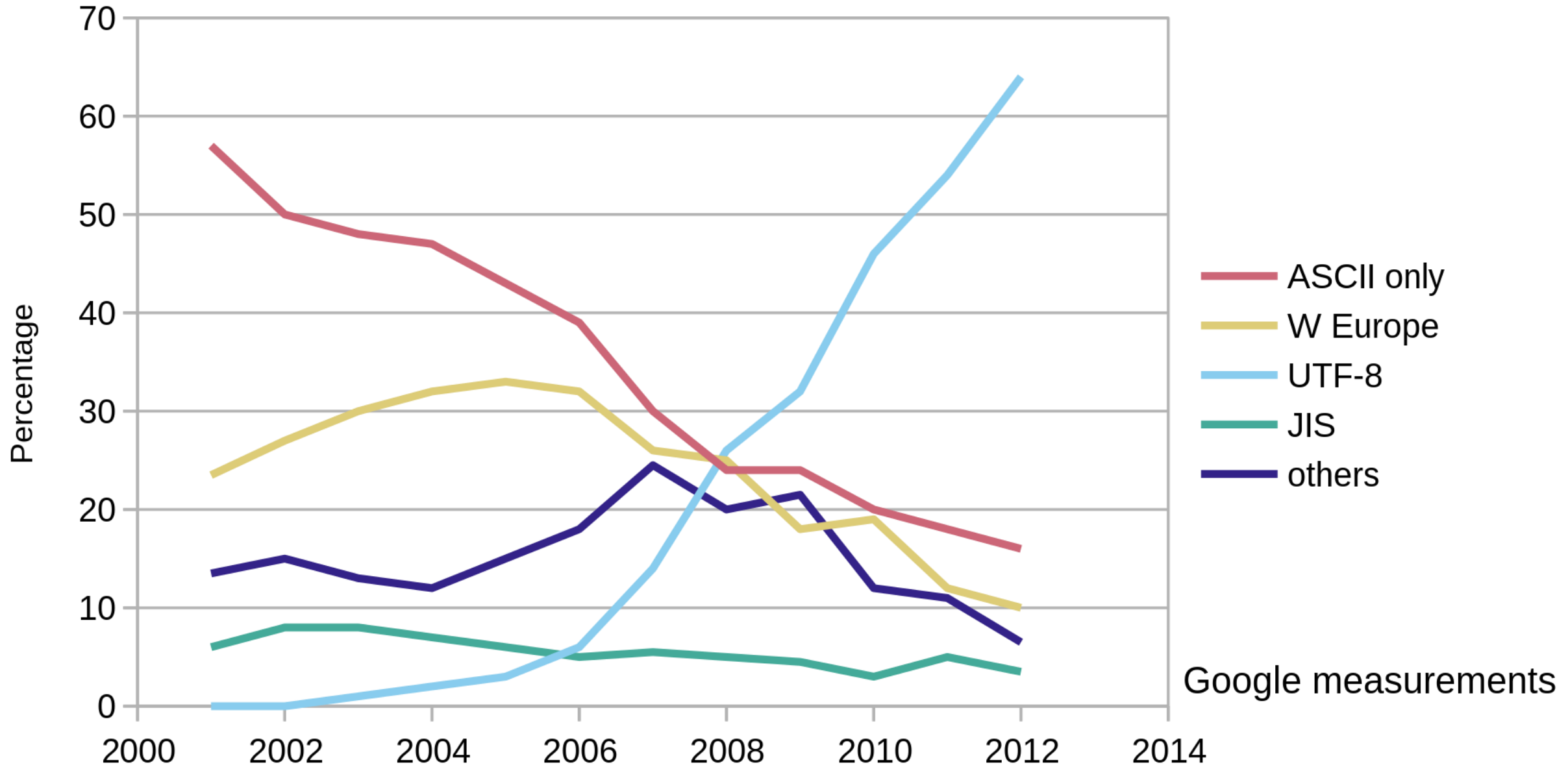
- UTF-8
 - 1-4x 8 bitů pro jeden znak

Code point ↔ UTF-8 conversion

First code point	Last code point	Byte 1	Byte 2	Byte 3	Byte 4
U+0000	U+007F	0xxxxxxx			
U+0080	U+07FF	110xxxxx	10xxxxxx		
U+0800	U+FFFF	1110xxxx	10xxxxxx	10xxxxxx	
U+10000	^[b] U+10FFFF	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

- kompatibilní s ISO/IEC 8859-1
- UTF-16
- (Čína:) GB 18030

Share of web pages with different encodings



Text: formáty

- tabulární data: CSV, TSV, ...
- značkovací jazyky: XML, HTML, LaTeX, ...
- data přiřazující hodnoty k atributům: JSON, ...
- formáty pro dokumenty: DOCX, PDF, ...
- relační databáze: SQL
- datové tabulky: R, Python, ...

- některé formáty oddělují strukturu a formátování, jiné ne

Text: struktura

- zejm. explicitní členění textu pomocí metadat
 - XML
 - HTML
 - JSON

JSON vs XML

[< Previous](#)[Next >](#)

Both JSON and XML can be used to receive data from a web server.

The following JSON and XML examples both define an employees object, with an array of 3 employees:

JSON Example

```
{ "employees": [
  { "firstName": "John", "lastName": "Doe" },
  { "firstName": "Anna", "lastName": "Smith" },
  { "firstName": "Peter", "lastName": "Jones" }
]}
```

XML Example

```
<employees>
  <employee>
    <firstName>John</firstName> <lastName>Doe</lastName>
  </employee>
  <employee>
    <firstName>Anna</firstName> <lastName>Smith</lastName>
  </employee>
  <employee>
    <firstName>Peter</firstName> <lastName>Jones</lastName>
  </employee>
</employees>
```

JSON is Like XML Because

- Both JSON and XML are "self describing" (human readable)
- Both JSON and XML are hierarchical (values within values)
- Both JSON and XML can be parsed and used by lots of programming languages
- Both JSON and XML can be fetched with an XMLHttpRequest

JSON is Unlike XML Because

- JSON doesn't use end tag
- JSON is shorter
- JSON is quicker to read and write
- JSON can use arrays

The biggest difference is:

XML has to be parsed with an XML parser. JSON can be parsed by a standard JavaScript function.

Why JSON is Better Than XML

XML is much more difficult to parse than JSON.
JSON is parsed into a ready-to-use JavaScript object.

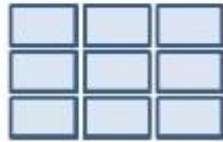
Datové struktury v R

Vector



- 1 column or row of data
- 1 type (numeric or text)

Matrix



- multiple columns and/or rows of data
- 1 type (numeric or text)

Data Frame



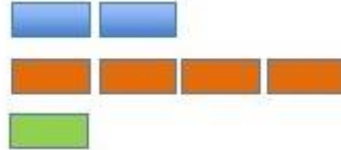
- multiple columns and/or rows of data
- multiple types

Factor



- It is 1 column or row
- Contains "level" data which describes "levels" of classification e.g. class label A or B

List



- a collection of entities with different lengths
- multiple types
- Multiple data structures (vectors, matrices and data frames)

```
x <- "dataset"
```

```
y <- 1:10
```

```
z <- as.numeric(y)
```

```
class(x)
```

```
typeof(x)
```

```
length(x)
```

```
attributes(x)
```

Data frame, Tibble

```
dat <- data.frame(id =  
letters[1:10], x = 1:10, y = 11:20)
```

```
dat
```

```
install.packages("tibble")
```

```
tib <- tibble::as_tibble(dat)
```

```
tib
```

Tidyverse

- "A framework for managing data that aims at making the cleaning and preparing steps [muuuuuuch] easier" (Julien Barnier).
- Main characteristics of a tidy dataset:
 - each variable is a column
 - each observation is a row
 - each value is in a different cell

country	year	cases	population
Afghanistan	1999	75	19987071
Afghanistan	2000	666	20095360
Brazil	1999	31737	172006362
Brazil	2000	80488	174004898
China	1999	212258	1272015272
China	2000	216766	128002583

variables

country	year	cases	population
Afghanistan	1999	75	19987071
Afghanistan	2000	666	20095360
Brazil	1999	31737	172006362
Brazil	2000	80488	174004898
China	1999	212258	1272015272
China	2000	216766	128002583

observations

country	year	cases	population
Afghanistan	1999	75	19987071
Afghanistan	2000	666	20095360
Brazil	1999	31737	172006362
Brazil	2000	80488	174004898
China	1999	212258	1272015272
China	2000	216766	128002583

values

Práce s digitálními daty

- Sběr
- Vytváření
- Obohacování
- Analýza
- Interpretace
- Uložení
- Komunikace

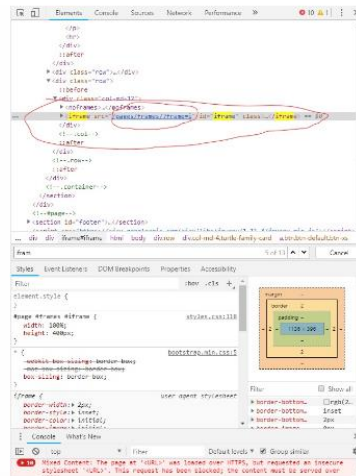
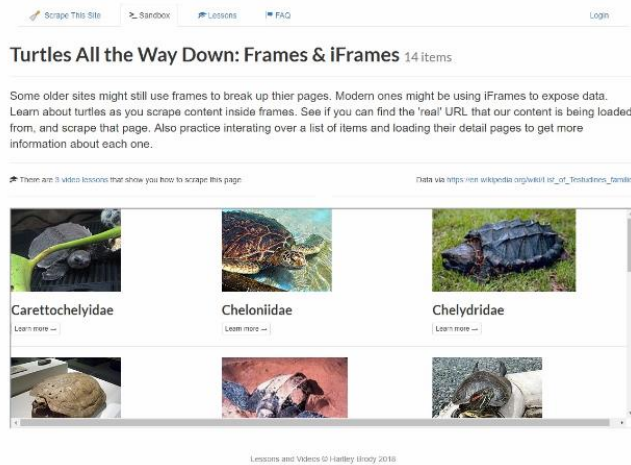
Sběr dat

- webscraping
- načtení tabulárních dat

Webscraping

2 Find the iframe

To extract turtles' name, we need to find the link to the iframe. Let's use Chrome Developer Tool to find the link to iframe: We can use ctrl+F and search for keyword "iframe" and here we find the link.



Let's do this in R and this time it should succeed:

```
library(rvest)
library(stringr)
my_url <- "https://scrapethissite.com/pages/frames/"
#extract source as hown in the image above
iframe_src <- html_session(my_url) %>%
  html_element("#iframe") %>%
  html_attr("src")
#get the url to that iframe
iframe_url <- str_c("https://scrapethissite.com",iframe_src)
#extract turtle names:
turtle_names <- html_session(iframe_url) %>%
  html_elements(".family-name") %>%
  html_text()
print(turtle_names)
```

```
## [1] "Carettochelyidae" "Cheloniidae" "Chelydridae"
## [4] "Dermatemydidae" "Dermochelyidae" "Emydidae"
## [7] "Geoemydidae" "Kinosternidae" "Platysternidae"
## [10] "Testudinidae" "Trionychidae" "Chelidae"
## [13] "Pelomedusidae" "Podocnemididae"
```

Načtení tabulárních dat

Read Tabular Data with readr

`read_*`(file, col_names = TRUE, col_types = NULL, col_select = NULL, id = NULL, locale, n_max = Inf, skip = 0, na = c("", "NA"), guess_max = min(1000, n_max), show_col_types = TRUE) See `?read_delim`

	→	<table border="1"><thead><tr><th>A</th><th>B</th><th>C</th></tr></thead><tbody><tr><td>1</td><td>2</td><td>3</td></tr><tr><td>4</td><td>5</td><td>NA</td></tr></tbody></table>	A	B	C	1	2	3	4	5	NA	read_delim ("file.txt", delim = " ") Read files with any delimiter. If no delimiter is specified, it will automatically guess. To make file.txt, run: <code>write_file("A B C\n1 2 3\n4 5 NA", file = "file.txt")</code>
A	B	C										
1	2	3										
4	5	NA										
	→	<table border="1"><thead><tr><th>A</th><th>B</th><th>C</th></tr></thead><tbody><tr><td>1</td><td>2</td><td>3</td></tr><tr><td>4</td><td>5</td><td>NA</td></tr></tbody></table>	A	B	C	1	2	3	4	5	NA	read_csv ("file.csv") Read a comma delimited file with period decimal marks. <code>write_file("A,B,C\n1,2,3\n4,5,NA", file = "file.csv")</code>
A	B	C										
1	2	3										
4	5	NA										
	→	<table border="1"><thead><tr><th>A</th><th>B</th><th>C</th></tr></thead><tbody><tr><td>1.5</td><td>2</td><td>3</td></tr><tr><td>4.5</td><td>5</td><td>NA</td></tr></tbody></table>	A	B	C	1.5	2	3	4.5	5	NA	read_csv2 ("file2.csv") Read semicolon delimited files with comma decimal marks. <code>write_file("A;B;C\n1,5;2;3\n4,5;5;NA", file = "file2.csv")</code>
A	B	C										
1.5	2	3										
4.5	5	NA										
	→	<table border="1"><thead><tr><th>A</th><th>B</th><th>C</th></tr></thead><tbody><tr><td>1</td><td>2</td><td>3</td></tr><tr><td>4</td><td>5</td><td>NA</td></tr></tbody></table>	A	B	C	1	2	3	4	5	NA	read_tsv ("file.tsv") Read a tab delimited file. Also <code>read_table()</code> . read_fwf ("file.tsv", <code>fwf_widths</code> (c(2, 2, NA))) Read a fixed width file. <code>write_file("A\tB\tC\n1\t2\t3\n4\t5\tNA\n", file = "file.tsv")</code>
A	B	C										
1	2	3										
4	5	NA										

USEFUL READ ARGUMENTS

<table border="1"><thead><tr><th>A</th><th>B</th><th>C</th></tr></thead><tbody><tr><td>1</td><td>2</td><td>3</td></tr><tr><td>4</td><td>5</td><td>NA</td></tr></tbody></table>	A	B	C	1	2	3	4	5	NA	No header <code>read_csv("file.csv", col_names = FALSE)</code>	<table border="1"><thead><tr><th>1</th><th>2</th><th>3</th></tr></thead><tbody><tr><td>4</td><td>5</td><td>NA</td></tr></tbody></table>	1	2	3	4	5	NA	Skip lines <code>read_csv("file.csv", skip = 1)</code>
A	B	C																
1	2	3																
4	5	NA																
1	2	3																
4	5	NA																
<table border="1"><thead><tr><th>x</th><th>y</th><th>z</th></tr></thead><tbody><tr><td>1</td><td>2</td><td>3</td></tr><tr><td>4</td><td>5</td><td>NA</td></tr></tbody></table>	x	y	z	1	2	3	4	5	NA	Provide header <code>read_csv("file.csv", col_names = c("x", "y", "z"))</code>	<table border="1"><thead><tr><th>A</th><th>B</th><th>C</th></tr></thead><tbody><tr><td>1</td><td>2</td><td>3</td></tr></tbody></table>	A	B	C	1	2	3	Read a subset of lines <code>read_csv("file.csv", n_max = 1)</code>
x	y	z																
1	2	3																
4	5	NA																
A	B	C																
1	2	3																
	Read multiple files into a single table <code>read_csv(c("f1.csv", "f2.csv", "f3.csv"), id = "origin_file")</code>	<table border="1"><thead><tr><th>A</th><th>B</th><th>C</th></tr></thead><tbody><tr><td>NA</td><td>2</td><td>3</td></tr><tr><td>4</td><td>5</td><td>NA</td></tr></tbody></table>	A	B	C	NA	2	3	4	5	NA	Read values as missing <code>read_csv("file.csv", na = c("1"))</code>						
A	B	C																
NA	2	3																
4	5	NA																
	Specify decimal marks <code>read_delim("file2.csv", locale = locale(decimal_mark = ";"))</code>																	

Import Spreadsheets

with readxl



READ EXCEL FILES

A	B	C	D	E
1	x1	x2	x3	x4
2	x	z	8	
3	y	7	9	10

`read_excel`(path, sheet = NULL, range = NULL) Read a .xls or .xlsx file based on the file extension. See front page for more read arguments. Also `read_xls()` and `read_xlsx()`.
`read_excel("excel_file.xlsx")`

READ SHEETS

A	B	C	D	E
e1	e2	e3		

`read_excel`(path, sheet = NULL) Specify which sheet to read by position or name.
`read_excel`(path, sheet = 1)
`read_excel`(path, sheet = "s1")

e1	e2	e3
----	----	----

`excel_sheets`(path) Get a vector of sheet names.
`excel_sheets("excel_file.xlsx")`

A	B	C	D	E
	A	B	C	D
e1				

To read multiple sheets:
1. Get a vector of sheet names from the file path.
2. Set the vector names to be the sheet names.
3. Use `purrr::map()` and `purrr::list_rbind()` to read multiple files into one

```
path <- "your_file_path.xlsx" data frame.
path |>
  excel_sheets() |>
  set_names() |>
  map(read_excel, path = path) |>
  list_rbind()
```

READXL COLUMN SPECIFICATION

Column specifications define what data type each column of a file will be imported as.

Use the `col_types` argument of `read_excel()` to set the column specification.

Guess column types

To guess a column type, `read_excel()` looks at the first 1000 rows of data. Increase with the `guess_max` argument.
`read_excel`(path, `guess_max` = Inf)

Set all columns to same type, e.g. character

`read_excel`(path, `col_types` = "text")

Set each column individually

`read_excel`(
 path,
 `col_types` = c("text", "guess", "guess", "numeric")
)

COLUMN TYPES

logical	numeric	text	date	list
TRUE	2	hello	1947-01-08	hello
FALSE	3.45	world	1956-10-21	1

- skip
- guess
- logical
- numeric
- text
- date
- list

Use `list` for columns that include multiple data types. See `tidyr` and `purrr` for list-column data.

Načtení tabulárních dat

with googlesheets4

READ SHEETS

	A	B	C	D	E
1	x1	x2	x3	x4	x5
2	x		z	8	
3	y	7	9	10	

x1	x2	x3	x4	x5
x	NA	z	8	NA
y	7	NA	9	10

`read_sheet(ss, sheet = NULL, range = NULL)`
Read a sheet from a URL, a Sheet ID, or a dribble from the googledrive package. See front page for more read arguments. Same as `range_read()`.

SHEETS METADATA

URLs are in the form:
`https://docs.google.com/spreadsheets/d/
SPREADSHEET_ID/edit#gid=SHEET_ID`

`gs4_get(ss)` Get spreadsheet meta data.
`gs4_find(...)` Get data on all spreadsheet files.

`sheet_properties(ss)` Get a tibble of properties for each worksheet. Also `sheet_names()`.

WRITE SHEETS

`write_sheet(data, ss = NULL, sheet = NULL)`
Write a data frame into a new or existing Sheet.

`gs4_create(name, ..., sheets = NULL)` Create a new Sheet with a vector of names, a data frame, or a (named) list of data frames.

`sheet_append(ss, data, sheet = 1)` Add rows to the end of a worksheet.



GOOGLESHEETS4 COLUMN SPECIFICATION

Column specifications define what data type each column of a file will be imported as.

Use the `col_types` argument of `read_sheet()`/
`range_read()` to set the column specification.

Guess column types

To guess a column type `read_sheet()`/
`range_read()` looks at the first 1000 rows of data.
Increase with `guess_max`.
`read_sheet(path, guess_max = Inf)`

Set all columns to same type, e.g. character

`read_sheet(path, col_types = "c")`

Set each column individually

col types: skip, guess, integer, logical, character
`read_sheets(ss, col_types = "_?ilc")`

COLUMN TYPES

l	n	c	D	L
TRUE	2	hello	1947-01-08	hello
FALSE	3.45	world	1956-10-21	1

- skip - "." or "-"
- guess - "?"
- logical - "l"
- integer - "i"
- double - "d"
- numeric - "n"
- date - "D"
- datetime - "T"
- character - "c"
- list-column - "L"
- cell - "C" Returns list of raw cell data.

Use list for columns that include multiple data types. See `tidyr` and `purrr` for list-column data.

Vytváření dat

- úprava dat pomocí LLM
- transformace
 - regulární výrazy (VSCode)
 - práce se sloupci (R, Python)
- čištění

Úprava dat pomocí LLM

- prompt

Encode the following table with popular baby names of 2022 in JSON:

Rank	Male name	Female name
1	Liam	Olivia
2	Noah	Emma
3	Oliver	Charlotte
4	James	Amelia
5	Elijah	Sophia
6	William	Isabella
7	Henry	Ava
8	Lucas	Mia
9	Benjamin	Evelyn
10	Theodore	Luna

- odpověď

```
{
  "names_2022": {
    "male_names": [
      {"rank": 1, "name": "Liam"},
      {"rank": 2, "name": "Noah"},
      {"rank": 3, "name": "Oliver"},
      {"rank": 4, "name": "James"},
      {"rank": 5, "name": "Elijah"},
      {"rank": 6, "name": "William"},
      {"rank": 7, "name": "Henry"},
      {"rank": 8, "name": "Lucas"},
      {"rank": 9, "name": "Benjamin"},
      {"rank": 10, "name": "Theodore"}
    ],
    "female_names": [
      {"rank": 1, "name": "Olivia"},
      {"rank": 2, "name": "Emma"},
      {"rank": 3, "name": "Charlotte"},
      {"rank": 4, "name": "Amelia"},
      {"rank": 5, "name": "Sophia"},
      {"rank": 6, "name": "Isabella"},
      {"rank": 7, "name": "Ava"},
      {"rank": 8, "name": "Mia"},
      {"rank": 9, "name": "Evelyn"},
      {"rank": 10, "name": "Luna"}
    ]
  }
}
```

Obhohacování dat

- reconciling (OpenRefine)
- geolokace
- NER (named entity recognition)
- textové korpusy: anotace
 - tokenizace, lemmatizace, rozpoznání slovních druhů


```
{
  "names_2022": {
    "male_names": [
      {"rank": 1, "name": "Liam", "wikipedia_link": "https://en.wikipedia.org/wiki/Liam"},
      {"rank": 2, "name": "Noah", "wikipedia_link": "https://en.wikipedia.org/wiki/Noah_(name)"},
      {"rank": 3, "name": "Oliver", "wikipedia_link": "https://en.wikipedia.org/wiki/Oliver_(name)"},
      {"rank": 4, "name": "James", "wikipedia_link": "https://en.wikipedia.org/wiki/James_(name)"},
      {"rank": 5, "name": "Elijah", "wikipedia_link": "https://en.wikipedia.org/wiki/Elijah_(given_name)"},
      {"rank": 6, "name": "William", "wikipedia_link": "https://en.wikipedia.org/wiki/William_(name)"},
      {"rank": 7, "name": "Henry", "wikipedia_link": "https://en.wikipedia.org/wiki/Henry_(given_name)"},
      {"rank": 8, "name": "Lucas", "wikipedia_link": "https://en.wikipedia.org/wiki/Lucas_(given_name)"},
      {"rank": 9, "name": "Benjamin", "wikipedia_link": "https://en.wikipedia.org/wiki/Benjamin"},
      {"rank": 10, "name": "Theodore", "wikipedia_link": "https://en.wikipedia.org/wiki/Theodore"}
    ],
    "female_names": [
      {"rank": 1, "name": "Olivia", "wikipedia_link": "https://en.wikipedia.org/wiki/Olivia"},
      {"rank": 2, "name": "Emma", "wikipedia_link": "https://en.wikipedia.org/wiki/Emma_(given_name)"},
      {"rank": 3, "name": "Charlotte", "wikipedia_link": "https://en.wikipedia.org/wiki/Charlotte_(given_name)"},
      {"rank": 4, "name": "Amelia", "wikipedia_link": "https://en.wikipedia.org/wiki/Amelia_(given_name)"},
      {"rank": 5, "name": "Sophia", "wikipedia_link": "https://en.wikipedia.org/wiki/Sophia_(name)"},
      {"rank": 6, "name": "Isabella", "wikipedia_link": "https://en.wikipedia.org/wiki/Isabella_(given_name)"},
      {"rank": 7, "name": "Ava", "wikipedia_link": "https://en.wikipedia.org/wiki/Ava_(name)"},
      {"rank": 8, "name": "Mia", "wikipedia_link": "https://en.wikipedia.org/wiki/Mia_(given_name)"},
      {"rank": 9, "name": "Evelyn", "wikipedia_link": "https://en.wikipedia.org/wiki/Evelyn_(name)"},
      {"rank": 10, "name": "Luna", "wikipedia_link": "https://en.wikipedia.org/wiki/Luna_(name)"}
    ]
  }
}
```

Analýza a interpretace dat

- výpočty (tabulky s čísly)
- vizualizace dat

- více ve cvičení 😊

Uložení dat

- GitHub
- Zenodo

Text-translation-using-Mistral-7B Public Watch 1 Fork 2 Star 3

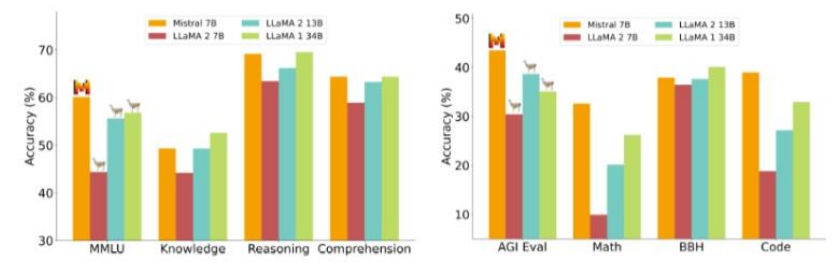
main 1 Branch 0 Tags Go to file Add file Code

Table listing repository files: LLM_for_Translation.ipynb, README.md

README

Text Translation using Mistral 7B

Mistral 7B is a powerful language model that has revolutionized natural language processing. It's open-source and free to use, making it popular among researchers and developers.



Mistral 7B has shown impressive performance on a number of text translation benchmarks. For example, on the WMT14 English-to-German translation task, Mistral 7B achieved a BLEU score of 30.4, which is better than the state-of-the-art at the time.

This project is just a practice attempt at understanding how to implement the Mistral 7B model to translate text given by user from English to French.

Future scope of the project includes integrating the code with a frontend so users can interact with the model on a deployed website and the results can be more presentable and accessible.

About

Text Translation using Mistral 7B

- javascript, css, python, html, translation, text, large-language-models, llm, llms

- Readme, Activity, 3 stars, 1 watching, 2 forks, Report repository

Releases

No releases published

Packages

No packages published

Languages



Featured communities



Biodiversity Literature Repository

A community to share publications related to bio-systematics.

[Browse](#)

Recent uploads

March 13, 2024 (v1)

Presentation

Open

NOAA NESDIS Operational Sea Surface Temperatures (SSTs)

Baker-Yeboah, Sheekela; Casey, Kenneth; Harris, Andy

This presentation was given by Sheekela Baker-Yeboah on March 7th, 2024 for a GHRSSST Talk. Find the event details and the recording of the GHRSSST Talk on the GHRSSST Website here.

Uploaded on March 13, 2024

Part of [Group for High Resolution Sea Surface Temperature \(GHRSSST\)](#)

0 0

March 13, 2024 (2024-03-13)

Dataset

Open

SARS-CoV-2 Sequenzdaten aus Deutschland

Robert Koch-Institut

Ein zentraler Bestandteil einer erfolgreichen Erregersurveillance ist das Verständnis der Verbreitung eines Erregers sowie seiner pathogenen Eigenschaften. Hierbei stellt das

Why use Zenodo?

- **Safe** — your research is stored safely for the future in CERN's Data Centre for as long as CERN exists.
- **Trusted** — built and operated by CERN and OpenAIRE to ensure that everyone can join in Open Science.
- **Citeable** — every upload is assigned a Digital Object Identifier (DOI), to make them citable and trackable.
- **No waiting time** — Uploads are made available online as soon as you hit publish, and your DOI is registered within seconds.
- **Open or closed** — Share e.g. anonymized clinical trial data with only medical professionals via our restricted access mode.
- **Versioning** — Easily update your dataset with our versioning feature.
- **GitHub integration** — Easily preserve your GitHub repository in Zenodo.
- **Usage statistics** — All uploads display standards compliant usage statistics

Komunikovat data

- RMarkdown
- Quarto



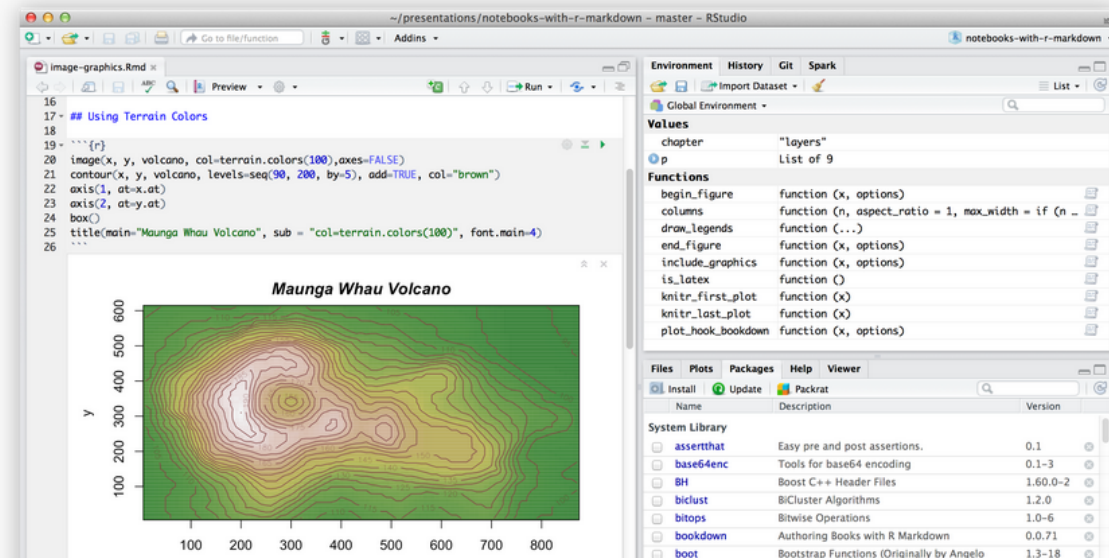
Analyze. Share. Reproduce.

Your data tells a story. Tell it with R Markdown.

Turn your analyses into high quality documents, reports, presentations and dashboards.

R Markdown documents are fully reproducible.

Use a productive [notebook interface](#) to weave together narrative text and code to produce elegantly formatted output. Use [multiple languages](#) including R, Python, and SQL.



The screenshot shows the RStudio interface with a notebook open. The code in the editor is as follows:

```
16
17 # Using Terrain Colors
18
19 ```{r}
20 image(x, y, volcano, col=terrain.colors(100), axes=FALSE)
21 contour(x, y, volcano, levels=seq(90, 200, by=5), add=TRUE, col="brown")
22 axis(1, at=x.at)
23 axis(2, at=y.at)
24 box()
25 title(main="Maunga Whau Volcano", sub = "col=terrain.colors(100)", font.main=4)
26 ...
```

The plot displays a terrain map of Maunga Whau Volcano with axes ranging from 0 to 800 on the x-axis and 0 to 600 on the y-axis. The terrain is color-coded from green (low elevation) to brown (high elevation). The plot is titled "Maunga Whau Volcano" with a subtitle "col=terrain.colors(100)".

The right-hand side of the RStudio window shows the Environment pane with the following variables:

Variable	Value
chapter	"Layers"
p	List of 9

The Functions pane lists the following functions:

Function	Signature
begin_figure	function(x, options)
columns	function(n, aspect_ratio = 1, max_width = if (n ...
draw_legends	function(...)
end_figure	function(x, options)
include_graphics	function(x, options)
is_latex	function()
knitr_first_plot	function(x)
knitr_last_plot	function(x)
plot_hook_bookdown	function(x, options)

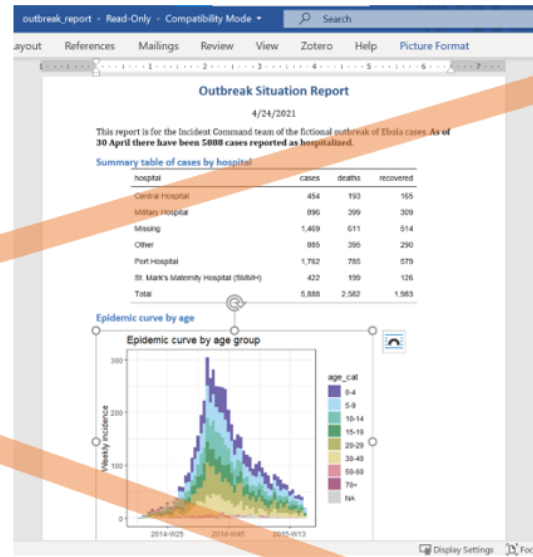
The Files pane shows the following packages installed:

Name	Description	Version
assertthat	Easy pre and post assertions.	0.1
base64enc	Tools for base64 encoding	0.1-3
BH	Boost C++ Header Files	1.60.0-2
biclust	BiCluster Algorithms	1.2.0
bitops	Bitwise Operations	1.0-6
bookdown	Authoring Books with R Markdown	0.0.71
boot	Bootstrap Functions (Originally by Angelo	1.3-18

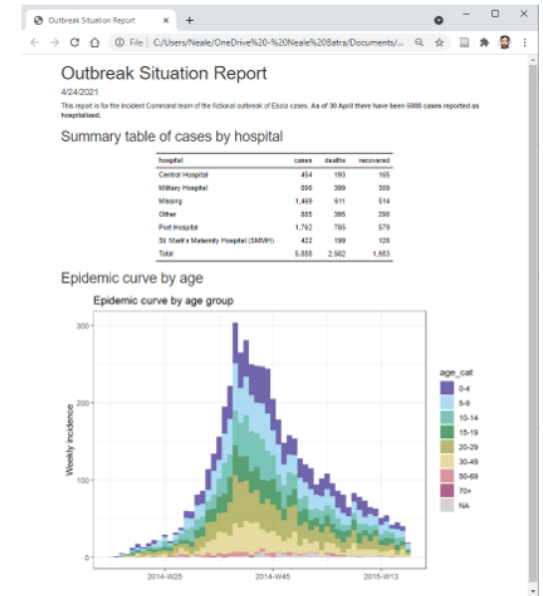
R Markdown script

```
1 ---  
2 title: "Outbreak Situation Report"  
3 date: "4/24/2021"  
4 output: word_document  
5 ---  
6  
7 ***[r setup, echo=FALSE]  
8 pacman::p_load(rio, here, tidyverse, incidence2, flextable)  
9 line1list <- rio::import(here::here("data", "case_line1lists", "line1list_cleaned.rds"))  
10  
11  
12 This report is for the Incident Command team of the fictional outbreak of Ebola cases.  
13 **As of 30 April there have been 5888 cases reported as hospitalized.**  
14  
15 ## Summary table of cases by hospital  
16 ***[r, echo=F, out.height="75%"]  
17 line1list %>%  
18   filter(!is.na(hospital)) %>%  
19   group_by(hospital) %>%  
20   summarise(cases = n(),  
21             deaths = sum(outcome == "Death", na.rm=T),  
22             recovered = sum(outcome == "Recover", na.rm=T)) %>%  
23   adorn_totals() %>%  
24   qflextable()  
25  
26  
27 ## Epidemic curve by age  
28  
29 ***[r, echo=F, warning=F, message=F, out.height = "75%", out.width="100%"]  
30 # create epidemic  
31 age_outbreak <- incidence(  
32   line1list,  
33   date_index = date_onset, # date of onset for x-axis  
34   interval = "week",      # weekly aggregation of cases  
35   groups = age_cat)  
36  
37 # plot  
38 plot(age_outbreak, n_breaks = 3, fill = age_cat, col_pal = muted, title = "Epidemic  
39   curve by age group")  
40
```

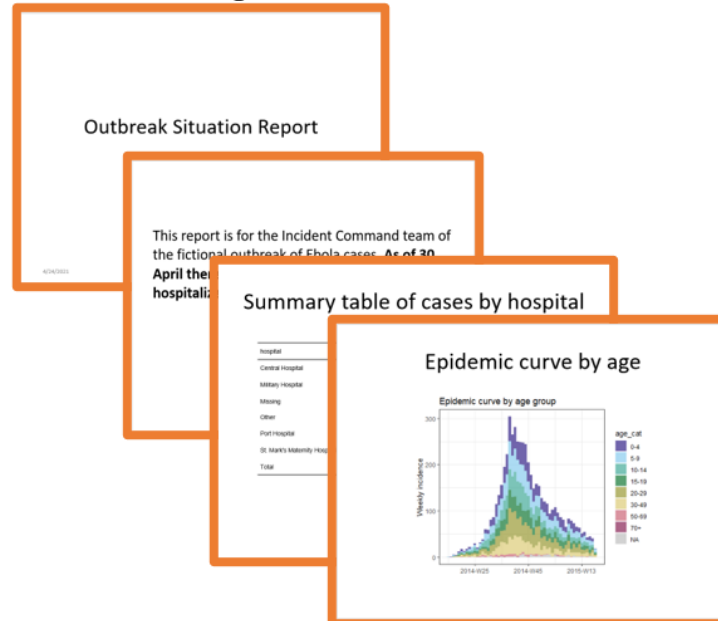
Word



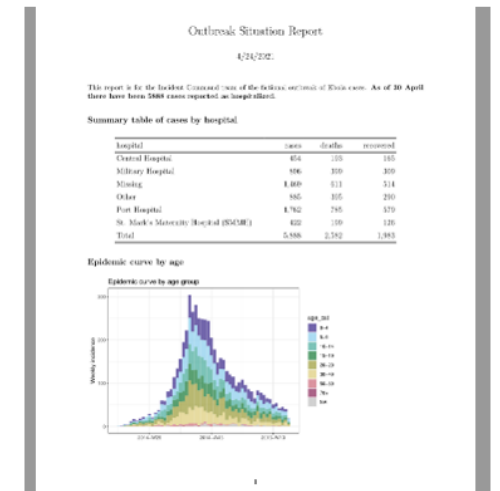
HTML



Powerpoint



PDF



Share your report as a...



html



ioslides



reveal.js



rtf



tuftes handout



book



pdf



dashboard



slidy



markdown



package vignette



website



Word



notebook



beamer



latex



custom template



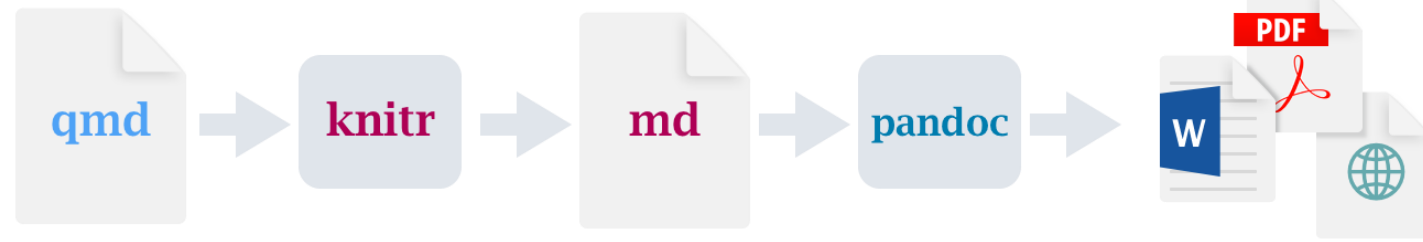
shiny app



01:00



Quarto



The screenshot shows the Quarto Visual editor interface. The top toolbar includes options for rendering and running. The main content area displays a rendered document with the following sections:

```
title: "Hello, Quarto"
format: html
editor: visual
```

```
{r}
#| label: load-packages
#| include: false


library(tidyverse)
library(palmerpenguins)
```

Meet Quarto

Quarto enables you to weave together content and executable code into a finished document. To learn more about Quarto see <https://quarto.org>.

Meet the penguins

The `penguins` data from the [palmerpenguins](#) package contains size measurements for `r` `nrow(penguins)` penguins from three species observed on three islands in the Palmer Archipelago, Antarctica.



The plot below shows the relationship between flipper and bill lengths of these penguins.

```
{r}
#| label: plot-penguins
#| warning: false
#| echo: false

ggplot(penguins,
  aes(x = flipper_length_mm, y = bill_length_mm)) +
  geom_point(aes(color = species, shape = species)) +
```

The screenshot shows the Quarto Source editor interface. The top toolbar includes options for rendering and running. The main content area displays the source code for the document:

```
1 ---
2 title: "Hello, Quarto"
3 format: html
4 editor: visual
5 ---
6
7 ```{r}
8 #| label: load-packages
9 #| include: false
10
11 library(tidyverse)
12 library(palmerpenguins)
13 ```
14
15 ## Meet Quarto
16
17 Quarto enables you to weave together content and executable code into a finished
18 document.
19 To learn more about Quarto see <https://quarto.org>.
20
21 ## Meet the penguins
22
23 {style="float:right;" fig-alt="Illustration of three
24 species of Palmer Archipelago penguins: Chinstrap, Gentoo, and Adelie. Artwork by
25 @allison_horst." width="300"}
26
27 The `penguins` data from the
28 https://allisonhorst.github.io/palmerpenguins "palmerpenguins
29 R package" package contains size measurements for `r` nrow(penguins)
30 penguins from three species observed on three islands in the Palmer Archipelago,
31 Antarctica.
32
33 The plot below shows the relationship between flipper and bill lengths of these
34 penguins.
35
36 ```{r}
37 #| label: plot-penguins
38 #| warning: false
39 #| echo: false
```

markdown.qmd

Render on Save

Source Visual

Text formatting

italic **bold** underline ~~strikeout~~ SMALL CAPS code superscript² and subscript₂

Headings

1st Level Header

2nd Level Header

3rd Level Header

Lists

- Bulleted list item 1
- Item 2
 - Item 2a
 - Item 2b


- Numbered list item 1
- Item 2. The numbers are incremented automatically in the output.

Links and images

<http://example.com>

[linked phrase](#)

<http://example.com>



optional caption text

w: 320 h: 77 px Lock ratio

Tables

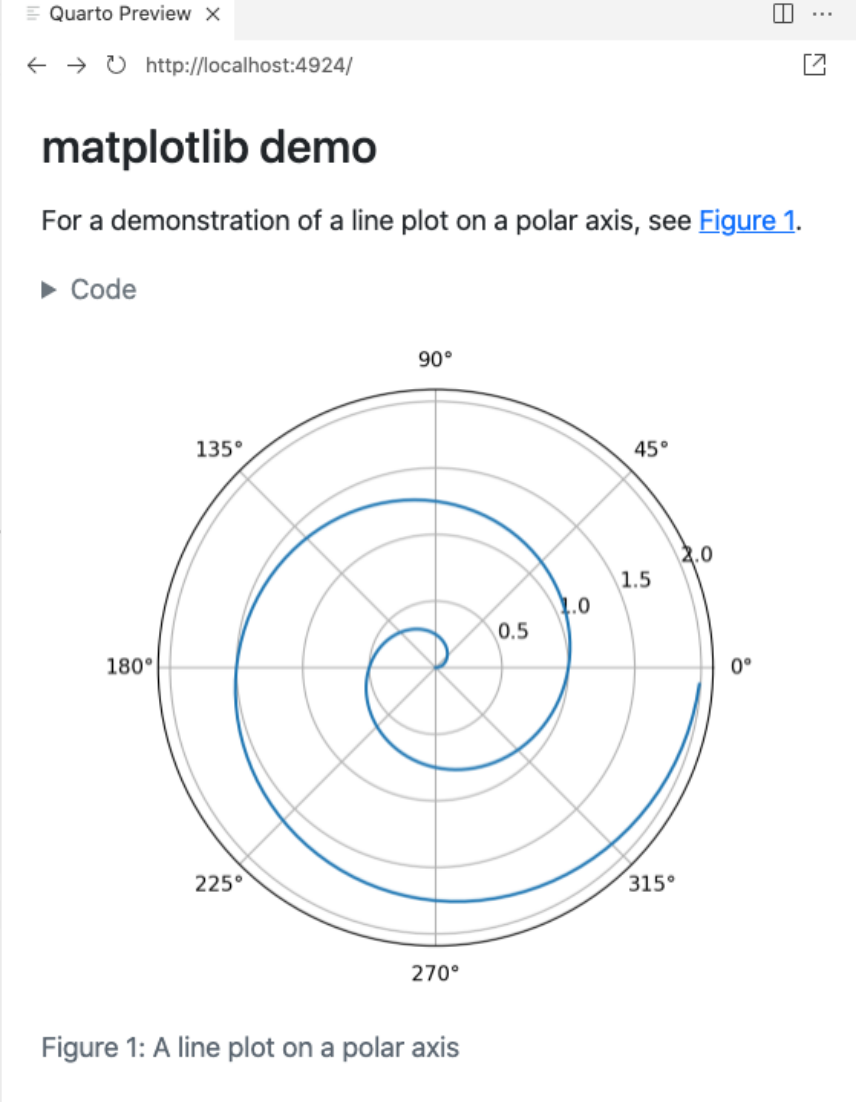
First Header	Second Header
Content Cell	Content Cell
Content Cell	Content Cell

/type to search...

- R Code Chunk**
Executable R chunk
- Python Code Chunk**
Executable Python chunk
- Div...**
Block containing other con...
- Bullet List**
List using bullets for items
- Numbered List**
List using numbers for items
- Heading 1**
Part heading

Tables 2

```
demo.qmd x
Users > jjallaire > Desktop > demo.qmd
1 ---
2 title: "matplotlib demo"
3 format:
4   html:
5     code-fold: true
6   jupyter: python3
7 ---
8
9 For a demonstration of a line plot on a polar axis, see
  @fig-polar.
10
11 > Run Cell
12 ```{python}
13 #| label: fig-polar
14 #| fig-cap: "A line plot on a polar axis"
15
16 import numpy as np
17 import matplotlib.pyplot as plt
18
19 r = np.arange(0, 2, 0.01)
20 theta = 2 * np.pi * r
21 fig, ax = plt.subplots(
22     subplot_kw = {'projection': 'polar'}
23 )
24 ax.plot(theta, r)
25 ax.set_rticks([0.5, 1, 1.5, 2])
26 ax.grid(True)
27 plt.show()
28
29
```



TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE QUARTO

```
document-css: false
link-citations: true
lang: en
title: matplotlib demo
jupyter: python3

Output created: demo.html

Watching files for changes
```

+ v ^ x

- zsh
- zsh
- Quarto Pre...