Text processing for data wrangling

featuring Regular Expressions

George MacKerron Economics, Sussex 2 October 2019

Testimonial

- *“saved me hours and hours and hours on my PhD”*
  — Antonia Schwarz

What I mean by *text*

- Plain text (anything copied into Notepad or TextEdit)
- Code
  - Stata .do files, R scripts, Python, Bash, C, Java, …
  - LaTeX
  - HTML, CSS, JavaScript
- Data
  - CSV, TSV, fixed-width formats
  - XML and its many children
    - XHTML, KML, XLSX, .plist, …
  - JSON
  - GeoJSON, Esri (ASCII) grid

Problems

- **My data is mangled**
  - Weird symbols, extra spaces, no carriage returns …
    (aka: My co-author says the data is mangled, but it looks fine to me …)
- **I have to find, replace or extract some things that follow a pattern**
  - Postcodes, email addresses, identifiers, dates, …
- **I need to match some things in the presence of errors or inconsistencies**
  - Names or identifiers of countries, firms, teams, products, …

Solutions

- Text encodings
- Regular Expressions
- Trigrams, Levenshtein distance

Key tool

- A good text editor
- Sublime Text (Windows, Mac & Linux)

Who am I?

- George MacKerron, Senior Lecturer
- Teaching: behavioural (Y3 U/G)
- Research: subjective wellbeing (happiness) and environment
- Background: BA Archaeology & Anthropology, software development, CTO
- Ask me about: happiness economics, programming (web, Ruby, Objective-C), SQL, AWS

https://www.youtube.com/watch?v=_Wo2M8Z-ULM

@jawj

LSE

mappiness.org.uk

Problem 1

My data is mangled

You wanted: “Naïvely, the café charged only £1 for a cachaça”.

You got:

1. â€œNaÃ¯vely, the cafÃ© charged only Â£1 for a cachaÃ§aâ€.

2. N a ? v e l y , t h e c a f ? c h a r g e d o n l y ? 1 f o r a c a c h a ? a .

3.

4. 胢亜썡皯汥ⱹ琠敨挠晡꧃挠慨杲摥漠汮⁹ꏂ‱潦慣档썡憧胢⺝

Why?

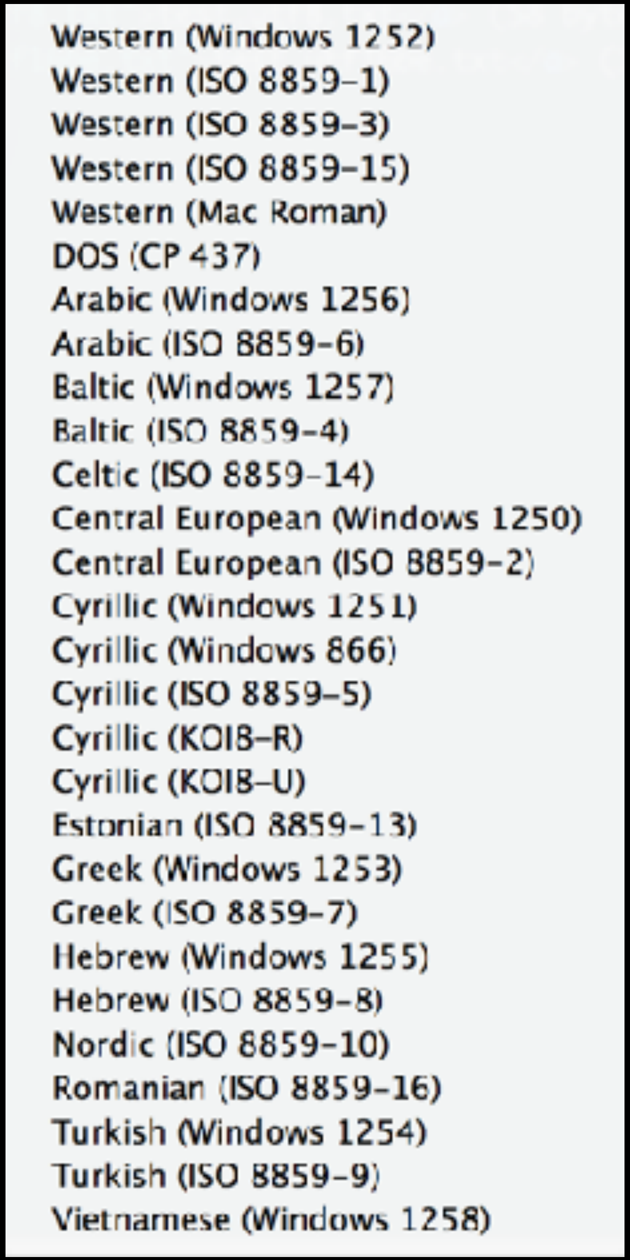
- Computers work in numbers: bits (0/1), bytes (8 bits, 0–255), ‘words’ (16 bits, 0–65,536), and so on

- We need a mapping from numbers to characters.

128	64	32	16	8	4	2	1	100	10	1
0	1	1	1	0	0	1	1	1	1	5
0	1	1	1	0	1	0	1	1	1	7
0	1	1	1	0	0	1	1	1	1	5
0	1	1	1	0	0	1	1	1	1	5
0	1	1	0	0	1	0	1	1	0	1
0	1	1	1	1	0	0	0	1	2	0

‘Code pages’

- One byte (0 – 255) per character
- Fine for many individual languages
  — e.g. English: A-Z (26), a-z (26), 0–9 (10), plus some punctuation and accents
- **But …**
  - Different languages and different systems use different mappings
  - You need to know which mapping was used
    — and this metadata can’t go *in* a plain text file
  - You can’t mix languages (especially Latin, Cyrillic, Hebrew, Arabic, …)
  - Chinese! Japanese! Korean!


Western (Windows 1252)
Western (ISO 8859-1)
Western (ISO 8859-3)
Western (ISO 8859-15)
Western (Mac Roman)
DOS (CP 437)
Arabic (Windows 1256)
Arabic (ISO 8859-6)
Baltic (Windows 1257)
Baltic (ISO 8859-4)
Celtic (ISO 8859-14)
Central European (Windows 1250)
Central European (ISO 8859-2)
Cyrillic (Windows 1251)
Cyrillic (Windows 866)
Cyrillic (ISO 8859-5)
Cyrillic (KOI8-R)
Cyrillic (KOI8-U)
Estonian (ISO 8859-13)
Greek (Windows 1253)
Greek (ISO 8859-7)
Hebrew (Windows 1255)
Hebrew (ISO 8859-8)
Nordic (ISO 8859-10)
Romanian (ISO 8859-16)
Turkish (Windows 1254)
Turkish (ISO 8859-9)
Vietnamese (Windows 1258)

Unicode

- Unicode (late 80s – present) attempts to assign a name and single unique number to every character in every language (including historical, academic, etc.)

- > 110,000 characters covering
  > 100 scripts and symbols (including emoji) — and growing

- e.g.
  U+062C (1580)
  = ج ARABIC LETTER JEEM

  U+1F92F (129327)
  = 🤯 SHOCKED FACE WITH EXPLODING HEAD

Unicode formats

- Bytes per character:
  - Fixed 4: UTF-32BE, UTF-32LE (BOM)
  - Variable 2+: UTF-16BE, UTF-16LE (BOM)
  - Fixed 2: UCS-2
  - Variable 1+: **UTF-8** <— use this if you can

So …

- When your data is mangled, it’s because the text was encoded with one mapping and decoded with another
- You need to find the right encoding to decode it with

My data is mangled

1. â€œNaÃ¯vely, the cafÃ© charged only Â£1 for a cachaÃ§aâ€.

2. N a ? v e l y , t h e c a f ? c h a r g e d o n l y ? 1 f o r a c a c h a ? a .

3.

4. 䡎愀쌩奸洿ℷ琠敫挠晨[illegible] 挠慨杲敤漠⁹ɛ‱潦⁲愠挠捨愠慼
   慣档쌩慡䡎月

How to fix it

- Open your file in Sublime Text
- Sublime Text may figure out the right encoding for you
- Or *File > Reopen with Encoding …* until it looks right
- If necessary, *File > Save with Encoding …*

Also: line-endings

- I have eaten▯the plums▯that were in▯the icebox▯ ▯and which▯you were probably▯saving▯for breakfast▯ ▯Forgive me▯they were delicious▯so sweet▯and so cold▯

How to fix it

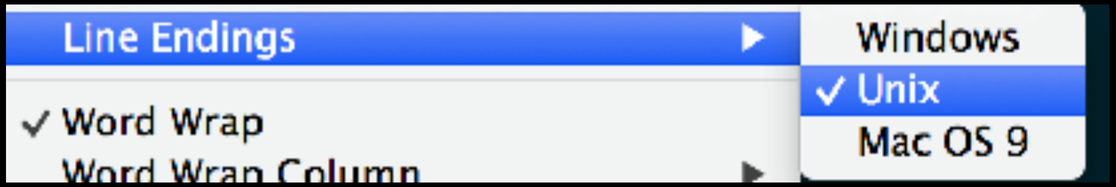
- Different systems have different ideas about how to start a new line

  - Mac OS X, Unix, Linux: \n = LF, Line Feed, character 10

  - Mac OS 9 and before: \r = CR, Carriage Return, character 13

  - Windows: \r\n

- Sublime Text can fix this for you too



Extremely short practical

- Install (portable/not portable) Sublime Text from https://www.sublimetext.com/3
- Download the three test files from Canvas
- Try opening and reopening them with various encodings

Problem 2

I have to find, replace or extract some things that follow a pattern

- Find names or addresses or repeated words
- Rearrange data fields
  e.g. 0.005,51.5,0 0.004,50.3,0 …
  ➛ 51.5 0.005,50.3 0.004, …
- Fix a broken CSV file
- Find hard-to-spot errors in my thesis

Regular Expressions

- Patterns that match a variety of actual text
- Letters, numbers and spaces stand for themselves, but some other characters have special meanings
  - e.g. Tom matches only the text Tom
  - e.g. Tom|Dick|Harry matches any of Tom or Dick or Harry
  - e.g. T.m matches Tom, Tim, T5m, T m, …
    i.e. T *any character* m

WHENEVER I LEARN A NEW SKILL I CONCOCT ELABORATE FANTASY SCENARIOS WHERE IT LETS ME SAVE THE DAY.
OH NO! THE KILLER MUST HAVE FOLLOWED HER ON VACATION!
BUT TO FIND THEM WE'D HAVE TO SEARCH THROUGH 200 MB OF EMAILS LOOKING FOR SOMETHING FORMATTED LIKE AN ADDRESS!
IT'S HOPELESS!
EVERYBODY STAND BACK.
I KNOW REGULAR EXPRESSIONS.
PERL!
TAP TAP

http://xkcd.com/208/

Let’s do it!

- We’ll …
  - Download *The Adventures of Sherlock Holmes*
  - Open in (or Copy+Paste into) Sublime Text
  - Find street addresses using Regular Expressions
- But first …

Character classes

- Characters or ranges of characters inside [] square brackets match any of those characters
  - e.g. [hnc]ow matches how, now and cow
  - e.g. [1-4] matches 1, 2, 3 and 4
  - e.g. [A-Za-z] matches any single letter of the alphabet, upper- or lower-case

Class shortcuts

- \d means a digit, [0-9]
- \s means a whitespace character, [\t\r\n] (space, tab or newline)
- \w means a 'word' character, [A-za-z0-9_]
- . means (almost) anything at all

Quantifiers

- Numbers inside {} curly brackets mean: match one or more repetitions of whatever came **immediately** before
  - e.g. \d{16}
    matches a credit card number (without spaces)
  - e.g. NO{1,4}!
    matches NO!, NOO!, NOOO! and NOOOO!
  - e.g. \s{0,}
    matches any amount of space (including none)

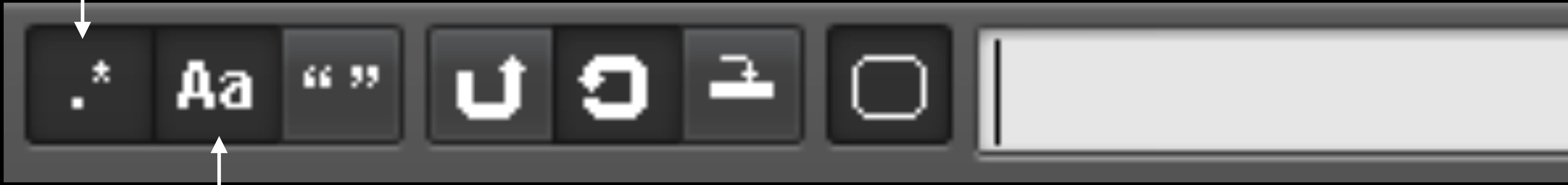
Quantifier shortcuts

- ? means none or one, {0,1}
  - e.g. expressions? matches expression and expressions
- * means zero or more, {0,}
  - e.g. 10*1 matches 11, 101, 1001, 10001, 100001, …
- + means one or more, {1,}
  - e.g. \w+ matches one whole word

How to find a regex

- Sublime Text: *Find > Find …* or Ctrl-F or ⌘F

Use Regular Expressions



Match case (so A is different from a)

Let’s actually do it!

- We’ll …
  - Download *The Adventures of Sherlock Holmes*
  - Open in (or Copy+Paste into) Sublime Text
  - Find street addresses using Regular Expressions

Hints

- Use your cheat sheet:
  character classes and quantifiers

- There are at least 7 addresses to find
  — including 221B, Baker Street

- How would you express the pattern you're looking for in English?

  - Some digits
    Maybe a letter
    Maybe a comma
    A space (or new line)
    A capital letter
    Some more letters

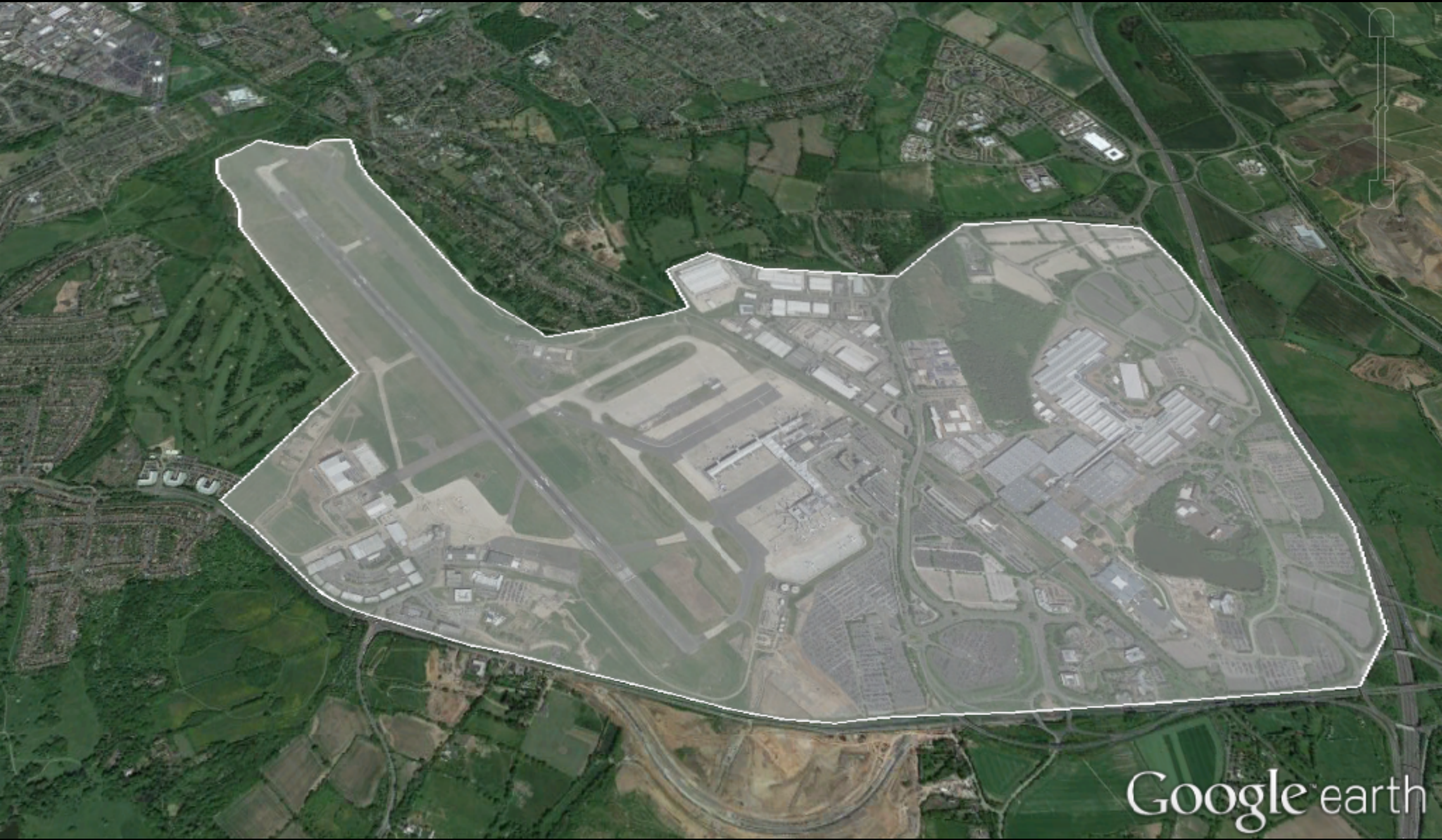
Solution

- `[0-9]+[A-Za-z]?,?\s+[A-Z][a-z]+`

- 7 Pope's Court, Fleet Street
  17 King Edward Street
  31 Lyon Place
  221B, Baker Street
  117, Brixton Road
  16A, Victoria Street
  226 Gordon Square

Rearrange data fields

- Airport polygons in a KML file to be inserted into a database as WKT


Google earth

Let’s do it!

- We’ll:
    - Download a KML file of Birmingham Airport
    - Open it in Sublime Text
    - Convert the coordinates to WKT format to insert in my database
- But first …

Groups

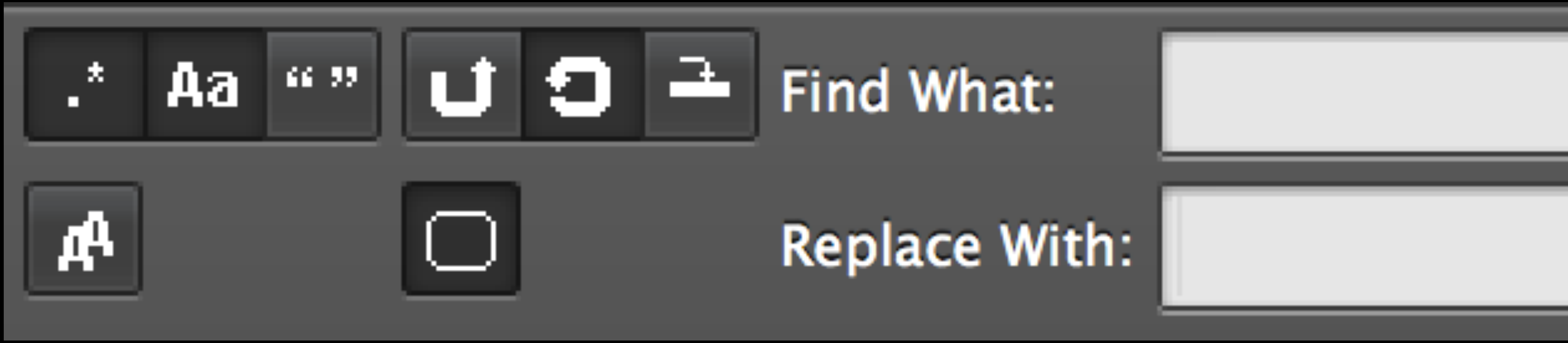
- Round brackets () define groups, and these have several uses
  - Quantifying **sequences**
    - e.g. (in)?flammable matches the synonyms flammable and inflammable
  - Specifying **alternatives** with | (= or)
    - e.g. \d+(st|nd|rd|th) matches 1st, 2nd, 33rd, 404th, … (also 1nd, 2rd, 3th, 4st, … but never mind)
  - And …

Capture groups

- Bracketed groups can be referenced as $*n* in your replacement text: $1 is the first group, $2 the second, … (while $0 is the whole match)
  - e.g. 19(\d0)s ➛ $1s replaces 1960s ➛ 60s, 1980s ➛ 80s, etc.
  - e.g. (March|April|May) (\d\d?), (\d{4})
    ➛ $2 $1 $3
    replaces May 4, 2014 ➛ 4 May 2014, etc.

How to replace a regex

- Sublime Text: *Find > Replace …* or Ctrl-H or ⌥⌘F



KML ➝ WKT

- Convert KML coordinates:
  - *longitude*,*latitude*,*alt longitude*,*latitude*,*alt longitude*,*latitude*,*alt* …
  - -1.760685207991166,52.45120844576951,0
    -1.759049981971205,52.45020460522435,0
    -1.756435134030001,52.44848442920233,0
- To WKT coordinates:
  - *longitude latitude*,*longitude latitude,longitude latitude*, …
  - *-1.760685207991166 52.45120844576951,-1.759049981971205 52.45020460522435,-1.756435134030001 52.44848442920233*

Solution

- ([-0-9.]+),([-0-9.]+),0 ➛ $2 $1,

My CSV data is broken

```
16059,ABBEY ST BATHANS NO 2                   ,RAIN,922957  ,CARLOS  ,1961-01-01 00:00,1963-12-3
16059,ABBEY ST BATHANS NO 2                   ,RAIN,922957  ,CLMSN   ,1961-01-01 00:00,1990-12-3
16059,ABBEY ST BATHANS NO 2                   ,RAIN,922957  ,WADRAIN ,1961-01-01 00:00,1963-12-3
10493,ABBEYCWMHIR, FORESTERS HOUSE NO 1       ,RAIN,466587  ,CLMSN   ,1961-01-01 00:00,1990-12-3
10493,ABBEYCWMHIR, FORESTERS HOUSE NO 1       ,RAIN,466586  ,CLMSN   ,1961-01-01 00:00,1990-12-3
10493,ABBEYCWMHIR, FORESTERS HOUSE NO 1       ,RAIN,466586  ,CARLOS  ,1961-01-01 00:00,1984-12-3
10493,ABBEYCWMHIR, FORESTERS HOUSE NO 1       ,RAIN,466587  ,WADRAIN ,1965-02-01 00:00,1966-01-3
10493,ABBEYCWMHIR, FORESTERS HOUSE NO 1       ,RAIN,466586  ,WADRAIN ,1965-02-01 00:00,1984-12-3
10493,ABBEYCWMHIR, FORESTERS HOUSE NO 1       ,RAIN,466587  ,WADRAIN ,1961-01-01 00:00,1965-01-3
10494,ABBEYCWMHIR, HALL                       ,RAIN,466615  ,CLMSN   ,1961-01-01 00:00,1990-12-3
12489,ABBEYSTEAD GARDENS                      ,RAIN,577794  ,WADRAIN ,1998-01-01 00:00,3999-12-3
12489,ABBEYSTEAD GARDENS                      ,RAIN,577793  ,WADRAIN ,1998-01-01 00:00,2009-05-3
12489,ABBEYSTEAD GARDENS                      ,RAIN,577793  ,WAMRAIN ,2009-06-01 00:00,3999-12-3
12489,ABBEYSTEAD GARDENS                      ,RAIN,577793  ,CARLOS  ,1961-01-01 00:00,3999-12-3
12489,ABBEYSTEAD GARDENS                      ,RAIN,577793  ,WADRAIN ,1961-01-01 00:00,1997-12-3
12489,ABBEYSTEAD GARDENS                      ,RAIN,577794  ,WADRAIN ,1991-01-01 00:00,1997-12-3
12489,ABBEYSTEAD GARDENS                      ,RAIN,577793  ,CLMSN   ,1961-01-01 00:00,1990-12-3
12488,ABBEYSTEAD RESR                         ,RAIN,577724  ,WADRAIN ,1961-01-01 00:00,1966-12-3
12488,ABBEYSTEAD RESR                         ,RAIN,577724  ,CLMSN   ,1961-01-01 00:00,1990-12-3
12488,ABBEYSTEAD RESR                         ,RAIN,577724  ,CARLOS  ,1961-01-01 00:00,1966-12-3
12491,ABBEYSTEAD RESR NO 2                    ,RAIN,577803  ,CLMSN   ,1961-01-01 00:00,1990-12-3
12491,ABBEYSTEAD RESR NO 2                    ,RAIN,577803  ,CARLOS  ,1980-01-01 00:00,3999-12-3
12491,ABBEYSTEAD RESR NO 2                    ,RAIN,577805  ,WADRAIN ,1992-01-01 00:00,3999-12-3
```

Let’s fix it!

- We’ll:
  - Download an extract of this CSV-ish file
  - Open it in Sublime Text
  - Make it valid CSV data
    (at least 2 approaches are possible)
- But first …

Anchors

- ^ matches the start of a line and $ matches the end of a line
  - e.g. ^\d+$ matches any integer, but **only** if it's the only thing on a line
- \b matches a word boundary
  - e.g. ing\b matches going but not ingot

Negation

- Inside a character class, ^ means **not**
  - e.g. [^,.] matches any single character **except** a comma or a full stop
- Capitalised shortcuts **reverse** their meanings
  - \D means a **non-**digit
    \S means **non-**whitespace
    \W means a **non-**word character
    \B means **not** a word boundary
  - e.g. ing\B matches ingot but not going

Greediness

- By default, regex quantifiers are **greedy**: they match the **longest** sequence possible
  - e.g. for the text 1,2,3,4,5 ,.*, matches ,2,3,4,
- If that's not what you want, there are two options:
  - Use an extra ? to specify non-greediness, and match the **shortest** sequence possible (giving ??, *?, +? and {}?) — e.g. ,.*?, matches ,2,
  - Be clearer about what you want to match — e.g. ,[^,]*, also matches ,2,

Can you fix this CSV?

```
16059,ABBEY ST BATHANS NO 2                   ,RAIN,922957  ,CARLOS  ,1961-01-01 00:00,1963-12-3
16059,ABBEY ST BATHANS NO 2                   ,RAIN,922957  ,CLMSN   ,1961-01-01 00:00,1990-12-3
16059,ABBEY ST BATHANS NO 2                   ,RAIN,922957  ,WADRAIN ,1961-01-01 00:00,1963-12-3
10493,ABBEYCWMHIR, FORESTERS HOUSE NO 1       ,RAIN,466587  ,CLMSN   ,1961-01-01 00:00,1990-12-3
10493,ABBEYCWMHIR, FORESTERS HOUSE NO 1       ,RAIN,466586  ,CLMSN   ,1961-01-01 00:00,1990-12-3
10493,ABBEYCWMHIR, FORESTERS HOUSE NO 1       ,RAIN,466586  ,CARLOS  ,1961-01-01 00:00,1984-12-3
10493,ABBEYCWMHIR, FORESTERS HOUSE NO 1       ,RAIN,466587  ,WADRAIN ,1965-02-01 00:00,1966-01-3
10493,ABBEYCWMHIR, FORESTERS HOUSE NO 1       ,RAIN,466586  ,WADRAIN ,1965-02-01 00:00,1984-12-3
10493,ABBEYCWMHIR, FORESTERS HOUSE NO 1       ,RAIN,466587  ,WADRAIN ,1961-01-01 00:00,1965-01-3
10494,ABBEYCWMHIR, HALL                       ,RAIN,466615  ,CLMSN   ,1961-01-01 00:00,1990-12-3
12489,ABBEYSTEAD GARDENS                      ,RAIN,577794  ,WADRAIN ,1998-01-01 00:00,3999-12-3
12489,ABBEYSTEAD GARDENS                      ,RAIN,577793  ,WADRAIN ,1998-01-01 00:00,2009-05-3
12489,ABBEYSTEAD GARDENS                      ,RAIN,577793  ,WAMRAIN ,2009-06-01 00:00,3999-12-3
12489,ABBEYSTEAD GARDENS                      ,RAIN,577793  ,CARLOS  ,1961-01-01 00:00,3999-12-3
12489,ABBEYSTEAD GARDENS                      ,RAIN,577793  ,WADRAIN ,1961-01-01 00:00,1997-12-3
12489,ABBEYSTEAD GARDENS                      ,RAIN,577794  ,WADRAIN ,1991-01-01 00:00,1997-12-3
12489,ABBEYSTEAD GARDENS                      ,RAIN,577793  ,CLMSN   ,1961-01-01 00:00,1990-12-3
12488,ABBEYSTEAD RESR                         ,RAIN,577724  ,WADRAIN ,1961-01-01 00:00,1966-12-3
12488,ABBEYSTEAD RESR                         ,RAIN,577724  ,CLMSN   ,1961-01-01 00:00,1990-12-3
12488,ABBEYSTEAD RESR                         ,RAIN,577724  ,CARLOS  ,1961-01-01 00:00,1966-12-3
12491,ABBEYSTEAD RESR NO 2                    ,RAIN,577803  ,CLMSN   ,1961-01-01 00:00,1990-12-3
12491,ABBEYSTEAD RESR NO 2                    ,RAIN,577803  ,CARLOS  ,1980-01-01 00:00,3999-12-3
12491,ABBEYSTEAD RESR NO 2                    ,RAIN,577805  ,WADRAIN ,1992-01-01 00:00,3999-12-3
```

Solutions

- We can use the fact that this file has fixed-width columns:
  - `^([^,]*,)(.{40})` ➛ `$1"$2"`
- But what if it didn’t? Well, since only one field has extra commas, we can count the commas before and after:
  - `^([^,]*,)(.*)((,[^,]*){23})$` ➛ `$1"$2"$3`

Proofreading

- Repeated words are a common problem, especially when the words are short and the the repetitions span a line break

Let’s do it!

- Embarrassingly, my final, corrected, submitted thesis has at least 4 errors of this sort
- Let’s find them!
- But first …

Backreferences

- We saw earlier that we can use the text matched by a capture group in our replacement expression — as $1, $2, …
- But we can also use capture group text in our search expression — as \1, \2, …
  - e.g. \b(\w)(\w)(\w)\w?\3\2\1\b matches palindromes of 6 or 7 letters — e.g. redder, rotator, …

Escaping with \

- Numbers and letters on their own are **always** literals — putting a \ in front **may** give them a special meaning
- **Some** other characters on their own have a special meaning — putting a \ in front **always** makes them literals
    - When in doubt, use a \ (e.g. , is actually a literal, but \, works just the same)

Let’s actually do it!

- Embarrassingly, my final, corrected, submitted thesis has at least 4 errors of this sort
- Let’s find them!

Solution

- \b(\w+) \1\b

- have have
  the the
  an an
  the the

Where else can I use RegExes?

- Stata (regexm, regexr, regexs)
- R (grep, gsub, …), SPSS, Matlab, …
- Terminal/Command Prompt
  - grep, awk, sed, …
- Python, Ruby, Perl, JavaScript, Java, …

Problem 3

I need to match some things in the presence of errors or inconsistencies

- Firms, football teams, regions, …
- Two approaches
  - Trigrams
  - Levenshtein distance

Football teams

- Two data sets
  - Stadium name, team name, latitude, longitude
  - Home team name, away team name, date, result

But …

- Hayes & Yeading — Hayes and Yeading United
- Wolverhampton Wanderers — Wolves
- Queens Park Rangers — QPR
- Cowdenbeath — Coedenbeath
- …

Trigrams

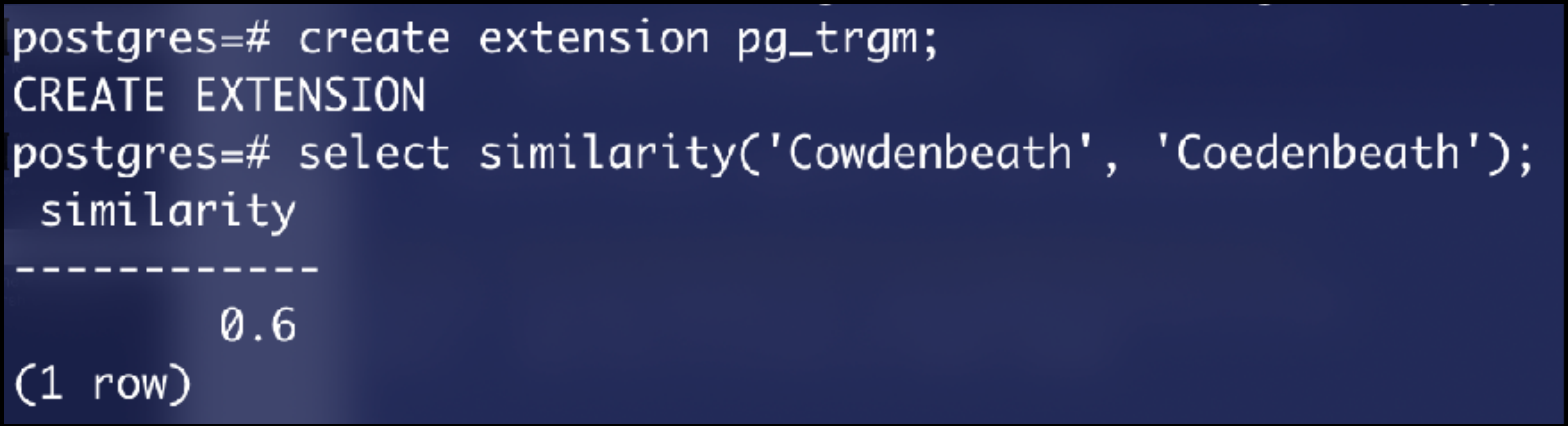
- A trigram is a group of three consecutive characters taken from a string.
- We can measure the similarity of two strings by counting the number of trigrams they share.
- This simple idea turns out to be very effective for measuring the similarity of words in many natural languages.
- Note: A string is considered to have two spaces prefixed and one space suffixed when determining the set of trigrams contained in the string.

— http://www.postgresql.org/docs/9.1/static/pgtrgm.html

Cowdenbeath vs Coedenbeath

- ··Cowdenbeath·
    - ··C, ·Co, Cow, owd, wde, den, enb, nbe, bea, eat, ath, th·
- ··Coedenbeath·
    - ··C, ·Co, Coe, oed, ede, den, enb, nbe, bea, eat, ath, th·

Calculating similarity



```
postgres=# create extension pg_trgm;
CREATE EXTENSION
postgres=# select similarity('Cowdenbeath', 'Coedenbeath');
 similarity
------------
        0.6
(1 row)
```

- Similarity
  = common / (total – common)
  = 9 / (12 + 12 – 9)
  = 9 / 15
  = 0.6

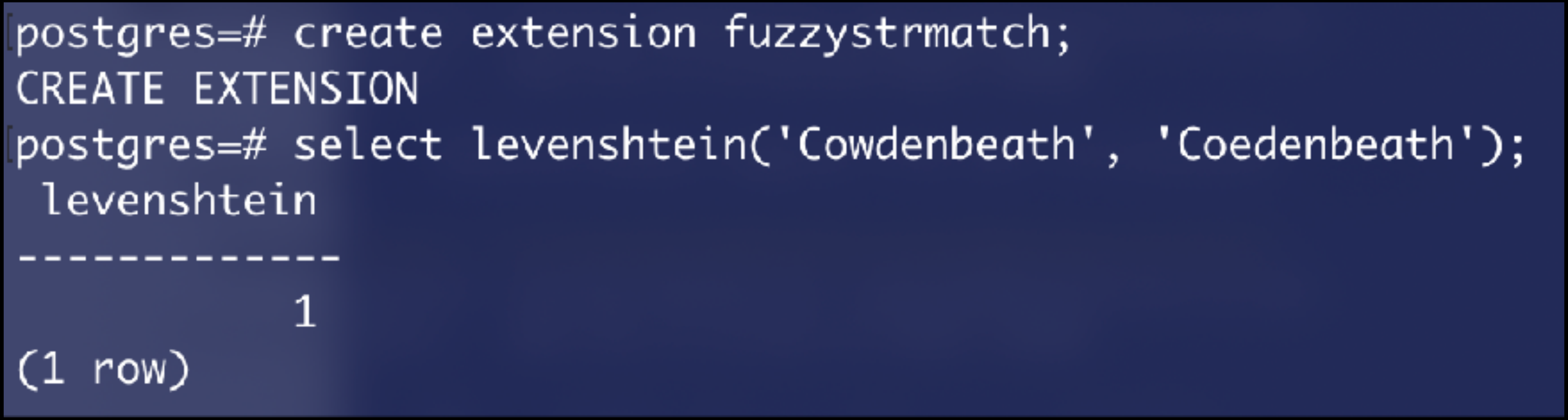
Where can I use trigrams?

- PostgreSQL
- R
- … ?

Levenshtein distance

- The minimum number of single-character edits (insertions, deletions or substitutions) required to change one word into the other
- Also known as: edit distance

Cowdenbeath vs Coedenbeath



```
postgres=# create extension fuzzystrmatch;
CREATE EXTENSION
postgres=# select levenshtein('Cowdenbeath', 'Coedenbeath');
 levenshtein
-------------
           1
(1 row)
```

- You can get from Cowdenbeath to Coedenbeath with one substitution, so the distance is 1

Where can I use Levenshtein distance?

- PostgreSQL
- R
- Stata
- …

Bonus text problems

- **I need to extract some data from XML/HTML**
  - Use XPath or CSS selectors
- **I need to see the differences between big text file A and big text file B**
  - Use diff or FileMerge
- **I need to keep track of a large set of files that I work on together**
  - Use version control: git

Web scraping

- You’ll very likely need regular expressions
  — but *don’t* try to use regular expressions alone!

Questions?