UNIVERSITY OF SUSSEX

Regular Expressions and inexact matching

George MacKerron

iPhD 860L1 Programming — 30 September 2024

Resources

Canvas 860L1

or: users.sussex.ac.uk/~gm268/iphd/regexps/

Today's problems

1. **I need to find, replace or extract some things that follow a pattern**
    - Postcodes, email addresses, identifiers, dates, …
2. **I need to match some things in the presence of errors or inconsistencies**
    - Names or identifiers for countries, firms, teams, products, …

Today’s solutions

1. Regular Expressions
2. Trigrams, Levenshtein distance

Is this useful?

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

Today’s key tool

- **Sublime Text**
  - A good text editor for Windows, Mac & Linux
  - Free indefinitely ‘for evaluation’
- Download from **sublimetext.com/download**
  - Trouble with Windows? Try the portable version

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

- Find names or addresses
- Rearrange some data fields
- Fix a broken CSV file
- Find hard-to-spot errors in a PhD 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

xkcd.com/208

Task 1

- 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 **character ranges** inside [] square brackets match any **one** of those characters
  - e.g. [hnc]ow matches how, now and cow (and only those: *not* hncow, for example)
  - e.g. [1-4] matches 1, 2, 3 and 4
  - e.g. [A-Za-z] matches any single letter of the ordinary Roman alphabet, upper- or lower-case

Classes: shortcuts

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

Quantifiers

- Numbers inside {*n*, *m*} curly brackets mean: match between *n* and *m* repetitions of whatever went **immediately** before
  - e.g. \d{16}
    matches a credit card number (with no spaces)
  - e.g. No{1,4}!
    matches No!, Noo!, Nooo! and Noooo!
  - e.g. \s{0,}
    matches any amount of space (including none)

Quantifiers: 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

RegExp cheat sheet

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 letter of the alphabet, upper- or lower-case

Shortcuts

\d means a digit, **[0-9]**

e.g. **\d\d** matches 00 – 99

\s means a whitespace character, **[\t\r\n]** (space, tab, newline)

\w means a 'word' character, **[A-za-z0-9_]**

. means anything at all (except possibly a newline)

Quantifiers

Numbers inside curly brackets **{}** mean: match one or more repetitions of whatever came *immediately* before.

{*number*} means exactly *number*
{*min*, *max*} means a range between *min* and *max* inclusive
{*min*,} means at least *min*

e.g. **\d{16}** matches a credit card number (no spaces)
e.g. **ba{1,3}d** matches bad, baad and baaad
e.g. **\s{1,}** matches any amount of white space

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 a whole word

Groups

Round brackets **()** define groups, and these have several uses:

Quantified sequences

e.g. **(in)?flammable** matches flammable, inflammable

Alternatives with **|**

e.g. **\d+(st|nd|rd|th)** matches 1st, 2nd, 33rd, 404th, …

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.

Anchors

^ matches the start of a line
$ 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 within 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 have reversed meanings:

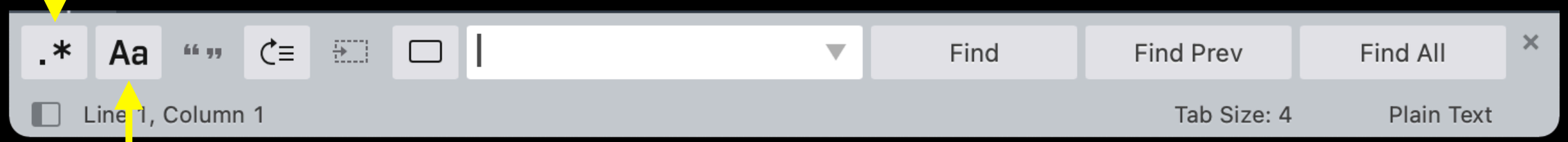
\D means a non-digit
\S means non-whitespace
\W means a non-word character
\B means not a word boundary

e.g. **ing\B** matches within ingot but not going

Finding regex matches

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

Use Regular Expressions



Case sensitivity
(so A is different from a)

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

Hints

- Use your cheat sheet:
  character classes and quantifiers
- There are at least 7 addresses to find,
  including Holmes' home address: 221B, Baker Street
- How would you express the pattern you're looking for in English?
  - Some numeric digits
    Maybe a letter
    Maybe a comma
    A space (or new line)
    A capital letter
    Some more letters

A possible 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

Transportation Research Part D 56 (2017) 43–54

Contents lists available at ScienceDirect

Transportation Research Part D

journal homepage: www.elsevier.com/locate/trd

Experience sampling in and around airports. Momentary subjective wellbeing, airports, and aviation noise in England

Daniel Fujiwara[a,b], Ricky N. Lawton[b,*], George MacKerron[c]

[a] London School of Economics and Political Science, Centre for Economic Performance, Houghton Street, London WC2A 2AE, United Kingdom
[b] Simetrica Research Consultancy, The Shepherds Building (West Entrance), Rockley Road, Shepherds Bush, London, W14 0DA, United Kingdom
[c] Department of Economics, University of Sussex, Jubilee Building, Falmer, Brighton BN1 9SL, United Kingdom

ARTICLE INFO

Keywords:
Aircraft noise
Subjective wellbeing
Airport location
Quality of life
Transport policy
Experience Sampling Method

ABSTRACT

We explore the wellbeing of people in and around English airports using real-time data from a large spatial positioning experience sampling dataset (Mappiness). We analyze the association between subjective wellbeing reported in the moment and aviation, in terms of airport location, aircraft noise, and activities within airports. This is the first time that a large Experience Sample Method (ESM) of momentary wellbeing measurements has been used to quantify the associations between aviation and subjective wellbeing. Being within areas of high levels of aircraft noise is associated with lower levels of happiness and relaxation. Those surveyed in proximity to airports report significantly lower levels of relaxation. These findings have important implications to policy. Exploiting the panel nature of the ESM data provides the strongest causal claims to date of the negative association between aviation activities and subjective wellbeing. The Mappiness application also allows us to assess the association between airports and wellbeing on those inside them, and divide activities within airports between those who work there and those who are passing through for travel purposes, as well as the effects of aircraft noise beyond airports. This gives us a broader insight into the range of impacts, both positive and negative, that aviation has on peoples' momentary wellbeing, which may be used to inform aviation noise mitigation and compensation policies in the future.

1. Introduction

Aviation provides a range of economic and social benefits, ranging from jobs and employment at the local and national level, to the individual leisure benefits of foreign holidays (Airports Commission, 2013; Caves, 2003; Reynolds et al., 2007). However,

Task 2

- We'll:
  - Download a KML (Google Earth) file of Birmingham Airport
  - Open it in Sublime Text
  - Convert the coordinates to WKT format to insert into a database
- **But first ...**

Groups

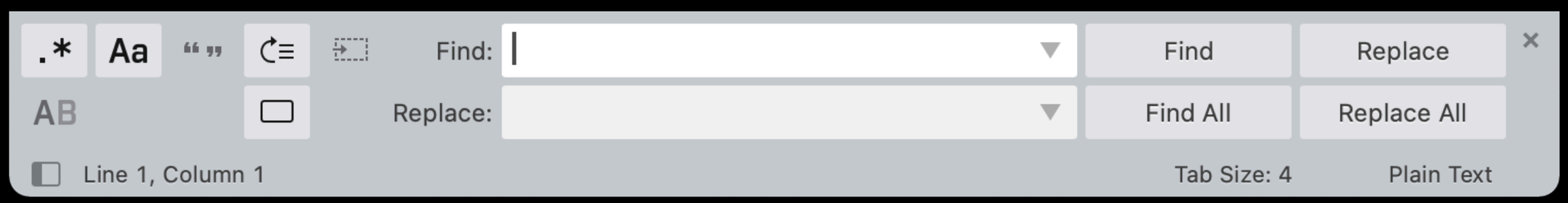
- Round brackets () define **groups**, and these have several uses:
  - Quantifying **sequences**
    - e.g. (in)?flammable matches the synonyms flammable and inflammable
  - Specifying **alternatives** with | (= or)
    - e.g. \d+(st|nd|rd|th) matches 1st, 2nd, 33rd, 404th, … (also 1nd, 2rd, 3th, 4st, … 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(\d0s) ➛ $1 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.

Replacing regex matches

- 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*

BHX.kml

BHX.kml

```
-1.760685207991166,52.45120844576951,0 -1.759049981971205,52.45020460522435,0
-1.756435134030001,52.44848442920233,0 -1.755041775692042,52.44776971709988,0
-1.755842316554083,52.44812684544019,0 -1.754514221132745,52.44759491498504,0
-1.753721193609166,52.44737030207365,0 -1.751309188149953,52.44684504829074,0
-1.74925363898757,52.44637794439268,0 -1.747163617076442,52.44606763283707,0
-1.744822823196203,52.44566456237806,0 -1.742224714268884,52.44523564300942,0
-1.738316260601113,52.44455152680645,0 -1.736158681285493,52.44425270987185,0
-1.734067078738503,52.44411050402036,0 -1.732048741097024,52.44427134594171,0
-1.726163918135212,52.44445403801316,0 -1.71981068111229,52.44473808748857,0
-1.711764963325122,52.44520118649855,0 -1.709567106891852,52.44684905133793,0
-1.70881819490036,52.45045420692982,0 -1.709763840813225,52.45370034046023,0
-1.710315977363456,52.45687657765354,0 -1.711769693074394,52.4608244123567,0
-1.712938213678402,52.46255750500632,0 -1.718391898656491,52.46296331700866,0
-1.722584468392754,52.46270458344581,0 -1.725129644190383,52.46127930866926,0
-1.726837966624156,52.46027289203563,0 -1.728333785287259,52.46034368696634,0
-1.732019706965475,52.46057486874325,0 -1.733716291262516,52.46063444360617,0
-1.736676821772629,52.46139303068992,0 -1.738921823365885,52.46061901889922,0
-1.738017755867781,52.45893818233888,0 -1.738787593571756,52.45873865732324,0
-1.741445462690799,52.45816046660566,0 -1.745315771902287,52.45759997816537,0
-1.748969869707926,52.45958954835731,0 -1.752656038250821,52.46308707571266,0
-1.754778320043315,52.46497573608866,0 -1.755566901444544,52.46585758417478,0
-1.75615679194606,52.46716978776546,0 -1.757832109296317,52.46713378851103,0
-1.758601113913711,52.46673184758529,0 -1.759866671120662,52.46684245519602,0
-1.761501617261514,52.46714933357994,0 -1.762798473833485,52.46646595369114,0
-1.763682903963597,52.46615061955627,0 -1.763409008653295,52.46514848759426,0
-1.762401415907416,52.46430863209218,0 -1.761967623848103,52.4635749494212,0
-1.761728756075182,52.4626018614508,0 -1.754822619009221,52.45609619487535,0
-1.756535797250171,52.45470370233204,0 -1.756725617703356,52.45461907894269,0
-1.760685207991166,52.45120844576951,0
```

.* Aa “” AB Find: Replace: Find Find All Replace Replace All

Line 1, Column 1 Tab Size: 4 XML

Solution

- ([-0-9.]+),([-0-9.]+),0 ? ➛ $1 $2,

My CSV data is broken

```
16059,ABBEY ST BATHANS NO 2                ,RAIN,922957  ,CARLOS  ,1961-01-01 00:00,1963-12-31
16059,ABBEY ST BATHANS NO 2                ,RAIN,922957  ,CLMSN   ,1961-01-01 00:00,1990-12-31
16059,ABBEY ST BATHANS NO 2                ,RAIN,922957  ,WADRAIN ,1961-01-01 00:00,1963-12-31
10493,ABBEYCWMHIR, FORESTERS HOUSE NO 1    ,RAIN,466587  ,CLMSN   ,1961-01-01 00:00,1990-12-31
10493,ABBEYCWMHIR, FORESTERS HOUSE NO 1    ,RAIN,466586  ,CLMSN   ,1961-01-01 00:00,1990-12-31
10493,ABBEYCWMHIR, FORESTERS HOUSE NO 1    ,RAIN,466586  ,CARLOS  ,1961-01-01 00:00,1984-12-31
10493,ABBEYCWMHIR, FORESTERS HOUSE NO 1    ,RAIN,466587  ,WADRAIN ,1965-02-01 00:00,1966-01-31
10493,ABBEYCWMHIR, FORESTERS HOUSE NO 1    ,RAIN,466586  ,WADRAIN ,1965-02-01 00:00,1984-12-31
10493,ABBEYCWMHIR, FORESTERS HOUSE NO 1    ,RAIN,466587  ,WADRAIN ,1961-01-01 00:00,1965-01-31
10494,ABBEYCWMHIR, HALL                    ,RAIN,466615  ,CLMSN   ,1961-01-01 00:00,1990-12-31
12489,ABBEYSTEAD GARDENS                   ,RAIN,577794  ,WADRAIN ,1998-01-01 00:00,3999-12-31
12489,ABBEYSTEAD GARDENS                   ,RAIN,577793  ,WADRAIN ,1998-01-01 00:00,2009-05-31
12489,ABBEYSTEAD GARDENS                   ,RAIN,577793  ,WAMRAIN ,2009-06-01 00:00,3999-12-31
12489,ABBEYSTEAD GARDENS                   ,RAIN,577793  ,CARLOS  ,1961-01-01 00:00,3999-12-31
12489,ABBEYSTEAD GARDENS                   ,RAIN,577793  ,WADRAIN ,1961-01-01 00:00,1997-12-31
12489,ABBEYSTEAD GARDENS                   ,RAIN,577794  ,WADRAIN ,1991-01-01 00:00,1997-12-31
12489,ABBEYSTEAD GARDENS                   ,RAIN,577793  ,CLMSN   ,1961-01-01 00:00,1990-12-31
12488,ABBEYSTEAD RESR                      ,RAIN,577724  ,WADRAIN ,1961-01-01 00:00,1966-12-31
12488,ABBEYSTEAD RESR                      ,RAIN,577724  ,CLMSN   ,1961-01-01 00:00,1990-12-31
12488,ABBEYSTEAD RESR                      ,RAIN,577724  ,CARLOS  ,1961-01-01 00:00,1966-12-31
12491,ABBEYSTEAD RESR NO 2                 ,RAIN,577803  ,CLMSN   ,1961-01-01 00:00,1990-12-31
12491,ABBEYSTEAD RESR NO 2                 ,RAIN,577803  ,CARLOS  ,1980-01-01 00:00,3999-12-31
12491,ABBEYSTEAD RESR NO 2                 ,RAIN,577805  ,WADRAIN ,1992-01-01 00:00,3999-12-31
```

Task 3

- 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, **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

- ^ *at the start of a character class* means: **not** any of these
  - e.g. [^,.] matches any single character **except** a comma or full stop
- Capitalised shortcuts **negate** or reverse their meanings
  - \D means any **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,
- Not what you want? Then there are two options:
  - Use an extra ? to specify non-greedy matching for the **shortest** sequence possible (giving ??, *?, +? and {}?) — e.g. ,.*?, matches ,2,
  - Be more explicit 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 00:00,1963-12-31
16059,ABBEY ST BATHANS NO 2                ,RAIN,922957  ,CLMSN   ,1961-01-01 00:00,1990-12-31
16059,ABBEY ST BATHANS NO 2                ,RAIN,922957  ,WADRAIN ,1961-01-01 00:00,1963-12-31
10493,ABBEYCWMHIR, FORESTERS HOUSE NO 1    ,RAIN,466587  ,CLMSN   ,1961-01-01 00:00,1990-12-31
10493,ABBEYCWMHIR, FORESTERS HOUSE NO 1    ,RAIN,466586  ,CLMSN   ,1961-01-01 00:00,1990-12-31
10493,ABBEYCWMHIR, FORESTERS HOUSE NO 1    ,RAIN,466586  ,CARLOS  ,1961-01-01 00:00,1984-12-31
10493,ABBEYCWMHIR, FORESTERS HOUSE NO 1    ,RAIN,466587  ,WADRAIN ,1965-02-01 00:00,1966-01-31
10493,ABBEYCWMHIR, FORESTERS HOUSE NO 1    ,RAIN,466586  ,WADRAIN ,1965-02-01 00:00,1984-12-31
10493,ABBEYCWMHIR, FORESTERS HOUSE NO 1    ,RAIN,466587  ,WADRAIN ,1961-01-01 00:00,1965-01-31
10494,ABBEYCWMHIR, HALL                    ,RAIN,466615  ,CLMSN   ,1961-01-01 00:00,1990-12-31
12489,ABBEYSTEAD GARDENS                   ,RAIN,577794  ,WADRAIN ,1998-01-01 00:00,3999-12-31
12489,ABBEYSTEAD GARDENS                   ,RAIN,577793  ,WADRAIN ,1998-01-01 00:00,2009-05-31
12489,ABBEYSTEAD GARDENS                   ,RAIN,577793  ,WAMRAIN ,2009-06-01 00:00,3999-12-31
12489,ABBEYSTEAD GARDENS                   ,RAIN,577793  ,CARLOS  ,1961-01-01 00:00,3999-12-31
12489,ABBEYSTEAD GARDENS                   ,RAIN,577793  ,WADRAIN ,1961-01-01 00:00,1997-12-31
12489,ABBEYSTEAD GARDENS                   ,RAIN,577794  ,WADRAIN ,1991-01-01 00:00,1997-12-31
12489,ABBEYSTEAD GARDENS                   ,RAIN,577793  ,CLMSN   ,1961-01-01 00:00,1990-12-31
12488,ABBEYSTEAD RESR                      ,RAIN,577724  ,WADRAIN ,1961-01-01 00:00,1966-12-31
12488,ABBEYSTEAD RESR                      ,RAIN,577724  ,CLMSN   ,1961-01-01 00:00,1990-12-31
12488,ABBEYSTEAD RESR                      ,RAIN,577724  ,CARLOS  ,1961-01-01 00:00,1966-12-31
12491,ABBEYSTEAD RESR NO 2                 ,RAIN,577803  ,CLMSN   ,1961-01-01 00:00,1990-12-31
12491,ABBEYSTEAD RESR NO 2                 ,RAIN,577803  ,CARLOS  ,1980-01-01 00:00,3999-12-31
12491,ABBEYSTEAD RESR NO 2                 ,RAIN,577805  ,WADRAIN ,1992-01-01 00:00,3999-12-31
```

Two solutions: hints

- Easy mode
  - Make use of the fixed-width columns
  - Count characters
- Hard mode
  - What if it the column widths **weren't** fixed?
  - Count the commas before and after

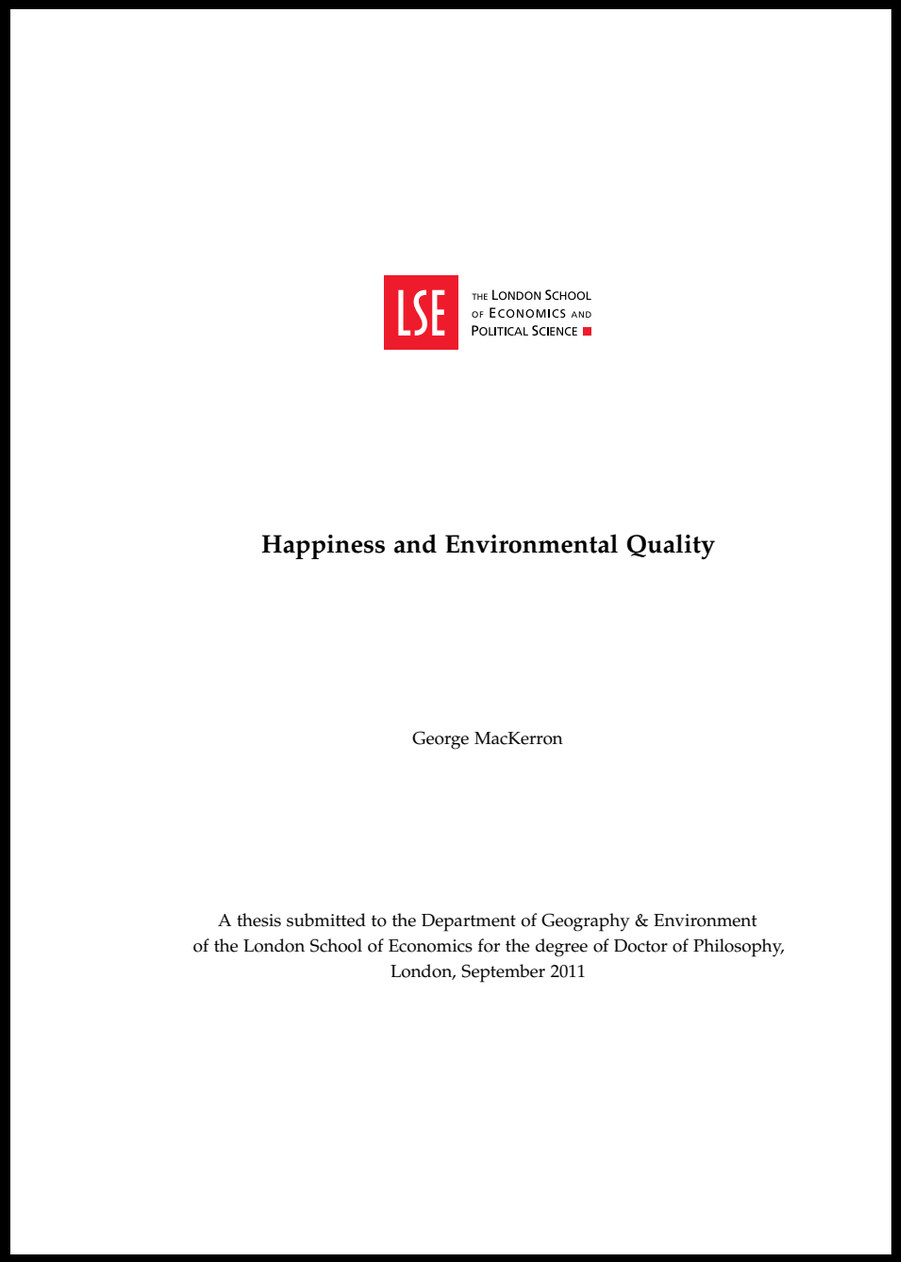
Solutions

- 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

Task 4



LSE THE LONDON SCHOOL OF ECONOMICS AND POLITICAL SCIENCE

Happiness and Environmental Quality

George MacKerron

A thesis submitted to the Department of Geography & Environment
of the London School of Economics for the degree of Doctor of Philosophy,
London, September 2011

- My final, corrected, submitted thesis has at least **4** errors of this sort
- Let's find them!
- **But first …**

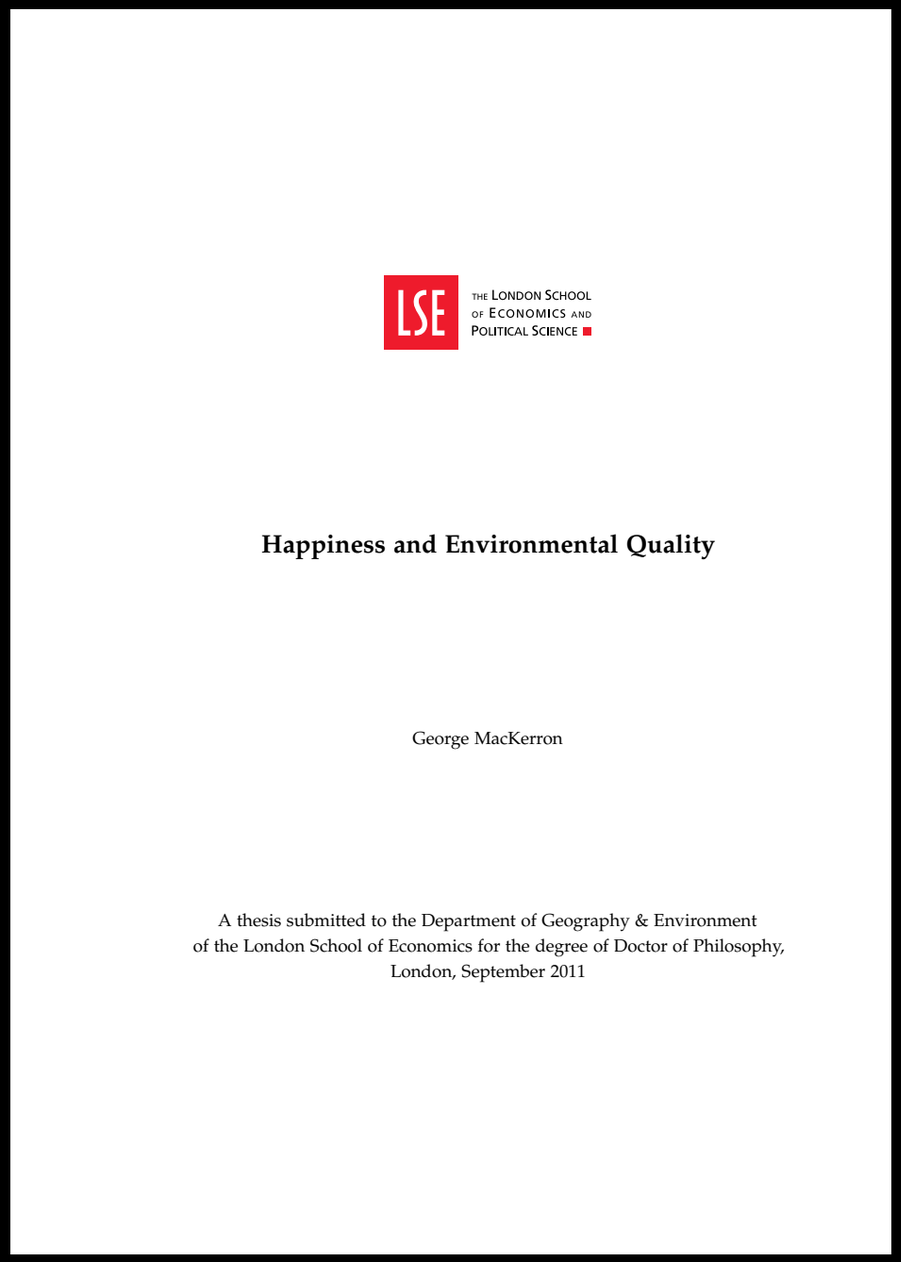
Backreferences

- 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 **later** in the same search expression — as \1, \2, …
  - e.g. \b(\w)(\w?)(\w?)\w?\3\2\1\b matches **palindromes** of 2 – 7 letters
    — e.g. cc, gig, kook, minim, redder, rotator, …

Escaping with \

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

Let’s do it!



LSE THE LONDON SCHOOL OF ECONOMICS AND POLITICAL SCIENCE

Happiness and Environmental Quality

George MacKerron

A thesis submitted to the Department of Geography & Environment
of the London School of Economics for the degree of Doctor of Philosophy,
London, September 2011

- 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 can I use RegExes?

- Stata (**`regexm`**, **`regexr`**, **`regexs`**)

```
parmest  // e.g. 2.dayofweek#7.hourofday
gen dayofweek = real(regexs(1)) if regexm(parm, "^([0-9]+)")
gen hourofday = real(regexs(1)) if regexm(parm, "#([0-9]+)")
```

- R (**`grep`**, **`gsub`**, …), SPSS, Matlab, …
- Terminal/Command Prompt
  - **`grep`**, **`awk`**, **`sed`**, …
- Your favourite programming language:
  Python, Ruby, Perl, JavaScript, Java, …

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

- **Trigrams** are groups of **three consecutive characters** taken from some text (e.g. Sussex —> Sus, uss, sse, sex)
- We can measure the similarity of two texts 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.
- Typically, the text is considered to have two spaces prefixed and one space suffixed when determining the set of trigrams it contains.
  - Why?

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?

- Stata
  - `ssc install matchit`
    `help matchit`
- 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?

- Stata
  - `ssc install strdist`
    `help strdist`
- PostgreSQL
- R
- …

Web scraping?

- You'll likely need regular expressions
  — but *don't* try to use regular expressions alone!
- There are query languages designed specifically for selecting parts of HTML documents
  - XPath or CSS selectors

Questions?