Playing With "Matches"

Using Regular Expressions in Create Lists

Richard V. Jackson
Huntington Library, Art Collections, and Botanical Gardens
San Marino, California

Why this presentation?

- Regular expressions in Create Lists are a powerful — but underutilized — tool
- User Manual (page #101608) provides little information
- Information elsewhere may be:
  - confusing or contradictory
  - not applicable to Create Lists
  - not applicable to the type of data found in Innovative records

Goals

- Introduce the construction of regular expressions in Create Lists
- Give practical examples
- Provide a sense of what situations regular expressions are best used in
- Cover some pitfalls and limits of regular expressions in Create Lists

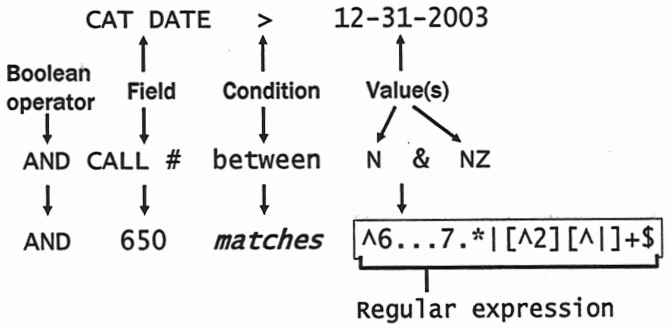
Handout — "Regular Expressions in Create Lists"

- Available now on IUG site for the conference
- Does not parallel this presentation, but does include many of the examples (plus others)
- Updated this year
- Contains:
  - Table explaining special characters (p. 1-2)
  - How data is stored in III records (p. 3)
  - Examples (p. 4-9)
  - Things that don't work (p. 10)
- PowerPoint slides will be made available later

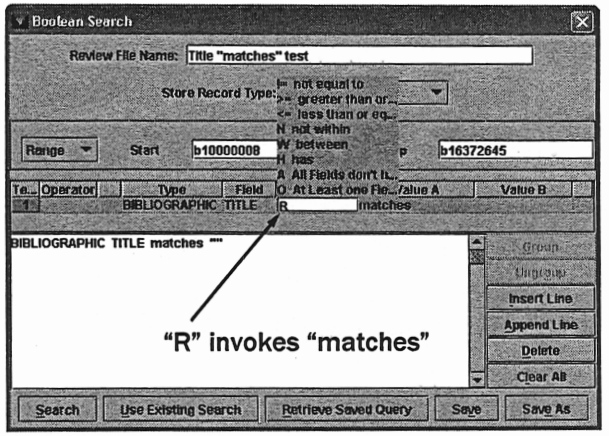
What are regular expressions?

- A powerful, sophisticated text processing tool — almost a miniature programming language
- Allow "fuzzier" matching, or finding records with particular patterns of data rather than specific values
- Widely used in many computer applications
- Somewhat limited in Create Lists, and restricted to the matching of data

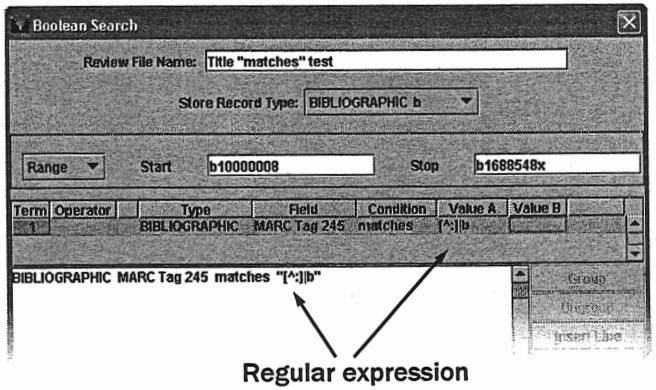
Regular expressions in Create Lists



The "matches" condition



Entering a "matches" search – Millennium



Entering a "matches" search – Innopac

```
Present search range: b1000000x to b1688546x
Find BIBLIOGRAPHIC records that satisfy the
following conditions
    245
          Enter boolean condition (=, ~,
          >, <, G, L, W, N, H, X)
```

"R" (for "matches") not shown

Entering a "matches" search – Innopac

```
Present search range: b1000000x to b1688546x
Find BIBLIOGRAPHIC records that satisfy the
following conditions
    245 matches

245 matches [^:=]|b
```

Regular expression

Conventions

In the examples that follow:

- Fields are from the Bibliographic record type, unless otherwise shown
- Regular expressions are shown in double quotes for clarity. *The quotes are not part of the expression and should not be entered.*
- The space character in regular expressions is shown as a light gray dot ("•")

Literal characters and metacharacters

- Literal characters: normal text characters that represent themselves in the match

  They include:

  A-Z a-z 0-9 <space> | most punctuation

- Metacharacters: perform some function in the regular expression

  They include:

  . [ ] + * { } ( ) ^ $ \

Literal characters

Example:

```
TITLE matches "cat"
TITLE has "cat"
```

} *same result*

Matches:

Cattle in the cold desert /|cJames A. Young ...

Intri**cat**e laughter in the satire of Swift and Pope ...

A bibliography of Austin Dobson /|**cat**tempted by Francis Edwin Murray.

Matching any character – the "dot": .

- Period (or "dot") matches any single character

➢ ***Handout page 4: example 1***

Problem: **Limit a search to titles published in the United States**

Solution:

```
COUNTRY matches "..u"
```


Matching any character – the "dot": .

Problem: **Match item locations containing characters at particular positions. (For example, say that all microform collections use a location code with 'm' in the fourth position.)**

Solution:

```
ITEM LOCATION matches "...m."
```


Character classes: [...]

- Represents any single character that is a member of the user-defined class

Example	*Matches*
[aeiou]	any of the letters *a*, *e*, *i*, *o*, or *u*
['"]	a single quote or double quote
[a-z]	any letter (upper or lower case)
[a-z0-9]	any letter or number
[14-79]	any of the numbers 1, 4, 5, 6, 7, or 9
[-•,.]	a hyphen, space, comma, or period

Character classes: [...]

➢ *Handout page 4: example 2*

Problem: **Find the phrase "gray wolf" in Titles or Notes, but include the variant spelling "grey" and the plural form "wolves"**

Solution:

```
   TITLE matches "gr[ae]y•wol[fv]"
OR NOTE matches "gr[ae]y•wol[fv]"
```

(Using the "has" condition, this search would require 8 search statements ORed together.)

Character classes: [...]

- **Remember that the character class represents only a single character in the match**

```
TITLE matches "b[aeou]t"
```

Matches	*Does NOT match*
Thomas Wolfe's al**bat**ross ...	... a**bout** ...
Seymour Lu**bet**zky : writings ...	... **beaut**iful ...
Ro**bot**s, men, and minds ...	... dou**bt**ful ...
The **but**terfly's freckled wings ...	

Negated character classes: [^...]

- **Represents any single character that is *NOT* a member of the defined class**

Example	*Matches*
[^•]	any character that is not a space
[^0-9]	any character that is not a number
[^av-z]	any character except *a*, *v*, w, *x*, *y*, or *z*

Negated character classes: [^...]

- **Negated character classes are particularly useful for finding invalid data**

➢ *Handout page 4: example 3*

Problem: **Some MARC fields have missing (or invalid) subfield codes. For example:**

```
                              h
245 10 Love for love|[microform] :|ba
       comedy /|cby William Congreve.
                                  y
650  0 United States|xHistory|Civil War,
       1861-1865.
```

Negated character classes: [^...]

Solution:

1) **Determine what subfield codes are valid for the particular MARC tag.**

   ***(E.g., for the 245 field, the valid codes are:* a, b, c, f, g, h, k, n, p, *and* s)**

2) **Construct a regular expression that matches a subfield delimiter ("|") followed by any character that is *not* one of the valid codes:**

```
245 matches "|[^abcfghknps]"
650 matches "|[^avxyz]"
```


Quantifiers: * + {*min* , *max*} {*num*}

- **Do not themselves represent any characters**
- **Apply to the preceding character (or group of characters), allowing multiple occurrences**
- **Can apply to character classes as well as literal characters**

Quantifiers – the asterisk ("star"): *

* **Preceding character(s) occur 0 or more times**

Example: `"|a0*523"`

"|a" "0", any number of times "523"

Matches:
|a523
|a0523
|a00523
[etc.]

Quantifiers – the plus: +

\+ **Preceding character(s) occur 1 or more times**

Example: `"[0-9]+ [a-z]+"`

1 or more numbers space 1 or more letters

Matches:
1812 Overture
76 trombones
101 photographs
7 arts

Quantifiers: {*min* , *max*}

{*min* , *max*} Preceding character(s) must occur *min* times, may occur *max* times

Example:

```
"|a[a-z]{1,3}[0-9]{1,4}"
```

"|a" 1-3 letters 1-4 numbers

Matches:

|aDA670
|aF73
|avid0023

Quantifiers: {*num*}

{*num*} Preceding character(s) must occur exactly *num* times

Example:

```
"008.{8}q"
```

"008" 8 unspecified characters "q"

Matches:

008••001130q
008••750601q

Grouping: (…)

- Allows a quantifier to apply to a string of multiple characters
- Particularly useful for making a string of characters optional — by adding {0,1}

Examples:

```
"Melvil(le){0,1}•Dewey"
"Thomas•(Alva•){0,1}Edison"
```

Matches:

Melville • Dewey
Melvil • Dewey
Thomas • Edison
Thomas • Alva • Edison

"Dot-star": .*

- Represents any number of unspecified characters (including none)

➢ *Handout page 4: example 4*

Problem: Find subfield codes that are repeated, but should not be, for example:

```
245 10 |a1876 :|ba novel /|by Gore Vidal.
245 10 |aCalifornia :|ca history /|cAndrew...
```

Solution:

```
245 matches "|a.*|a"
OR  245 matches "|b.*|b"
OR  245 matches "|c.*|c"
```

Position indicators – the dollar sign: $

- Represents the end of field *position* (does not itself stand for a character)
- Anchors what precedes it to the end of the field
- Must appear last in the regular expression

Example:

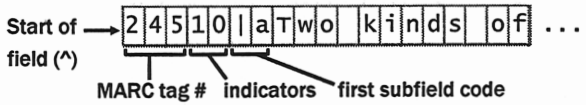
```
CALL # matches "196[0-9]$"
```

Matches: "090 |aQK47|b.F87 1967"

Does <u>not</u> match: "090 |aPE1963|b.C5"

Position indicators – the circumflex: ^

- Represents the start of field *position* (does not itself stand for a character)
- Anchors what follows it to the start of the field
- Must appear first in the regular expression
- For MARC variable tags, the field begins with the tag number as shown below



➢ ***More information on Handout page 3 (Section 2)***

"Escaping" a metacharacter: \

- The backslash indicates that the following character should be treated as a literal

<u>Example</u>	***<u>Matches</u>***
`\.\.\.$`	3 periods at the end of field
`\{[0-9]{3}\}`	any diacritic
`([a-z]\.){2,3}`	2 or 3 "initials" without spaces (e.g., "W.E.B.", "B.C.")

"{" any 3 numbers "}"

a letter followed by "." occurring 2 or 3 times

Putting it together ...

The real power of regular expressions comes from combining metacharacters in various ways

Examples

Problem: Find title statements that have a title proper only, that is, 245 with only a |a.

Solution:

```
245  matches  "|a[^|]+$"
```

Do not include |a in the Field name.

Examples

➢ ***Handout page 7: example 12***

Problem: **Limit a search to titles published *outside* of the United States.**

Solution:

```
COUNTRY matches "[^u]$"
```

Alternate version:

```
COUNTRY matches "^..[^u]"
```

Variation (Canada outside of Ontario and Quebec)

```
COUNTRY matches "[^oq].c"
```


Using the fixed-length fields

- Prior to Release 2005, regular expressions were very limited with the fixed-length fields — the expression could not contain more characters than the field itself
- This restriction has been lifted in Millennium (but not in character-based)
- Complex regular expressions may now be used with (as far as I can tell) all fixed-length fields, including fields with single-character codes

Examples

➢ ***Similar to handout page 9: example 18***

Problem: **Millennium Statistics reports a number of "bad codes" in the Item Status field.**

Solution:

```
ITEM STATUS matches [^oaqz12c-g]
```

Previous method:

```
     ITEM STATUS does not equal "o"
AND  ITEM STATUS does not equal "a"
AND  ITEM STATUS does not equal "q"
     [etc., etc.]
```

Examples

When searching for "bad codes," if the hyphen is a valid code, be sure to list it first (after "^") in the negated character class:

```
ORDER CLAIM matches
"[^-a-fnrz123567]"
```

The first hyphen is the hyphen character.

The second one indicates the range of letters a-f.

Examples

➢ *Handout page 8: example 15*

Problem: Find subject headings with 2nd indicator 4. Include MARC tags 600, 610, 611, 630, 650, and 651, but exclude 655 and 69x.

Solution: The MARC tag and indicators can be included in the search. Use the start of field indicator (^) so you don't match other numbers:

```
SUBJECT matches "^6[0135][01].4"
```

start of field "6"; cannot be "9"; cannot be "5"; any 1st ind. 2nd ind. "4"

Examples

➢ *Handout page 7: example 14*

Problem: Find titles with wrong filing indicators:

```
245 04 |aGrand Tour :|bthe lure of Italy ...
245 12 |aThe history of Charlotte Temple ...
```

Solution:

```
   245 matches "^245.2|a.[^•'"]"
OR 245 matches "^245.3|a..[^•'"]"
OR 245 matches "^245.4|a...[^•'"]"
[etc.]
```

245 tag, 2nd ind. 4; "|a"; any 3 characters; not space, ', or "

Examples

➢ *Handout page 6: example 10*

Problem: Find titles proper (MARC tag 245 |a) that are longer than 256 characters.

Solution: A quantifier cannot be greater than 127, so use multiple quantified subexpressions:

```
245|a matches "|a.{100}.{100}.{57}"
```

Note that this will match titles proper that *contain* 257 characters — they can be longer.

Examples

Problem: Find ISBNs with fewer than 10 characters.

Solution:

```
020 matches "|a.{0,9}$"
```

To find a string of characters shorter than a certain length, the string must be "anchored" at both ends. In this case, "|a" marks the beginning and "$" marks the end.

Examples

➢ *Handout page 9: example 20*

Problem: Find records with 2 (or more) call nos. The call nos. are distinquished only by length.

Solution:

```
     ( CALL # matches "|a.{6}$"
AND    CALL # matches "|a.{7}"  )
OR   ( CALL # matches "|a.{7}$"
AND    CALL # matches "|a.{8}"  )
     [etc.]
```


Examples

➢ *Handout page 9: example 19*

Problem: Find books where the number of pages given in 300 |a is 25 or fewer.

Complications:

- "Less than" won't work ("249 p." < "25 p.")
- Pages may not be given ("12 v.")
- Need to avoid matching part of the number ("23 p." will match "623 p.")
- May need to allow for preliminary pages (e.g., "iv, 24 p.")

Examples

Solution: Use 3 separate searches OR'd together to account for 1-9 pages, 10-19 pages, and 20-25 pages. Specify that the character preceding the page number is not a number:

```
   300|a matches "[^0-9][1-9]•p\."
OR 300|a matches "[^0-9]1[0-9]•p\."
OR 300|a matches "[^0-9]2[0-5]•p\."
```

Note: If there are no preliminary pages, the first character class ("[^0-9]") will match the "a" of "|a". Unfortunately, this will also match, for example: "123 p., 16 p. of plates"

Examples

➢ *Handout page 8: example 17*

Problem: Find any 856 tags that contain neither |3 (materials specified) nor |z (public note).

Solution: Because a record may have multiple 856s, only a regular expression will guarantee finding records where the |3 and |z are lacking in the same tag:

```
856 matches "^856..(|[^3z][^|]*)+$"
```


Pitfalls – negated character classes

- Even a negated character class ("[^…]") still has to match some character

Example:

```
DESCRIPTION matches "cm[^.]"
```

Matches: "300 |a56 p.;|c28 cm +|eatlas."

Does not match: "300 |a144 p.;|c23 cm"

Pitfalls – "zero" quantifiers

- Any subexpression quantified by * or {0,*max*} will itself match nothing, and therefore anything. These should always be preceded and followed by more specific subexpressions

Example:

```
041 matches "(lat){0,1}(grc){0,1}"
```

Both "lat" and "grc" are optional, so neither is required. This search retrieves *every* record that has an 041 tag of any kind.

Pitfalls – trying to do too much

- Everything in the search string must match
- To find more than one error, use separate search statements

Example: Find 700 fields that have invalid 1st or 2nd indicators:

```
700  matches  "^700[^013][^•2]"
```

matches only if both indicators are wrong. Use:

```
     700  matches "^700[^013]"
OR   700  matches "^700.[^•2]"
```

What Doesn't Work in Create Lists

- Regular expressions in Create Lists lack some features that may be available in other implementations, including:
  - Optional items — e.g., `"colou?r"`
  - Alternatives — e.g., `"(Donated by:|Gift of:)"`
  - Back references — e.g., `"([a-z]+)•\1"`
  - Special character classes, positions, etc. introduced with "\" — e.g., `"\d"`, `"\W"`, `"\b"`, `"\s"`, `"\012"`

➢ ***See Handout page 10 (Section 4) for more information, and some possible workarounds***

Conclusion

- Effective use of regular expressions in Create Lists requires:
  - Knowledge of regular expression syntax
  - Familiarity with your data
  - Familiarity with MARC format
- Practice and experiment! With Create Lists (unlike Global Update), there is little danger of messing things up

More information ...

Friedl, Jeffrey E. F.
Mastering Regular Expressions
2nd ed.
Sebastopol, CA: O'Reilly, 2002.
460 pp.

... $

Richard V. Jackson
Head of Copy Cataloging / Database Manager
Huntington Library, Art Collections, and Botanical Gardens
San Marino, California
rjackson@huntington.org
http://www.huntington.org/