Literary Data: Some Approaches

Andrew Goldstone
http://www.rci.rutgers.edu/~ag978/litdata

February 19, 2015. Data-type wrap-up; regular expressions.

factors

```
laureate_genre <- factor(c("novel", "short story", "novel",
                           "poetry", "novel"))
laureate_genre
```

```
[1] novel       short story novel       poetry
[5] novel
Levels: novel poetry short story
```

- `levels(laureate_genre)` for the levels

lists

hierarchy

```
clues <- list(
  absent=c("Assyrian", "Sensible Course"),
  present=list(
    unnecessary=c("Duchess", "Race"),
    necessary=list(
      invisible=c("Scandal", "Twisted"),
      visible=list(
        undecodable=c("Boscombe", "Five"),
        decodable=c("Red-Headed", "Identity")
      )
    )
  )
)
```

clues

```
$absent
[1] "Assyrian"        "Sensible Course"

$present
$present$unnecessary
[1] "Duchess" "Race"

$present$necessary
$present$necessary$invisible
[1] "Scandal" "Twisted"

$present$necessary$visible
$present$necessary$visible$undecodable
[1] "Boscombe" "Five"

$present$necessary$visible$decodable
[1] "Red-Headed" "Identity"
```

clues$present$unnecessary

```
[1] "Duchess" "Race"
```

clues$present$necessary$visible$decodable

```
[1] "Red-Headed" "Identity"
```

data frames

```
laureates[1:5, c("firstname", "surname", "year",
                 "bornCountry")]
```

```
  firstname      surname year bornCountry
1   Patrick      Modiano 2014      France
2     Alice        Munro 2013      Canada
3        Mo          Yan 2012       China
4     Tomas  Tranströmer 2011      Sweden
5     Mario Vargas Llosa 2010        Peru
```

frame indexing

```
frm[rows, cols]
```

- *blank*: keep them all
- number: choose that row/column
- numeric vector: choose those rows/columns
- logical: filter those rows/columns
- character vector: choose these named rows/columns
  (but rows don't have names by default)

shorthand

```
frm$colname == frm[, "colname"]
frm$colname[rows] == frm[rows, "colname"]
```

query logic

```
recent_flags <- laureates$year >= 2010
laureates[recent_flags, c("surname", "bornCity")]
```

```
       surname  bornCity
1      Modiano     Paris
2        Munro   Wingham
3          Yan     Gaomi
4  Tranströmer Stockholm
5 Vargas Llosa  Arequipa
```

- homework questions?

ordering

```
laureates[order(laureates$surname,
                laureates$firstname)[1:5],
          c("surname", "year")]
```

```
      surname year
49      Agnon 1966
38 Aleixandre 1977
54     Andric 1961
48   Asturias 1967
46    Beckett 1969
```

string search

```
grep(pattern, s) # which elements match pattern?
```

- Which of `laureates$bornCountry` contain `"now"`?

```
grep("now", laureates$bornCountry)
laureates$bornCountry[grep("now", laureates$bornCountry)]
```

or, for short

```
grep("now", laureates$bornCountry, value=T)
```

```
 [1] "Persia (now Iran)"
 [2] "Free City of Danzig (now Poland)"
 [3] "USSR (now Russia)"
 [4] "Austria-Hungary (now Czech Republic)"
 [5] "Russian Empire (now Lithuania)"
 [6] "Crete (now Greece)"
 [7] "Russian Empire (now Poland)"
 [8] "Austria-Hungary (now Ukraine)"
 [9] "Ottoman Empire (now Turkey)"
[10] "Bosnia (now Bosnia and Herzegovina)"
[11] "French Algeria (now Algeria)"
[12] "Russian Empire (now Finland)"
[13] "Russian Empire (now Poland)"
[14] "Prussia (now Germany)"
[15] "Prussia (now Germany)"
[16] "East Friesland (now Germany)"
[17] "British India (now India)"
[18] "Tuscany (now Italy)"
[19] "Schleswig (now Germany)"
```

string substitution

```
gsub(pattern, replacement, s) # globally replace

gsub("male", "male-identified", laureates$gender)
```

a new grammar: patterns

- most characters match themselves
- . matches any character

```
grep("197.", laureates$year, value=T)
```

```
 [1] "1979" "1978" "1977" "1976" "1975" "1974" "1973"
 [8] "1972" "1971" "1970"
```

meta: backslash

`\\`	next normal character is special
`\\d`	a digit
`\\s`	a white-space character (space, tab….)
`\\w`	a "word character" (letters…)
`\\D`	anything but a digit
`\\S`	anything but white space
`\\W`	anything but a word character

```
grep("\\W", laureates$surname, value=T)
```

```
[1] "Vargas Llosa"   "Le Clézio"      "García Márquez"
[4] "Martin du Gard" "O'Neill"        "von Heidenstam"
```

[*Edited 2/21/15*: This used to show `perl=T` but that was misleading. That option can work around *some* encoding issues but there are other approaches to encoding problems that are more comprehensive, and which I've used to fix this slide.

See Gries for examples of the kind of patterns that require `perl=T`.]

zero-width

`^`	the start of the string
`$`	the end of the string
`\\b`	a word boundary

```
grep("^Hungary", laureates$bornCountry, value=T)
grep("Hungary", laureates$bornCountry, value=T)
```

make-your-own classes

- `[...]` matches exactly one, *except*
  - `a-z` means the *range* (code order)
  - initial `^` means opposite day

quantifiers

`?`	one or none of previous
`*`	zero or more
`+`	one or more
`{n}`	exactly `n`
`{n,m}`	from `n` to `m` (can omit either)

```
spacey <- c("Doris Lessing",
            "Doris  Lessing",
            "Doris   Lessing")
grep("Doris Lessing", spacey)
```

```
[1] 1
```

- How to match all three?

```
grep("Doris\\s+Lessing", spacey)
```

```
[1] 1 2 3
```

anchors

```
grep("^M.*o", laureates$surname, value=T)
```

```
[1] "Modiano"  "Munro"    "Morrison" "Mahfouz"
[5] "Milosz"   "Montale"  "Mommsen"
```

```
grep("^M.*o$", laureates$surname, value=T)
```

```
[1] "Modiano" "Munro"
```

meta: backslash (2)

`\\`	next special character is normal
`\\. \\*`	a literal period, a literal asterisk
`\\+ \\?`	literal + and ?
`\\( \\[ \\{`	literal, literal, literal
`\\\\`	literal backslash

time to get grammatical

`(...)`*q*	quantifier *q* applies to everything in `(...)`
`(...|...)`	one or the other of the sides of the `|`

```
grep("^(\\w+ ){2,}", laureates$firstname, value=T)
```

```
 [1] "Sir Vidiadhar Surajprasad"
 [2] "Sir Winston Leonard Spencer"
 [3] "André Paul Guillaume"
 [4] "Carl Friedrich Georg"
 [5] "Carl Gustaf Verner"
 [6] "Gerhart Johann Robert"
 [7] "Count Maurice (Mooris) Polidore Marie Bernhard"
 [8] "Paul Johann Ludwig"
 [9] "Selma Ottilia Lovisa"
[10] "Christian Matthias Theodor"
```

pattern substitution

- in substitution string, \\n corresponds to *n*th parenthesized expression in pattern

```
many_names <- laureates$firstname[c(7, 99)]
many_names
```

```
[1] "Jean-Marie Gustave"   "Selma Ottilia Lovisa"
```

```
gsub("(\\w+) .*$", "\\1", many_names)
```

```
[1] "Jean-Marie" "Selma"
```

cleanup

```
tricky_years <- c("1774.", "[1793]", "[1795?]", "1792-96.")
gsub("^som(eth)ing$", "\\1", tricky_years)
```

```
gsub("^\\D*(\\d{4}).*$", "\\1", tricky_years)
```

```
[1] "1774" "1793" "1795" "1792"
```

next

- Hockey, McCarty, McPherson, Kirschenbaum
- http://www.rci.rutgers.edu/~ag978/litdata/hw5
- read Gries according to the guide in homework 5
- groups…