A systematic approach to data cleaning with R

Mark van der Loo

markvanderloo.eu | @markvdloo

Budapest | September 3 2016
Demos and other materials

https://github.com/markvanderloo/satRday
The statistical value chain
From raw to technically correct data
From technically correct to consistent data

Contents

▶ The statistical value chain
▶ From raw data to technically correct data
  ▶ Strings and encoding
  ▶ Regexp and approximate matching
  ▶ Type coercion
▶ From technically correct data to consistent data
  ▶ Data validation
  ▶ Error localization
  ▶ Correction, imputation, adjustment
The statistical value chain
From raw to technically correct data
From technically correct to consistent data

Mark van der Loo
A systematic approach to data cleaning with R
Statistical value chain

- **Raw data**
  - type checking, normalizing
- **Technically correct data**
  - fix and impute
- **Consistent data**
  - estimate, analyze, derive, etc.
- **Statistical results**
  - tabulate, plot
- **Formatted output**

Mark van der Loo | A systematic approach to data cleaning with R
## The statistical value chain

From raw to technically correct data
From technically correct to consistent data

### Concepts

#### Technically correct data

- Well-defined format (data structure)
- Well-defined types (numbers, date/time,string, categorical... )
- Statistical units can be identified (persons, transactions, phone calls...)
- Variables can be identified as properties of statistical units.
- Note: tidy data $\subset$ technically correct data

#### Consistent data

- Data satisfies demands from domain knowledge
- (more on this when we talk about validation)
From raw to technically correct data
Dirty tabular data

Demo

Coercing while reading: /table
Tabular data: long story short

- `read.table`: R’s swiss army knife
  - fairly strict (no sniffing)
  - Very flexible
  - Interface could be cleaner (see this talk)

- `readr::read_csv`
  - Easy to switch between strict/lenient parsing
  - Compact control over column types
  - Fast
  - Clear reports of parsing failure
Really dirty data

Demo

Output file parsing: /parsing
A few lessons from the demo

- (base) R has great text processing tools.
- Need to work with regular expressions\(^1\)
- Write many small functions extracting single data elements.
- Don’t overgeneralize: adapt functions as you meet new input.
- Smart use of existing tools (\texttt{read.table(text=)})

\(^1\)Mastering Regular Expressions (2006) by Jeffrey Friedl is a great resource
Packages for standard format parsing

- `jsonlite`: parse JSON files
- `yaml`: parse yaml files
- `xml2`: parse XML files
- `rvest`: scrape and parse HTML files
Some tips on regular expressions with R

- stringr has *many* useful shorthands for common tasks.
- Generate regular expressions with `rex`

```r
library(rex)
# recognize a number in scientific notation
rex(one_or_more(digit),
    , maybe("." ,one_or_more(digit))
    , "E" %or% "e"
    , one_or_more(digit))
```

```r
## (?:[[:digit:]]+)(?:\.(?:[[:digit:]]+))?(?:E|e)(?:[[:digit:]]+)
```
Regular expressions

Express a pattern of text, e.g.

"(a|b)c*" = \{"a", "ac", "acc", ..., "b", "bc", "bcc", ...\}

<table>
<thead>
<tr>
<th>Task</th>
<th>stringr function</th>
</tr>
</thead>
<tbody>
<tr>
<td>string detection</td>
<td>str_detect(string, pattern)</td>
</tr>
<tr>
<td>string extraction</td>
<td>str_extract(string, pattern)</td>
</tr>
<tr>
<td>string replacement</td>
<td>str_extract(string, pattern, replacement)</td>
</tr>
<tr>
<td>string splitting</td>
<td>str_split(string, pattern)</td>
</tr>
</tbody>
</table>

Base R: grep grepl | regexpr regmatches | sub gsub | strsplit
String normalization

Bring a text string in a standard format, e.g.

- **Standardize upper/lower case (casefolding)**
  - `stringr`: `str_to_lower`, `str_to_upper`, `str_to_title`
  - `base R`: `tolower`, `toupper`

- **Remove accents (transliteration)**
  - `stringi`: `stri_trans_general`
  - `base R`: `iconv`

- **Re-encoding**
  - `stringi`: `stri_encode`
  - `base R`: `iconv`

- **Uniformize encoding (unicode normalization)**
  - `stringi`: `stri_trans_nfkc` (and more)
Encoding
Encoding in R

Tell R how to interpret it.
(here: lowercase it)

Single element of a character vector.
Encoding in R

Encoding(X) <- "UTF-8"

only changes the labels, not the data

Use iconv() to change the encoding.
Encoding in R

Demo

Normalization, re-encoding, transliteration: /strings
A few tips

Detect encoding: `stringi::stri_enc_detect`
Conversion options: `iconvlist()` `stringi::stri_enc_list()`
Approximate text matching

Regex

all strings over alphabet \( \Sigma \)

\( (a1b)c^+ \)

\{ "a", "ac", "acc"... \} 
\union \{ "b", "bc"... \}

Approximate Matching

\( \Sigma^* \)

\{ s \in \Sigma^* : \delta(\text{"hello"}, s) \leq 1 \}

\text{hellow, hallo, } \text{ello, } \text{hello, } \delta = 1

\text{... } 
\text{...}
Demo

Approximate matching and normalization: /matching
Approximate text matching: edit-based distances

<table>
<thead>
<tr>
<th>Distance</th>
<th>substitution</th>
<th>deletion</th>
<th>insertion</th>
<th>transposition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hamming</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>LCS</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Levenshtein</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>OSA</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Damerau-Levenshtein</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

*Substrings may be edited only once.

"leela" → "leea" → "leia"

```r
stringdist::stringdist("leela","leia",method="dl")
```

```
## [1] 2
```

Mark van der Loo
A systematic approach to data cleaning with R
Some pointers for approximate matching

- Normalisation and approximate matching are complementary
- See my useR2014 talk or paper on stringdist for more distances
- The fuzzyjoin package allows fuzzy joining of datasets
Other good stuff

- lubridate: extract dates from strings
  
  ```r
  lubridate::dmy("17 December 2015")
  ```
  
  ```r
  ## [1] "2015-12-17"
  ```

- tidyr: many data cleaning operations to make your life easier

- readr: Parse numbers from text strings
  
  ```r
  readr::parse_number(c("2%","6%","0.3%"))
  ```
  
  ```r
  ## [1] 2.0 6.0 0.3
  ```
From technically correct to consistent data
The statistical value chain
From raw to technically correct data
From technically correct to consistent data

The mantra of data cleaning

- Detection (data conflicts with domain knowledge)
- Selection (find the value(s) that cause the violation)
- Correction (replace them with better values)
Detection, AKA data validation

Informally:
Data Validation is checking data against (multivariate) expectations about a data set.

Validation rules
Often these expectations can be expressed as a set of simple validation rules.
Data validation

Demo

The validate package /validate
The validate package, in summary

- Make data validation rules explicit
- Treat them as objects of computation
  - store to / read from file
  - manipulate
  - annotate
- Confront data with rules
- Analyze/visualize the results
Tracking changes when altering data
Tracking changes in rule violations

The statistical value chain
From raw to technically correct data
From technically correct to consistent data

Total #checks

Verifiable
- Violateda
  - Still violated
  - New violated
- Satisfied
  - Still satisfied
  - New satisfied

Not verifiable
- Still not verifiable
- New not verifiable

Mark van der Loo
A systematic approach to data cleaning with R
Use rules to correct data

Main idea
Rules restrict the data. Sometimes this is enough to derive a correct value uniquely.

Examples
- Correct typos in values under linear restrictions
  - $123 + 45 \neq 177$, but $123 + 54 = 177$.
- Derive imputations from values under linear restrictions
  - $123 + \text{NA} = 177$, compute $177 - 123 = 54$.

Both can be generalized to systems $Ax \leq b$. 
Deductive correction and imputation

Demo

The deductive package: /deductive.
Selection, or: error localization

Fellegi and Holt (1976)

Find the least (weighted) number of fields that can be imputed such that all rules can be satisfied.

Note

- Solutions need not be unique.
- Random one chosen in case of degeneracy.
- Lowest weight need not guarantee smallest number of altered variables.
Error localization

Demo

The `errorlocate` package: `/errorlocate`
Notes on `errorlocate`

- For in-record rules
- Support for
  - linear (in)equality rules
  - Conditionals on categorical variables (if male then not pregnant)
  - Mixed conditionals (has job then age $\geq 15$)
  - Conditionals w/linear predicates (staff $> 0$ then staff cost $> 0$)
- Optimization is mapped to MIP problem.
Missing values

Mechanisms (Rubin):

- **MCAR**: missing completely at random
- **MAR**: $P(Y = \text{NA})$ depends on value of $X$
- **MNAR**: $P(Y = \text{NA})$ depends on value of $Y$
### Purpose of imputation vs prediction

- **Prediction**: estimate a single value (often for a single use)
- **Imputation**: estimate values such that the completed data set allows for valid inference\(^a\)

\(^a\)This is very difficult!

### Imputation methods

- Deductive imputation
- Imputation based on predictive models
- Donor imputation (knn, pmm, sequential/random hot deck)
Predictive model-based imputation

\[ \hat{y} = \hat{f}(x) + \epsilon \]

e.g. Linear regression

\[ \hat{y} = \alpha + x^T \hat{\beta} + \epsilon \]

- Residual:
  - \( \epsilon = 0 \) Impute expected value
  - \( \epsilon \) drawn from observed residuals \( e \)
  - \( \epsilon \sim N(0, \sigma) \) parametric residual, \( \hat{\sigma}^2 = \text{var}(e) \)

- Multiple imputation (Bayesian bootstrap)
  - Draw \( \beta \) from parametric distribution, impute multiple times.
Donor imputation (hot deck)

Method variants:

- **Random hot deck:** copy value from random record.
- **Sequential hot deck:** copy value from previous record.
- **k-nearest neighbours:** draw donor from k nearest neighbours.
- **Predictive mean matching:** copy value closest to prediction.

Donor pool variants:

- per variable
- per missing data pattern
- per record
Many multivariate methods seem relatively *ad hoc*, and more theoretical and empirical comparisons with alternative approaches would be of interest.

Demo time

Demo

Imputation / imputation

- VIM: visualisation, GUI, extensive methodology
- simputation: simple, scriptable interface to common methods
Methods supported by simputation

- Model based (optionally add [non-]parametric random residual)
  - linear regression
  - robust linear regression
  - CART models
  - Random forest
- Donor imputation (including various donor pool specifications)
  - k-nearest neighbour (based on gower’s distance)
  - sequential hotdeck (LOCF, NOCB)
  - random hotdeck
  - Predictive mean matching
- Other
  - (groupwise) median imputation (optional random residual)
  - Proxy imputation (copy from other variable)
Credits

- deductive Mark van der Loo, Edwin de Jonge
- errorlocate Edwin de Jonge, Mark van der Loo
- gower Mark van der Loo
- jsonlite Jeroen Ooms, Duncan Temple Lang, Lloyd Hilaiel
- magrittr Stefan Milton Bache, Hadley Wickham
- rex Kevin Ushey Jim Hester, Robert Krzyzanowski
- simputation Mark van der Loo
- stringdist Mark van der Loo, Jan van der Laan, R Core, Nick Logan
- stringi Marek Gagolewski, Bartek Tartanus
- stringr Hadley Wickham, RStudio
- tidyR Hadley Wickham, RStudio
- validate Mark van der Loo, Edwin de Jonge
- VIM Matthias Templ, Andreas Alfons, Alexander Kowarik, Bernd Prantner
- xml2 Hadley Wickham, Jim Hester, Jeroen Ooms, RStudio, R foundation