Finding and handling data errors

errorlocate

Edwin de Jonge and Mark van der Loo

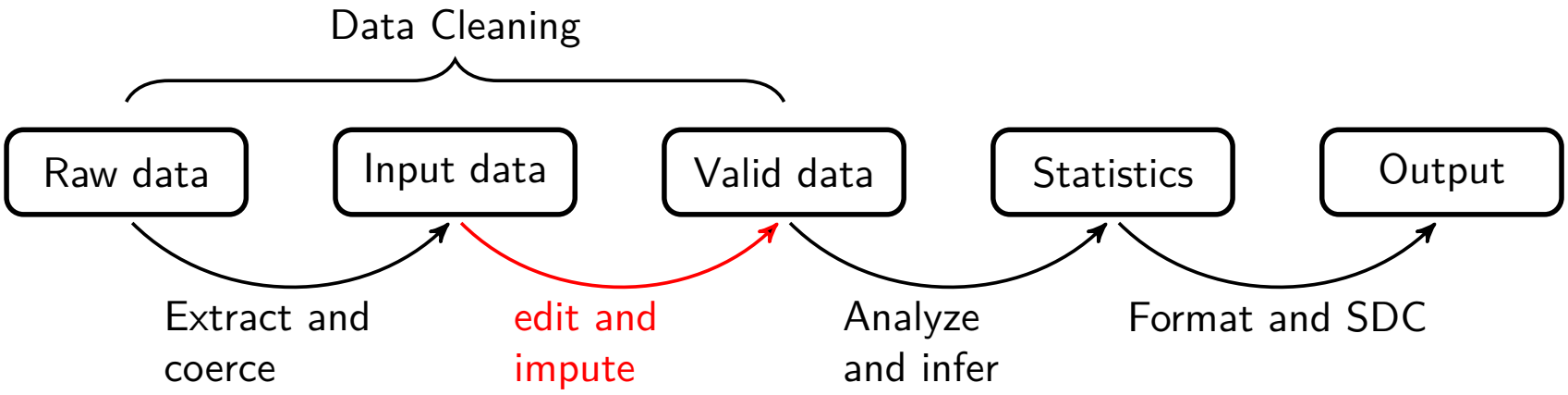
uRos2018 Tutorial Session, The Hague

Packages

To run the examples please install:

```
install.packages("errorlocate")
```

Finding errors in records.



Validation rules?

As we saw package `validate` allows to:

- formulate explicit data rule that data must conform to:

```
library(validate)
rules <- validator(
  age >= 0,
  age < 150,
  if (driver_license == TRUE) age >= 16
)
```

Explicit validation rules:

- Give a clear overview what the data must conform to.
- Can be used to reason about.
- Can be used to fix/correct data!
- Find error, and when found correct it.

Note:

- Manual editing is error prone, not reproducible and not feasible for large data sets.
- Large rule sets have (very) complex behavior, e.g. entangled rules: adjusting one value may invalidate other rules.

`validate` tells us:

- which **records** are **invalid**
- which data **rules** are **violated**.

`errorlocate` tells us:

- which field/**variable**(s) should be fixed to make a record valid.

Data example

```
rules <- validator(
  age >= 0,
  age < 150,
  if (driver_license == TRUE) age >= 16
)

# invalid data
car_owners <- data.frame(age = 160, driver_license = TRUE)
```

Question: *Which variable (likely) is incorrect? Why?*

Data example

```
rules <- validator(
  age >= 0,
  age < 150,
  if (driver_license == TRUE) age >= 16
)

# invalid data
car_owners <- data.frame(age = 160, driver_license = TRUE)
```

Clearly age is incorrect, because it violates the second constraint.

Data example 2

```
rules <- validator(
  age >= 0,
  age < 150,
  if (driver_license == TRUE) age >= 16
)

car_owners <- data.frame(age = 10, driver_license = TRUE)
```

Question: *Which variable (likely) is incorrect? Why?*

Data example 2

```
rules_dl <- validator(
  age >= 0,
  age < 150,
  if (driver_license == TRUE) age >= 16
)

car_owners <- data.frame(age = 10, driver_license = TRUE)
```

It depends on the quality of `age` and `driver_license`. We can add more weight to `age` if we think that variable has better quality.

Error localization

Error localization is a procedure that points out fields in a data set that can be altered or imputed in such a way that all validation rules can be satisfied.

Assignment:

```
rules <- validator(
  if (married == TRUE) age >= 16,
  if (attends == "kindergarten") age <= 6
)

persons <- data.frame( age      = 3
                     , married  = TRUE
                     , attends  = "kindergarten"
                     )
```

a) check with `validate` which rules are violated.
b) What should be changed to this record to "correct" it? Why?

Feligi Holt (FH) formalism:

> *Find the minimal (weighted) number of variables that cause the invalidation of the data rules.*

Makes sense if no further knowledge on the error mechanism is available!
R package `errorlocate` (second generation of `editrules`) implements FH.

Feligi Holt (FH) formalism:

But there are exceptions...

- In balance sheets, swapping variables (2 edits) sometimes makes more sense then adjusting one value (1 edit). (see R package:`deducorrect`).
- In some data, spreading a surplus or shortage on a variable over many variables is sensible. (see R package: `rspa`).

errorlocate

- R-package that implements FH.
- Is extensible (you can plug in your own detection stuff)
- provides:
  - `locate_errors`
  - `replace_errors`
  - R5 classes to add your own stuff.

errorlocate::locate_errors

```
locate_errors( data.frame( age  = 3
                         , married = TRUE
                         , attends = "kindergarten"
                         )
     , validator( if (married == TRUE) age >= 16
                , if (attends == "kindergarten") age <= 6
                )
     )
```

```
## call:  x$locate(data = data, weight = weight, ...)
## located  1  error(s).
## located  0  missing value(s).
## Use 'summary', 'values', '$errors' or '$weight', to expl
```

errorlocate::locate_errors

```
locate_errors( data.frame( age  = 3
                         , married = TRUE
                         , attends = "kindergarten"
                         )
     , validator( if (married == TRUE) age >= 16
                , if (attends == "kindergarten") age <= 6
                )
     )$errors
```

```
##         age married attends
## [1,] FALSE    TRUE   FALSE
```

Assignment (small examples)

a) Find the error with `locate_errors`:

```
data.frame( age = 26, married = TRUE
          , attends= "kindergarten")
```

b) Find the error with `locate_errors`:

```
data.frame( age = 15, married = TRUE
          , attends= "kindergarten")
```

Do you agree?

c) Run `locate_errors` with the `rules_dl` (!) and add a weight of 2 to age.

```
data.frame(age=10,driver_license = TRUE)
```

Removing errors

- Detecting errors is very useful, but then what?
- Fixing philosophy is:
  - Find erroneuous values.
  - Remove them (i.e. make them `NA`).
  - Impute them with sensible values.

Note

We could also remove erroneous records completely, but often this result in *over-deletion* and introduces a *bias*.

errorlocate::replace_errors

- Locates errors and replaces them with `NA`.

```
replace_errors(
    data.frame( age      = 3
              , married = TRUE
              , attends = "kindergarten"
              )
  , validator( if (married == TRUE) age >= 16
             , if (attends == "kindergarten") age <= 6
             )
)
```

```
##   age married       attends
## 1   3      NA kindergarten
```

Assignment

a) Use the data set `retailers` from validation.
b) Use validate to find out which records are `faulty`
c) use `locate_errors` to find some errors.
d) use `replace_errors` to "fix" the data set.

Internal workings:

`errorlocate`:

- translates error localization problem into a **mixed integer problem**, which is solved with `lpsolveAPI`.
- contains a small framework for implementing your own error localization algorithms.

Pipe friendly

The `replace_errors` function is pipe friendly:

```
rules <- validator(age < 150)

data_noerrors <-
  data.frame(age=160, driver_license = TRUE) %>%
  replace_errors(rules)

errors_removed(data_noerrors) # contains errors removed
```