TALL SOM FORTELLER

Introduction course

SUSIE JENTOFT, ASLAUG FOSS

DAY 3: 5TH MARS 2025

Statistisk sentralbyrå
Statistics Norway

Agenda: Part 1

	Wednesday 26th February	**Friday 28th February**	**Wednesday 5th Mars**
12:00	• Introduction • Github • Basic calculations • Objects	• Review • Data manipulation	• **Review** • **Data validation**
12:45	Exercise 1	Exercise 3	**Exercise 5**
13:30	• Logical statements • Read in data	• Merging datasets • Graphics	• **Data imputation**
14:00	Exercise 2	Exercise 4	**Exercise 6**
14:50 – 15:00			**Summary**

Review (day 2)

- Use ..._join() for merging
- Select some rows: filter( )
- Select variables/columns with select()
- Summary statistic: summarise ( )
- Plot: ggplot( ), aes( ), geom_...( )

Exercise 4 review

Outlier detecion - validation

- **Data validation** is an activity aimed at **verifying** whether the value of a data item comes from the given set of **acceptable values**. (OECD glossary)

- Methodology for data validation, EUROSTAT (https://ec.europa.eu/eurostat/cros/content/ess-handbook-methodology-data-validation-v11-rev2018-0_en)

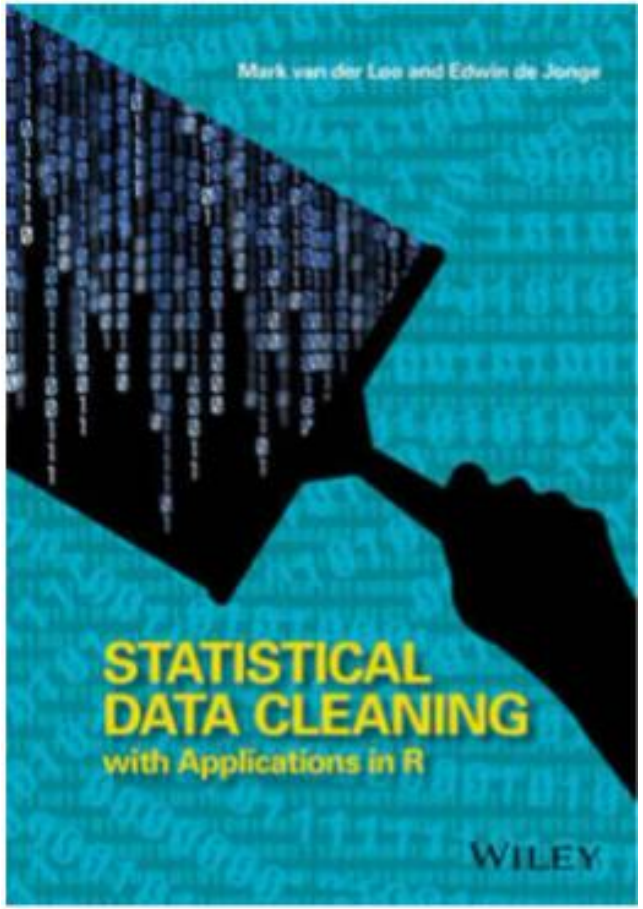
Validate package in R

- The validate package is intended to make:
  - Checking the data easy
  - Maintaining the rules easy
  - Possible to reproduce the results

- Build by Mark van der Loo and Edwin de Jonge, Statistics Netherlands

Validate package in R

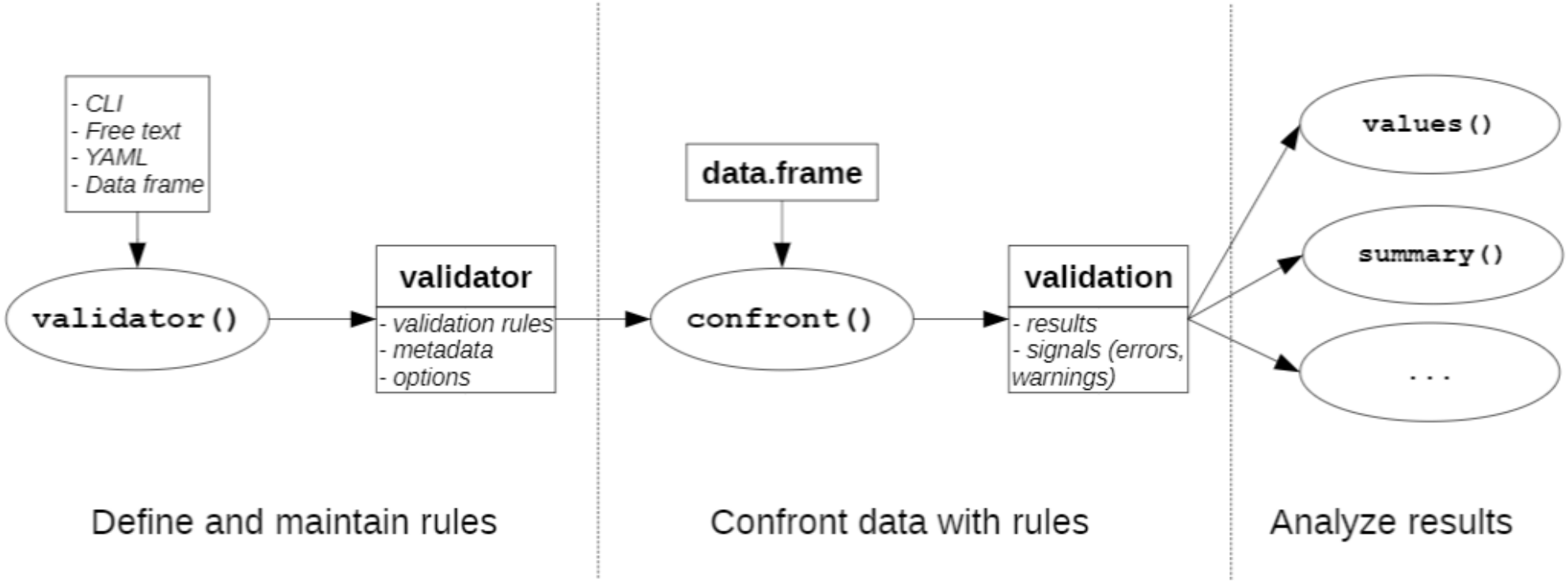
- test data against a reusable set of data validation rules:
- investigate, summarise, and visualise data validation results;
- investigate, summarise, and visualise rule sets.



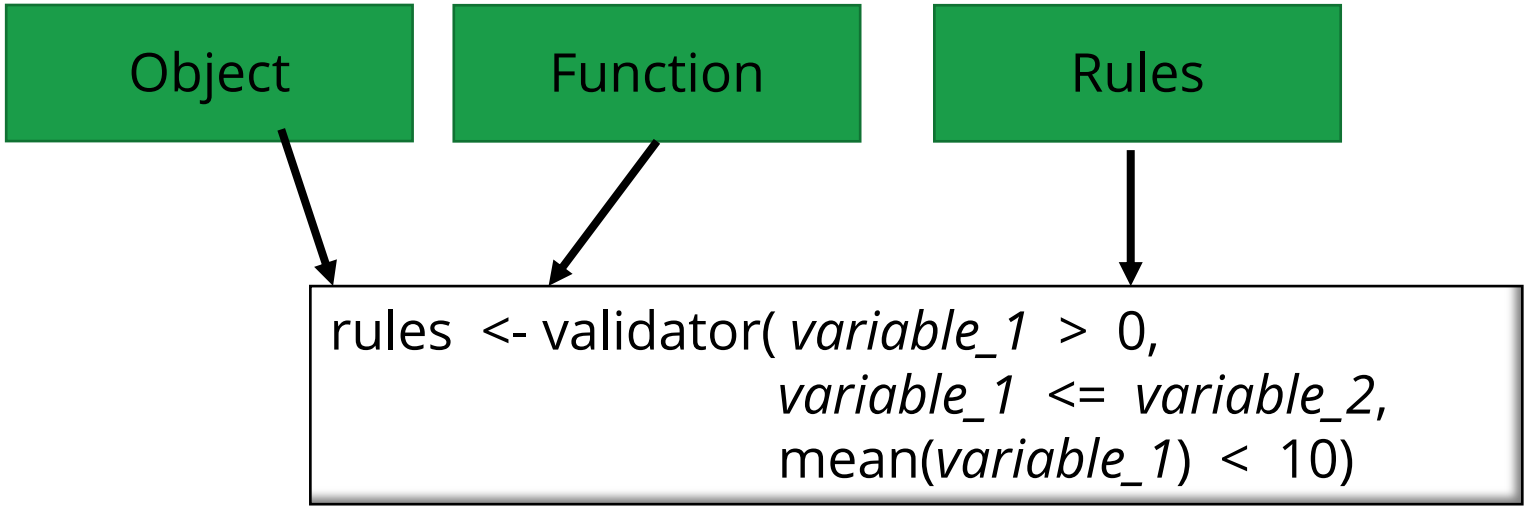
Mark van der Loo and Edwin de Jonge, Statistics Netherlands

Statistisk sentralbyrå
Statistics Norway

The validate package



Validator


Object
Function
Rules
rules <- validator(variable_1 > 0,
variable_1 <= variable_2,
mean(variable_1) < 10)

Variable type

- Numeric variable: is.numeric(variable_name)
- Character variable: is.character(variable_name)
- Type checks: any function starting with is.

Field length: nchar & field_length()

- nchar(variable_name) >= value

- field_length(variable_name, min=2, max=3)

Format: number_format()

- number_format(variable_name, format="d.dd")

format	match	non-match
`0.dddd`	`"0.4321"`	`"0.123"`, `"1.4563"`
`d.ddEdd`	`"3.14E00"`	`"31.14E00"`
`d.*Edd`	`"0.314E01"`, `"3.1415297E00"`	`"3.1415230"`
`d.dd*`	`"1.23"`, `"1.234"`,...	`"1.2"`

Numeric ranges: in_range()

- in_range(variable_name, min=0, max=1)
  - includes the boundaries of the range,

- variable_name < =1

Code lists : %in% list

- fruit %in% c("aple","pear","orange","banana")

Uniqueness: is_unique()

- is_unique() checks whether combinations of variables (usually key variables) uniquely identify a record.

- is_unique(region, period, measure)

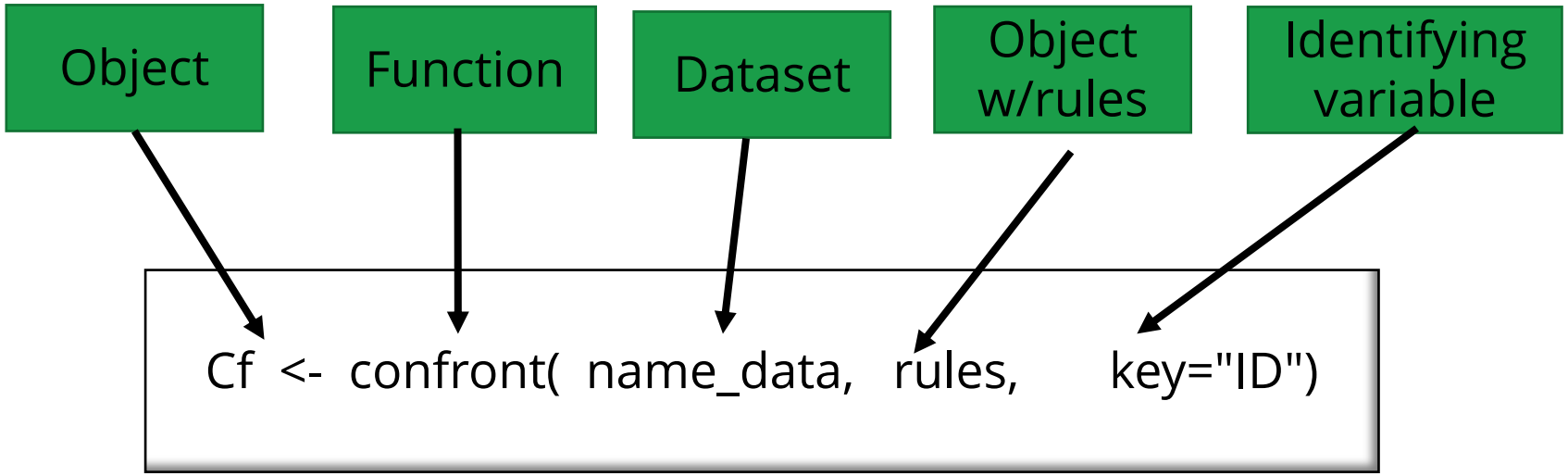
Completeness or missing

- is_complete(variable_name)
- !is.na(variable_name)
  - «!» means not

Validation rule syntax

- Type checks: any function starting with is.
- Binary comparisons: <, <=, ==, !=, >=, > and %in%.
- Unary logical operators: !, all(), any().
- Binary logical operators: &, &&, |, ||
- logical implication, e.g. if (staff > 0) staff.costs > 0
- For more functions, see: https://cran.r-project.org/web/packages/validate/vignettes/cookbook.html

Confront data with rules



The outcome of confronting data set with rules

- summary: summarize output; returns a data.frame
- aggregate: aggregate validation in several ways
- values: Get the values in an array, or a list of arrays if rules have different output dimension structure
- errors: Retrieve error messages caught during the confrontation
- warnings: Retrieve warning messages caught during the confrontation.
- sort : aggregate and sort in several ways

Metadata for the rules

- The following functions can be used to **get** or **set metadata**:
  - origin : Where was a rule defined?
  - names : The name per rule
  - created : when were the rules created?
  - label : Short description of the rule
  - description: Long description of the rule
  - meta: Set or get generic metadata

Summary

summary(cf)

	name	items	passes	fails	nNA	error	warning	expression
1	V1	4	3	1	0	FALSE	FALSE	var1 > 0
2	V2	4	3	1	0	FALSE	FALSE	(var1 - var2) <= 1e-08
3	V3	1	1	0	0	FALSE	FALSE	mean(var1) < 10

- How many data items were checked against each rule
- How many items passed, failed or resulted in NA
- Whether the check resulted in an error (could not be performed) or gave an error
- The expression that was actually evaluated to perform the check.

Aggregate

aggregate(cf)

```
> aggregate(cf)
   npass nfail nNA rel.pass rel.fail rel.NA
V1     3     1   0     0.75     0.25      0
V2     3     1   0     0.75     0.25      0
V3     1     0   0     1.00     0.00      0
```

keys	If confront was called with key=
npass	Number of items passed
nfail	Number of items failing
nNA	Number of items resulting in NA
rel.pass	Relative number of items passed
rel.fail	Relative number of items failing
rel.NA	Relative number of items resulting in NA

Values

values(cf)

```
> values(cf)
[[1]]
     V1    V2
1  TRUE  TRUE
2  TRUE FALSE
3 FALSE  TRUE
4  TRUE  TRUE

[[2]]
       V3
[1,] TRUE
```

#Dataset with indikators

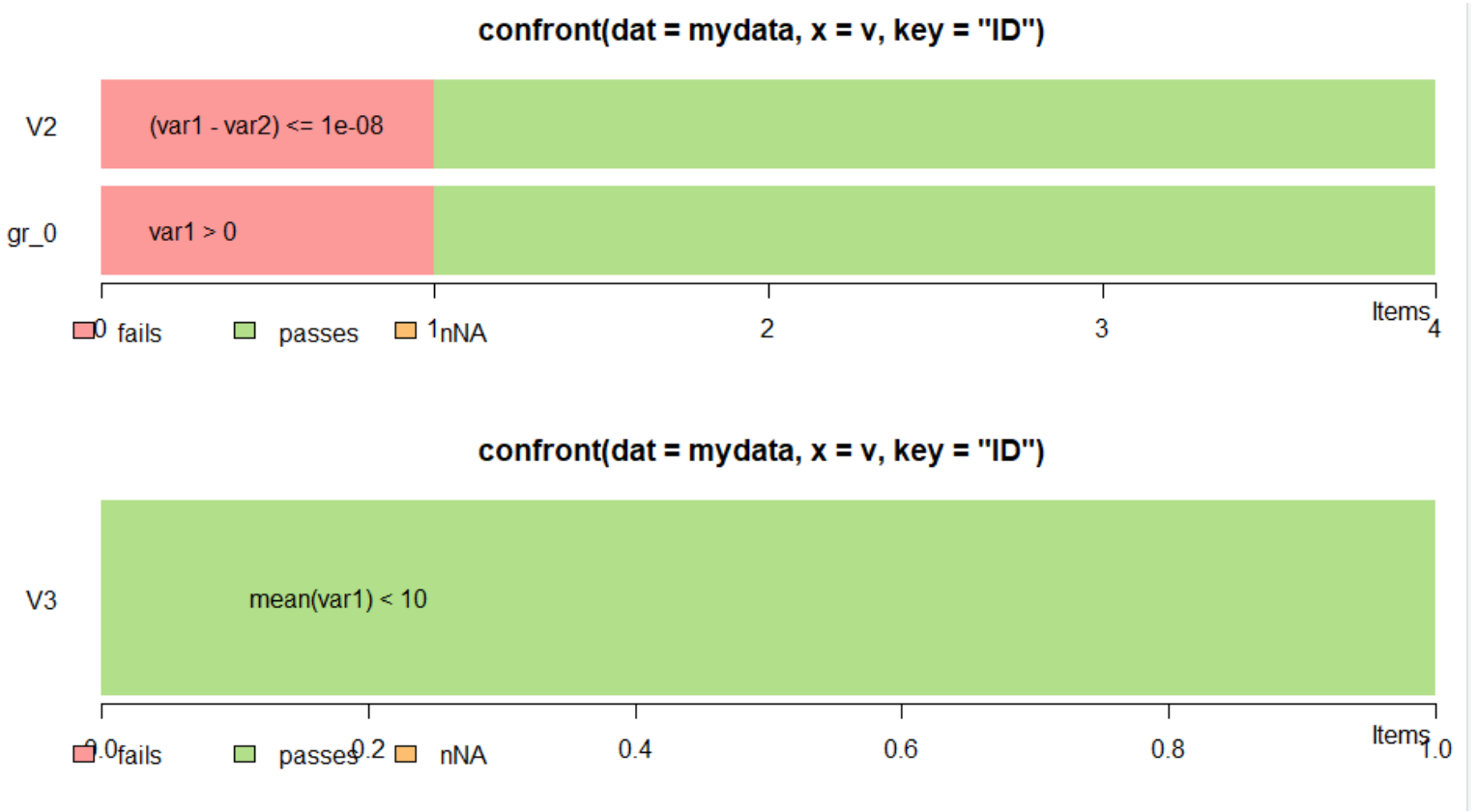
```
ind<-as.data.frame(values(cf))

# add indikators to datasett
mydata2 <- mydata %>%
  mutate(greater_0 = pull(ind, V1),
         V2= pull(ind, V2))
```

	ID	var1	var2	V1	V2
1	1	2	9	TRUE	TRUE
2	2	9	1	TRUE	FALSE
3	3	-1	4	FALSE	TRUE
4	4	7	8	TRUE	TRUE

Graphics

plot(cf)



Exercise 5:

- Exercise 5 is in the file : Exercises_day3.R
- Need to download R-package:

✓validate

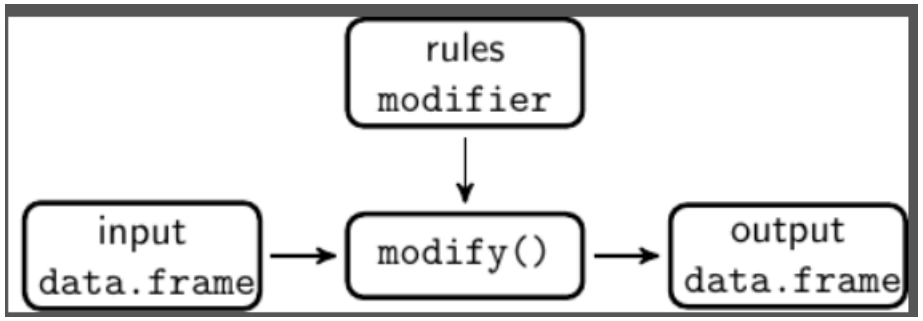
Exercise 5 review

Rule based imputation with «dcmodify»

- **'if this do that'** type of statements.
- Based on **expert knowledge**.
- All 'data modifying rules' are **gathered**.
- Easy to maintain and document the rules

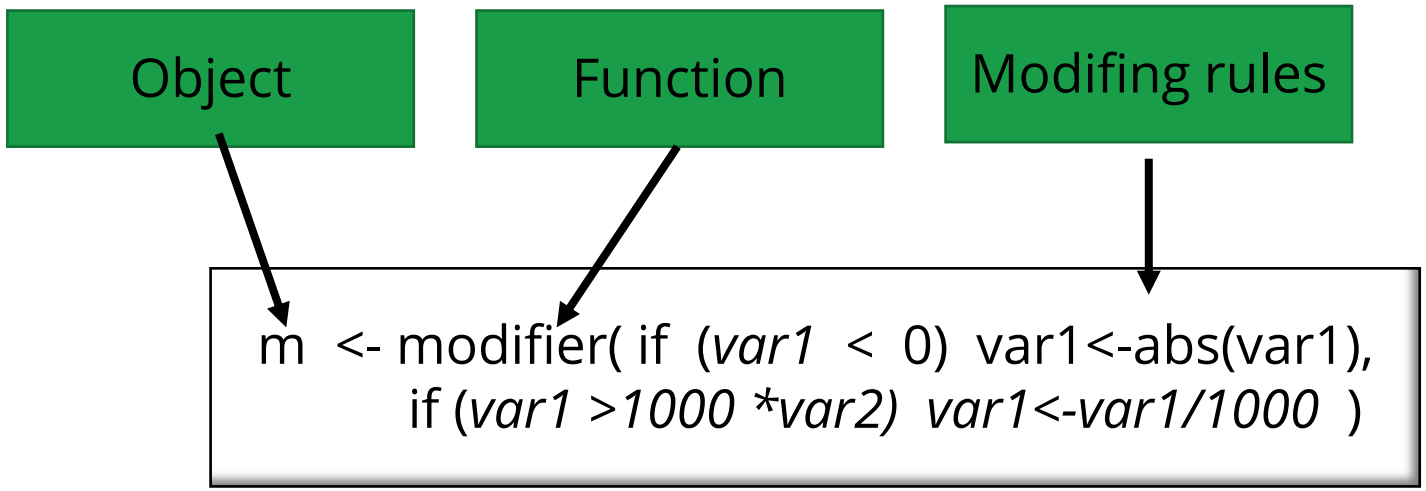
- Build by Mark van der Loo and Edwin de Jonge, Statistics Netherlands
- https://cran.r-project.org/web/packages/dcmodify/vignettes/introduction.html

Basic workflow

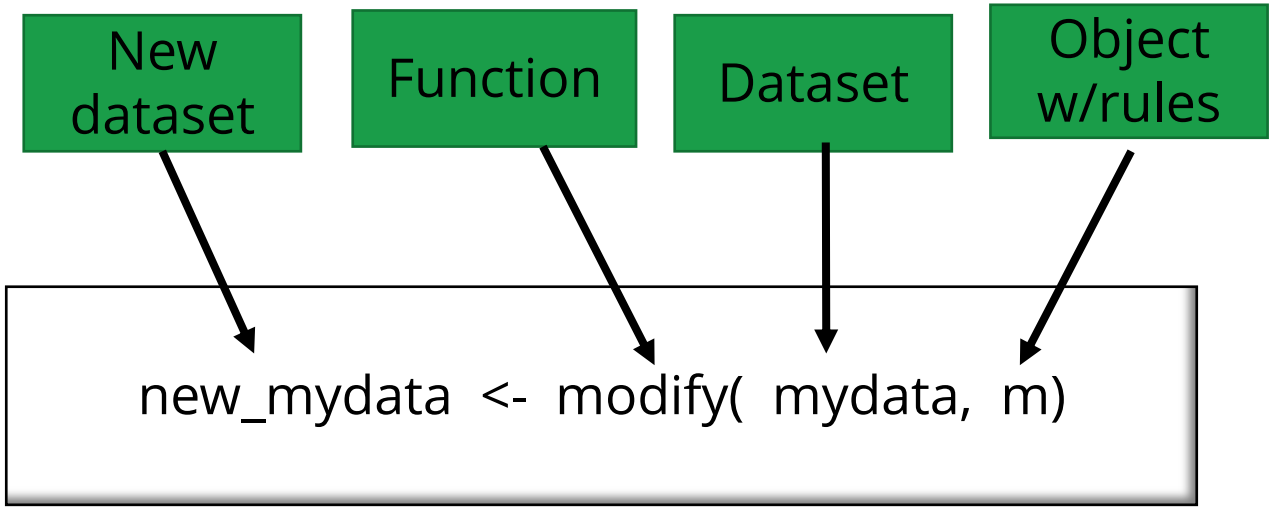


- data: This is your data, currently this must be stored in a data.frame.
- modifier: This is an object that stores data modification rules.
- modify: This is a function that applies the rules in a modifier to your data.

Modifier – defining rules



Modifying data with rules



Model based imputation with “simputation”

- A package to make imputation simpler!
- Number of commonly used single imputation methods
- Each with a similar and simple interface

- Build by Mark van der Loo and Edwin de Jonge, Statistics Netherlands
- More information: https://cran.r-project.org/web/packages/simputation/vignettes/intro.html

Imputation methods available

Model based imputation

- linear regression
- robust linear regression
- ridge/elasticnet/lasso regression
- CART models (decision trees)
- Random forest

Multivariate imputation

- Imputation based on the expectation-maximization algorithm
- missForest (=iterative random forest imputation)

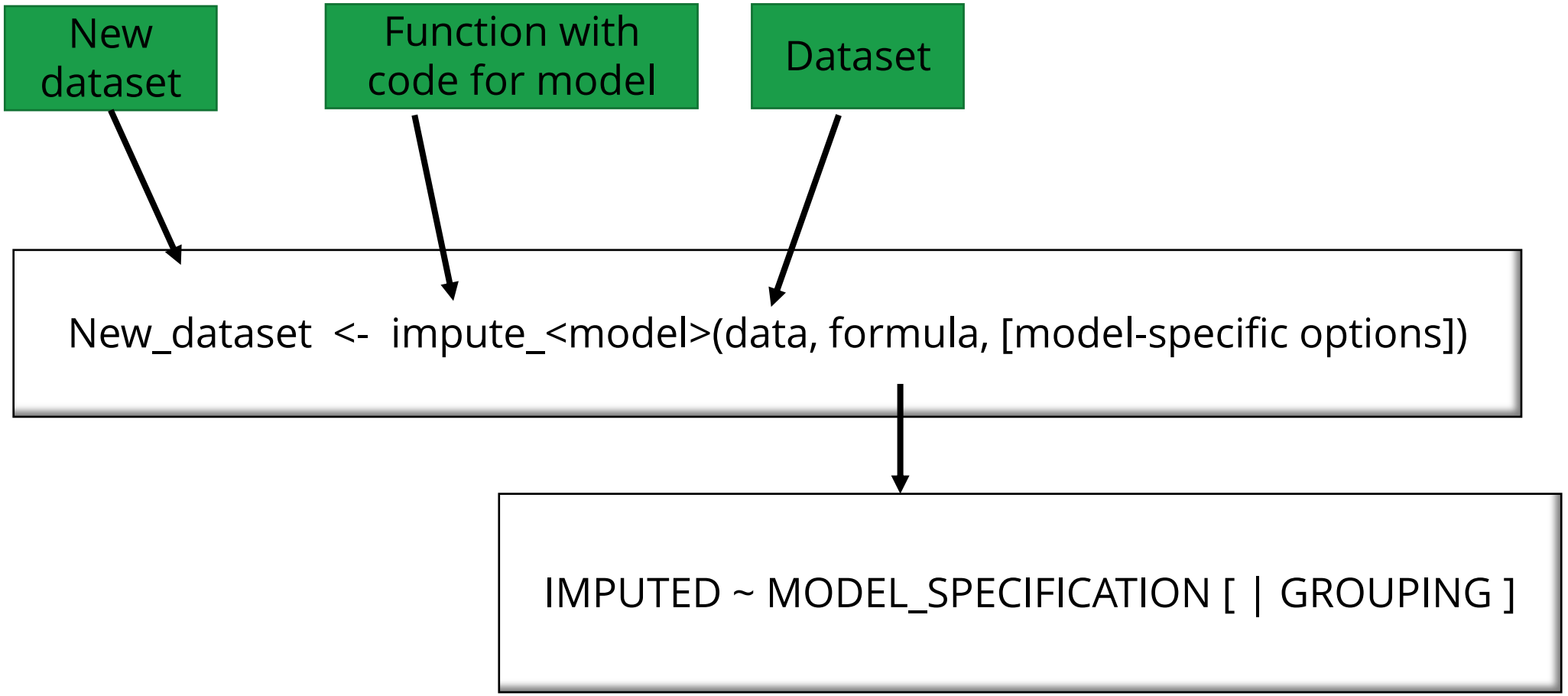
Donor imputation

- k-nearest neigbour (based on gower's distance)
- sequential hotdeck (LOCF, NOCB)
- random hotdeck
- Predictive mean matching

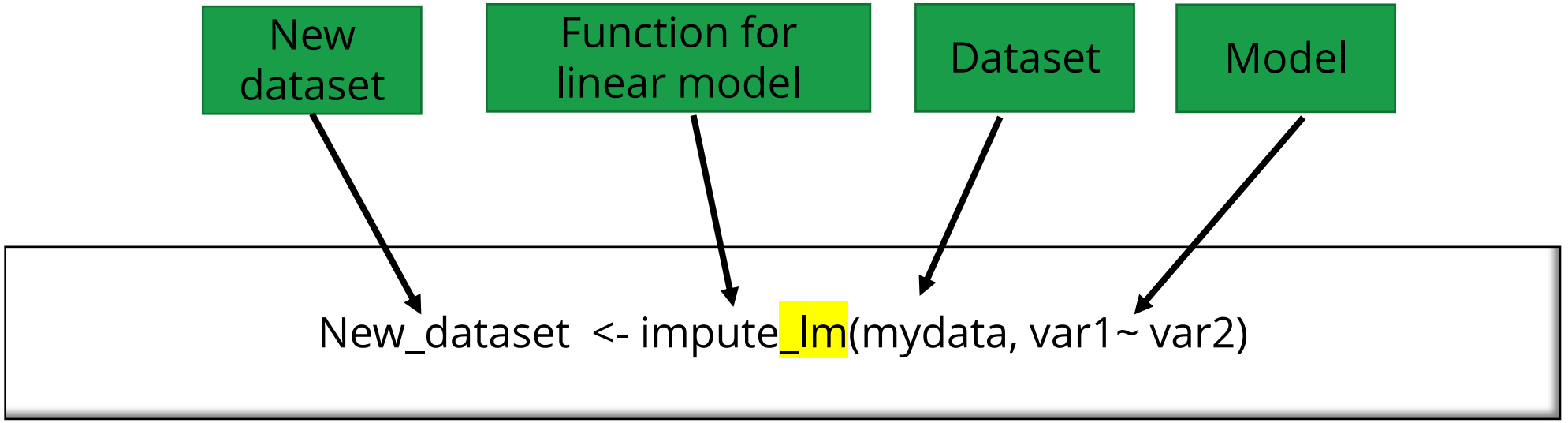
Other

- (groupwise) median imputation (optional random residual)
- Proxy imputation: copy another variable or use a simple transformation to compute imputed values.
- Apply trained models for imputation purposes.

General setup



Example linear regression, lm



Grouping data for imputation

- Use | in the formula argument to specify groups.

```
New_dataset  <- impute_lm(mydata, var1~ var2 | GROUPS )
```

Chaining imputation methods

Using the %>% operator from the popular magrittr allows for a very compact specification.

```
library(magrittr)

newdata<- mydata %>%
  impute_lm(var1 ~ var2) %>%
  impute_median(var2) %>%
  impute_cart(var3 ~ .)
```

Similar model for multiple variables

- imputation model for multiple variables at once.
- For example, to impute both var1 and var2 with a similar robust linear model, do the following.

```
newdata <- impute_rlm(mydata, var1 + var2 ~ var3)
```

Logging changes with «lumberjack»

- Easy logging of changes in data.
- Possible to study the effect of imputation
- Operator %>>%

```
library(lumberjack)
Logger<-cellwise$new(key="ID")

out <- mydata %>>%
  start_log() %>>%
  impute_lm(var1 ~ var2) %>>%
  dump_log(file="mylog.csv", stop=TRUE)
```

Example: Index of retail sales

```
#rette opp 1000-feil og setter de som har <lik> til missing for å kunne imputere
mod <- modifier(
  if (is.na(OMS)) OMS <- 0,
  if (is.na(NACE)) NACE <- "47111",
  if (is.na(NACE2)) NACE2 <- "47",
  if (OMS_FMND > 0 & OMS> 0 & 750 < OMS/OMS_FMND & OMS/OMS_FMND < 1400)  OMS <- OMS/1000,
  if (OMS > 0 & OMS == OMS_FAAR ) OMS <- NA,
  if (OMS > 0 & OMS == OMS_FMND) OMS <- NA
  )

logger <- cellwise$new(key="ID")

out<- doi %>>%
start_log(logger) %>>%
modify(mod) %>>%
impute_rlm(OMS ~ OMS_FMND +OMS_FAAR) %>>%
impute_rlm(OMS ~ OMS_FMND) %>>%
dump_log(file="minlog.csv", stop=TRUE)
log<-read.csv("minlog.csv")
dim(log)
head(log)
```

	step	time	srcref	expression	key	variable	old	new
	<int>	<fct>	<lgl>	<fct>	<dbl>	<fct>	<int>	<dbl>
1	1	2020-10-15 11:13:14 CEST	NA	modify(mod)	14219230025	OMS	474146	474.146
2	1	2020-10-15 11:13:14 CEST	NA	modify(mod)	14219230026	OMS	213740	213.740
3	1	2020-10-15 11:13:14 CEST	NA	modify(mod)	14219230027	OMS	484528	484.528
4	1	2020-10-15 11:13:14 CEST	NA	modify(mod)	14219230028	OMS	493670	493.670
5	1	2020-10-15 11:13:14 CEST	NA	modify(mod)	14219230029	OMS	529103	529.103
6	1	2020-10-15 11:13:14 CEST	NA	modify(mod)	14219230030	OMS	209617	209.617

Exercise 6:

- Exercise 6 is in the file : Exercises_day3.R
- Need to download R-packages:

✓dcmodify

✓simputation

Exercise 6 review

Summary

- Remember library( )
- Read in files: read_csv( ) read_dta( )
- New variable: mutate( )
- Select some rows: filter( )
- Summary: summarise( )
- Plot: ggplot( ), aes( ), geom_...( )
- Draw sample: sample_n(), sample_frac()
- Validate: validator(), confront(), summary()
- Rule based imputation: modifier(), modify()
- Model based imputation: impute_<model>()