Data Transformation with **dplyr 1.0**
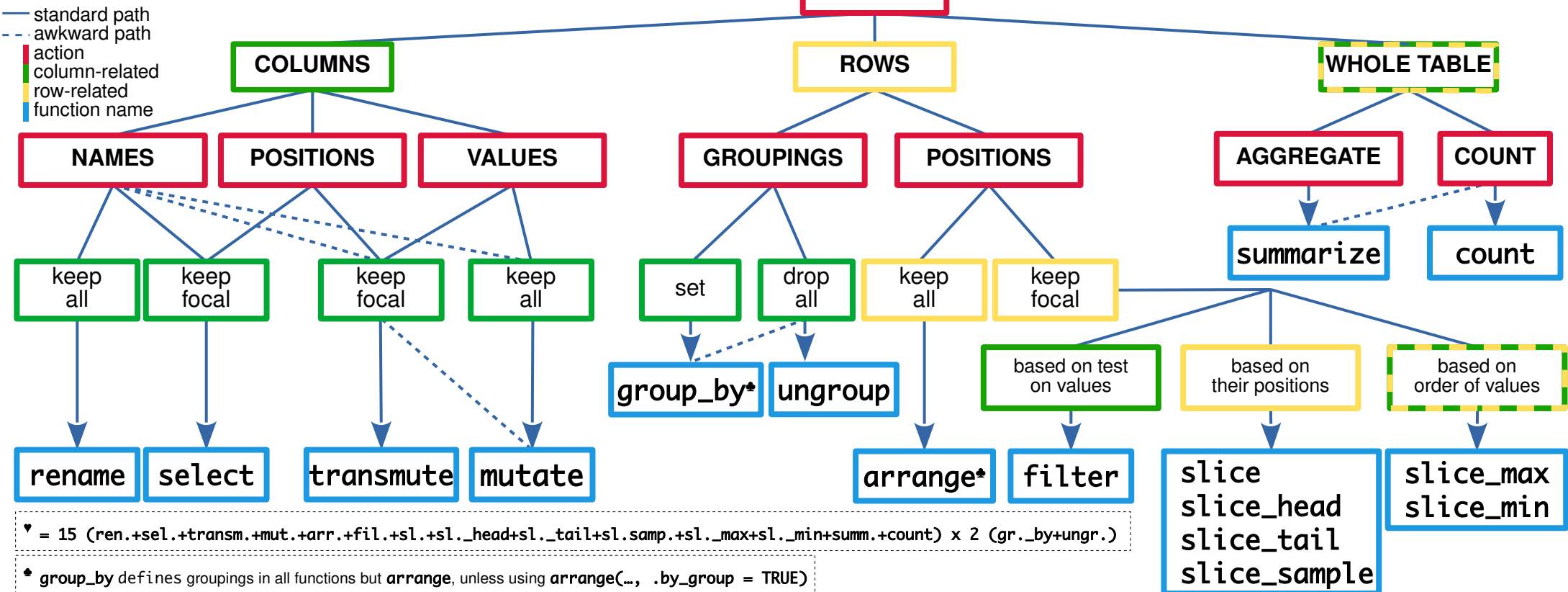
A guide to 30♥ behaviors applied to one tibble (`tbl`) or data.frame

© R Data Berlin

🐦 @rdataberlin https://github.com/courtiol/Rguides

dplyr

Legend:
- —— standard path
- – – – awkward path
- action (red)
- column-related (green)
- row-related (yellow)
- function name (blue)

MODIFY

COLUMNS — **ROWS** — **WHOLE TABLE**

COLUMNS
- **NAMES**
  - keep all
  - keep focal
- **POSITIONS**
  - keep focal
  - keep all
- **VALUES**

From NAMES:
- keep all → `rename`
- keep focal → `select`

From POSITIONS / VALUES:
- keep focal → `transmute`
- keep all → `mutate`

ROWS
- **GROUPINGS**
  - set
  - drop all
  - → `group_by`♣ / `ungroup`
- **POSITIONS**
  - keep all → `arrange`♣
  - keep focal → `filter`

WHOLE TABLE
- **AGGREGATE** → `summarize`
- **COUNT** → `count`

From summarize:
- based on test on values → `arrange`♣ / `filter`
- based on their positions → `slice` / `slice_head` / `slice_tail` / `slice_sample`
- based on order of values → `slice_max` / `slice_min`

♥ = 15 (ren.+sel.+transm.+mut.+arr.+fil.+sl.+sl._head+sl._tail+sl.samp.+sl._max+sl._min+summ.+count) x 2 (gr._by+ungr.)

♣ group_by defines groupings in all functions but **arrange**, unless using `arrange(…, .by_group = TRUE)`

SYNTAX: `tbl %>% verb()`

[font style varies to tease apart placeholders from true R_commands]

```
tbl %>% rename(new_name_col_X = old_name_col_X)

tbl %>% select(name_col_X, name_col_Y [+ std_op.♦], selection_helper♠("text"))

tbl %>% transmute(name_col_Z = fn‡(name_col_X))

tbl %>% mutate(name_col_Z = fn‡(name_col_X))

tbl %>% group_by(name_col_X, name_col_Y) %>% verb() %>% ungroup()

tbl %>% arrange(name_col_X, desc(name_col_Y))

tbl %>% filter(fn_test_vectorized(name_col_X), fn_test_vectorized(name_col_Y))

tbl %>% slice(row_indices); tbl %>% slice_head/tail/sample(number_rows_to_keep);

tbl %>% slice_min/max(name_col_X, n = nb_rows_to_keep [or prop = proportion_rows_to_keep])

tbl %>% summarize(name_col_Z = fn‡(name_col_X))

tbl %>% count(); tbl %>% count(name_col_X)
```

♦ standard operators may be used to combine (`c()`, `&`, `|`) or negate elements (`!`)

♠ selection helpers from pkg **tidyselect** may be used to select columns based on:
- column values → **where**(fn), e.g. fn = **is.numeric**
- column names → **starts_with**("text"), **ends_with**("text"), **contains**("text"), **matches**("regex"), **num_range**("text", min:max), **all_of**(vector_of_text), **any_of**(vector_of_text)
- column positions → **everything**(), **last_col**()

‡ if the function fn does not return a scalar or a nrow-long output, use **list**(fn()) to create a list column (i.e. for nesting the content); for creating multiple columns at once fn should return a *data.frame* or *tibble* and no name should be defined when calling the **dplyr** verb (i.e. ~~name_col_Z =~~; if a name is defined, the output will be nested); to unnest to content of a list column, use one of the functions provided by the pkg **tidyr** (e.g. **unnest_wider**)