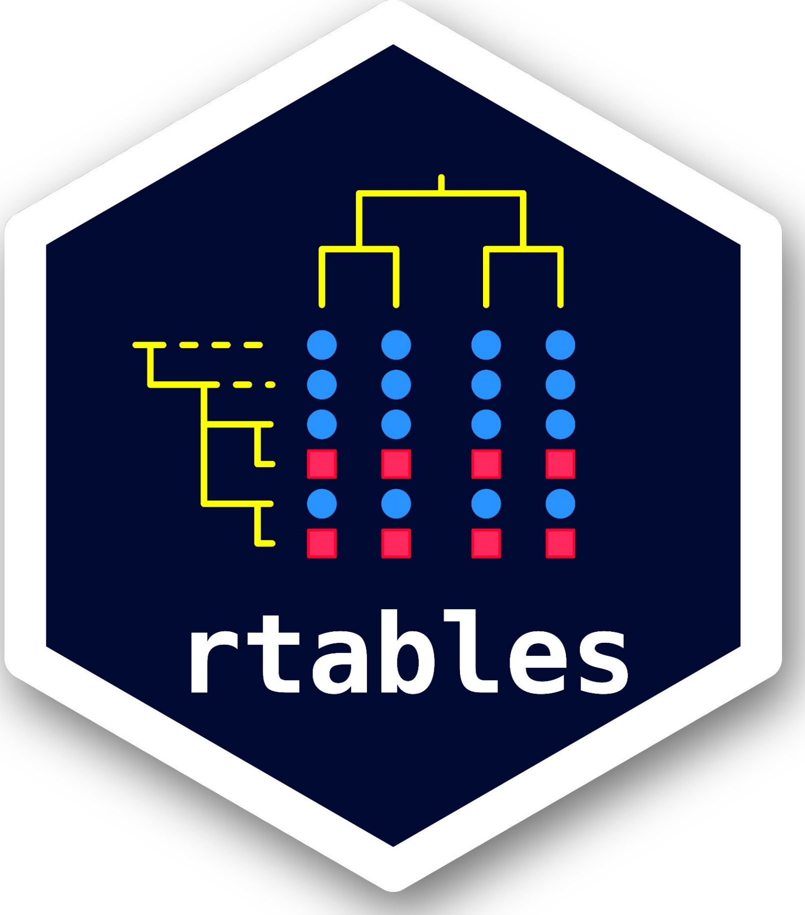
Creating Submission-Quality Clinical Trial Reporting Tables in R with `rtables`

R in Pharma Workshop 2022

October 26, 2022

Adrian Waddell, Gabriel Becker


rtables

What you will learn

- rtables fundamentals
- advanced tables generation
- exporting tables to various formats including
    - txt, pdf, rtf, html
    - Pagination
- using pharma specific convenience functions from the tern package

Housekeeping

- Slides are available here: https://tinyurl.com/3sx6xzj6
- RStudio Cloud has pre-configured Assignments
  - rtables is installed
  - Git repo is already cloned (url)
- Ask questions in slido: tinyurl.com/2p84puzf
- Course completion will result with a credily badge
- rtables pkgdown documentation: roche.github.io/rtables

we use rtables v0.5.3.1 for the workshop

Install it manually

```
cran_pkgs <- c(
  "htmltools", "magrittr", "broom", "car", "checkmate",
  "cowplot", "dplyr", "emmeans", "forcats", "ggplot2", "gridExtra",
  "gtable", "labeling", "lifecycle", "Rdpack", "rlang", "scales",
  "tibble", "tidyr", "r2rtf", "remotes"
)

install.packages(cran_pkgs)

remotes::install_github("insightsengineering/formatters")
remotes::install_github("Roche/rtables")
remotes::install_github("insightsengineering/tern")
```

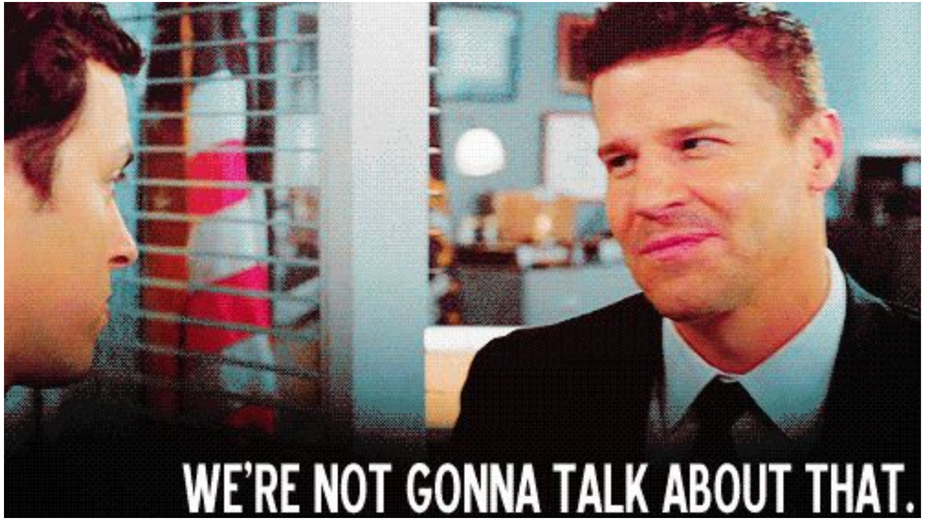
* rtables and formatters are also on CRAN

What is rtables

- rtables is open source R package to create and render a wide variety of tables
    - `install.packages("rtables")`
- Main authors are Gabriel Becker (maintainer) and Adrian Waddell
    - funded by Roche
- Can be used to create most standard tables used to analyze and report clinical trials data
- Sophisticated mechanism for cell value derivations
- Provides various formatting and rendering options

What about listings?

Prototype that uses `formatters` (`rtables` rendering machinery backend) can be found here: https://github.com/insightsengineering/rlistings



`rtables` Conceptual Model

0th Law of Computing (Statistical or Otherwise)

Let the computer do tasks that are

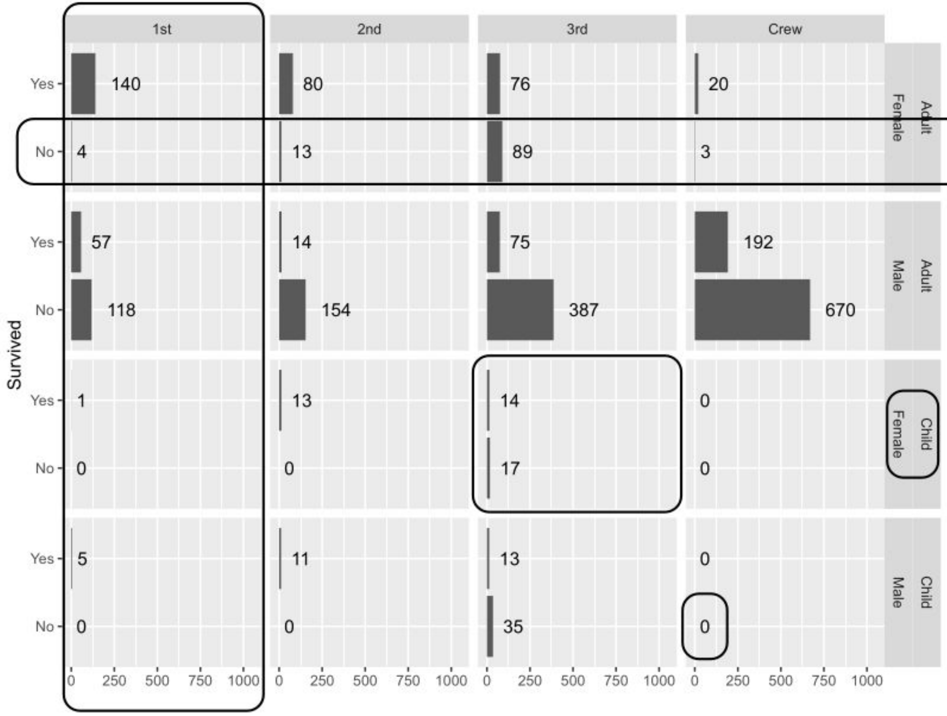
- Tedious
- Repetitive
- Human-error prone

That’s what it’s there for!

The First Step In Creating a Table

Calculating cell values, right?

Reporting Tables Are Faceted Data Visualizations



	1st	2nd	3rd	Crew
Adult				
Female				
Yes	140	80	76	20
No	4	13	89	3
Male				
Yes	57	14	75	192
No	118	154	387	670
Child				
Female				
Yes	1	13	14	0
No	0	0	17	0
Male				
Yes	5	11	13	0
No	0	0	35	0

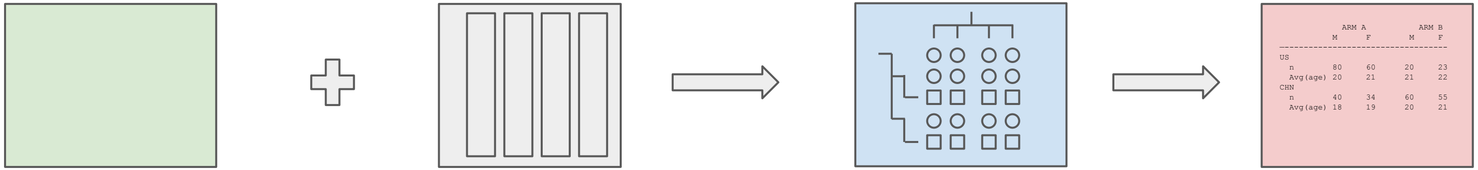
Imagine Manually Subsetting Facet Data When Using `ggplot2` (or `lattice`)

Subsetting data and calculating facet statistics

Humans

Computers

Basics of rtables



```
           ARM A            ARM B
         M       F        M       F
--------------------------------------
US
  n          80      60       20      23
  Avg(age)   20      21       21      22
CHN
  n          40      34       60      55
  Avg(age)   18      19       20      21
```

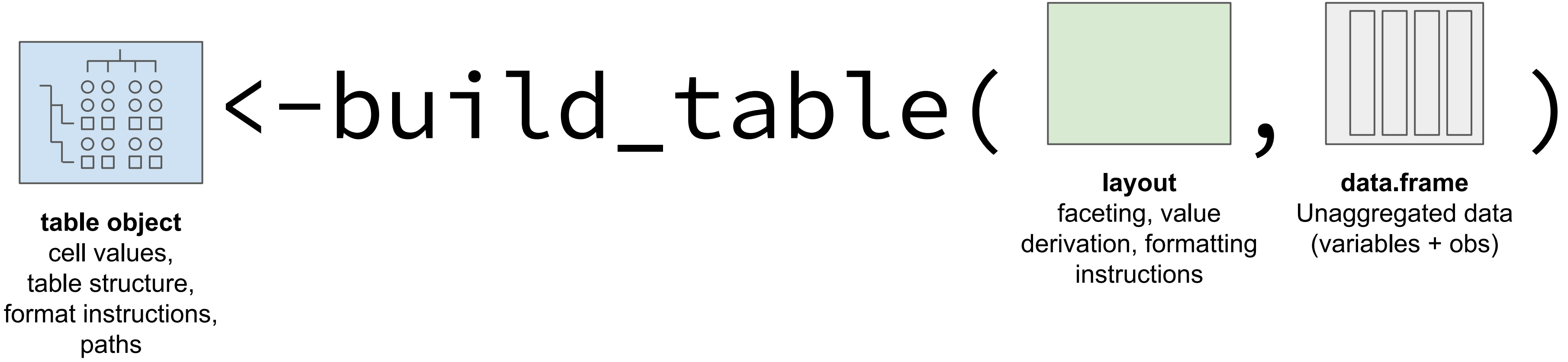
layout
faceting, value derivation, formatting instructions

data.frame
Unaggregated data (variables + obs)

table object
cell values, table structure, format instructions, paths

formatted table
such as ASCII, pdf, rtf

Basics of rtables



```
<-build_table( , )
```

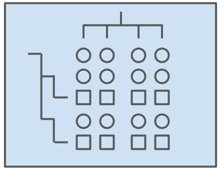
table object
cell values,
table structure,
format instructions,
paths

layout
faceting, value derivation, formatting instructions

data.frame
Unaggregated data (variables + obs)

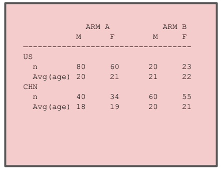
Basics of rtables

```
print(
```



```
)
```

table object
cell values,
table structure,
format instructions,
paths



```
             ARM A            ARM B
           M      F        M      F
------------------------------------
US
  n        80     60       20     23
  Avg(age) 20     21       21     22
CHN
  n        40     34       60     55
  Avg(age) 18     19       20     21
```

formatted table
here ASCII

Basics of rtables

```r
library(rtables)

lyt <- basic_table() |>
         split_cols_by("ARM") |>
         analyze("AGE", mean, format = "xx.xx")

tbl <- build_table(lyt, ex_adsl)

print(tbl)
```

Basics of rtables

```
library(rtables)

lyt <- basic_table() |>
           split_cols_by("ARM") |>
           analyze("AGE", mean, format = "xx.xx")

tbl <- build_table(lyt, ex_adsl)

print(tbl)
```

layout
faceting, value
derivation, formatting
instructions

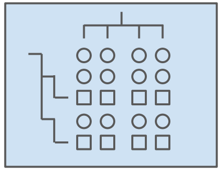
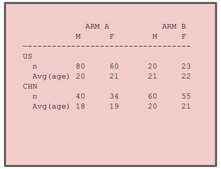
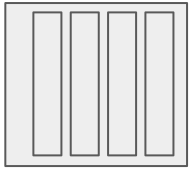


table object
cell values,
table structure,
format instructions,
paths



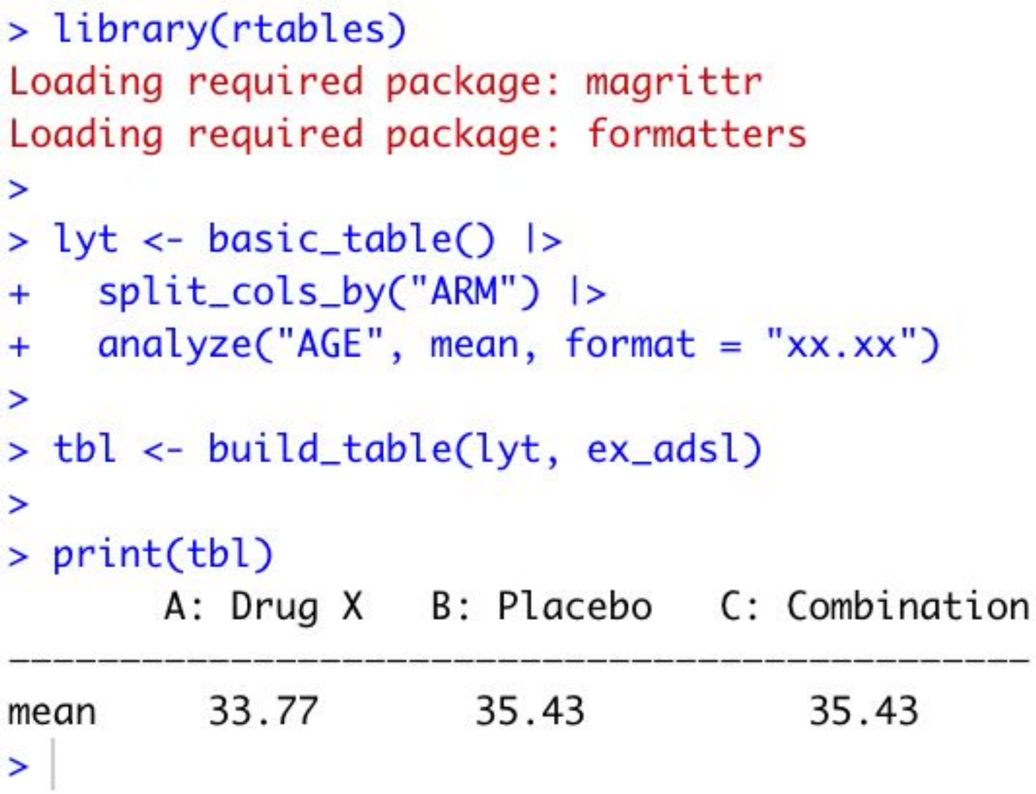
```
              ARM A           ARM B
            M      F        M      F
-----------------------------------------
US
  n         80     60       20     23
  Avg(age)  20     21       21     22
CHN
  n         40     34       60     55
  Avg(age)  18     19       20     21
```

formatted table
here ASCII



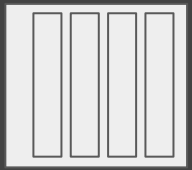
data.frame
Unaggregated data
(variables + obs)

Basics of rtables



```
> library(rtables)
Loading required package: magrittr
Loading required package: formatters
>
> lyt <- basic_table() |>
+   split_cols_by("ARM") |>
+   analyze("AGE", mean, format = "xx.xx")
>
> tbl <- build_table(lyt, ex_adsl)
>
> print(tbl)
       A: Drug X   B: Placebo   C: Combination
——————————————————————————————————————————————
mean     33.77       35.43          35.43
> 
```

exercise 01



data.frame
Unaggregated data
(variables + obs)

layout
faceting, value
derivation, formatting
instructions

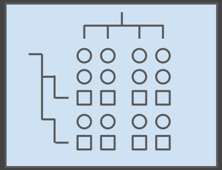
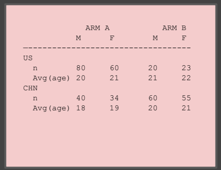


table object
cell values,
table structure,
format instructions,
paths



```
             ARM A          ARM B
           M      F       M      F
------------------------------------
US
  n        80     60      20     23
  Avg(age) 20     21      21     22
CHN
  n        40     34      60     55
  Avg(age) 18     19      20     21
```

formatted table
such as ASCII, pdf,
rtf

Building a Demographic Table end 2 end

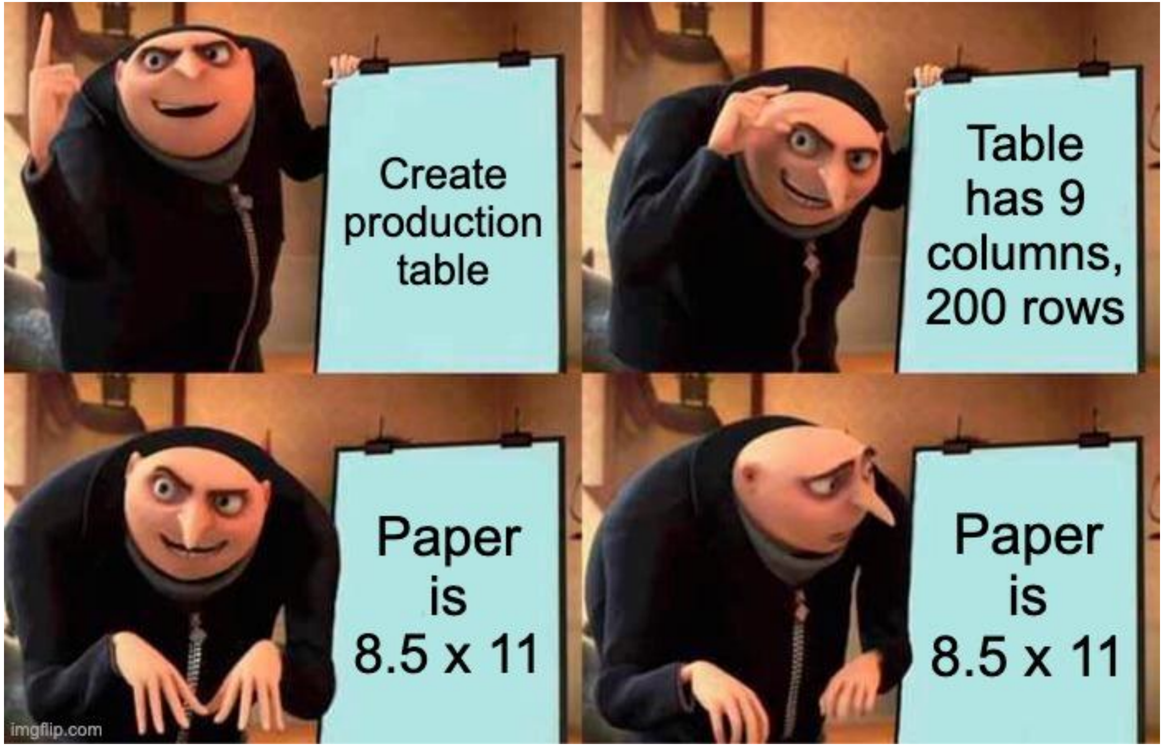
- Add number of subjects for each arm (column)
- Use variable labels
- Add titles, footer
- Analyze more than 1 variable
- Add "All Patients" column

exercise 02

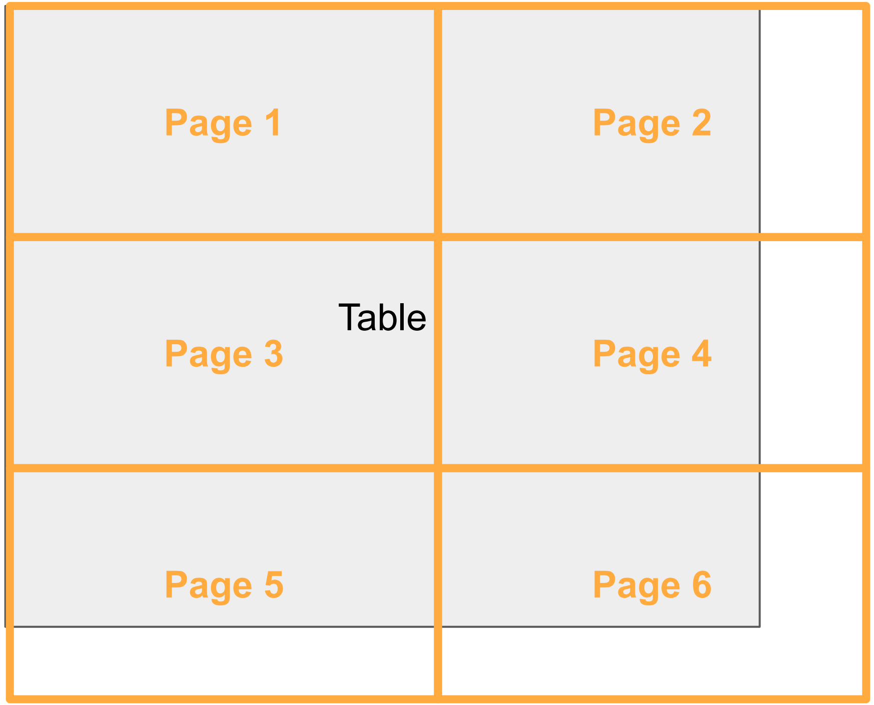
demographic table

exercise: add titles and footnotes

Pagination

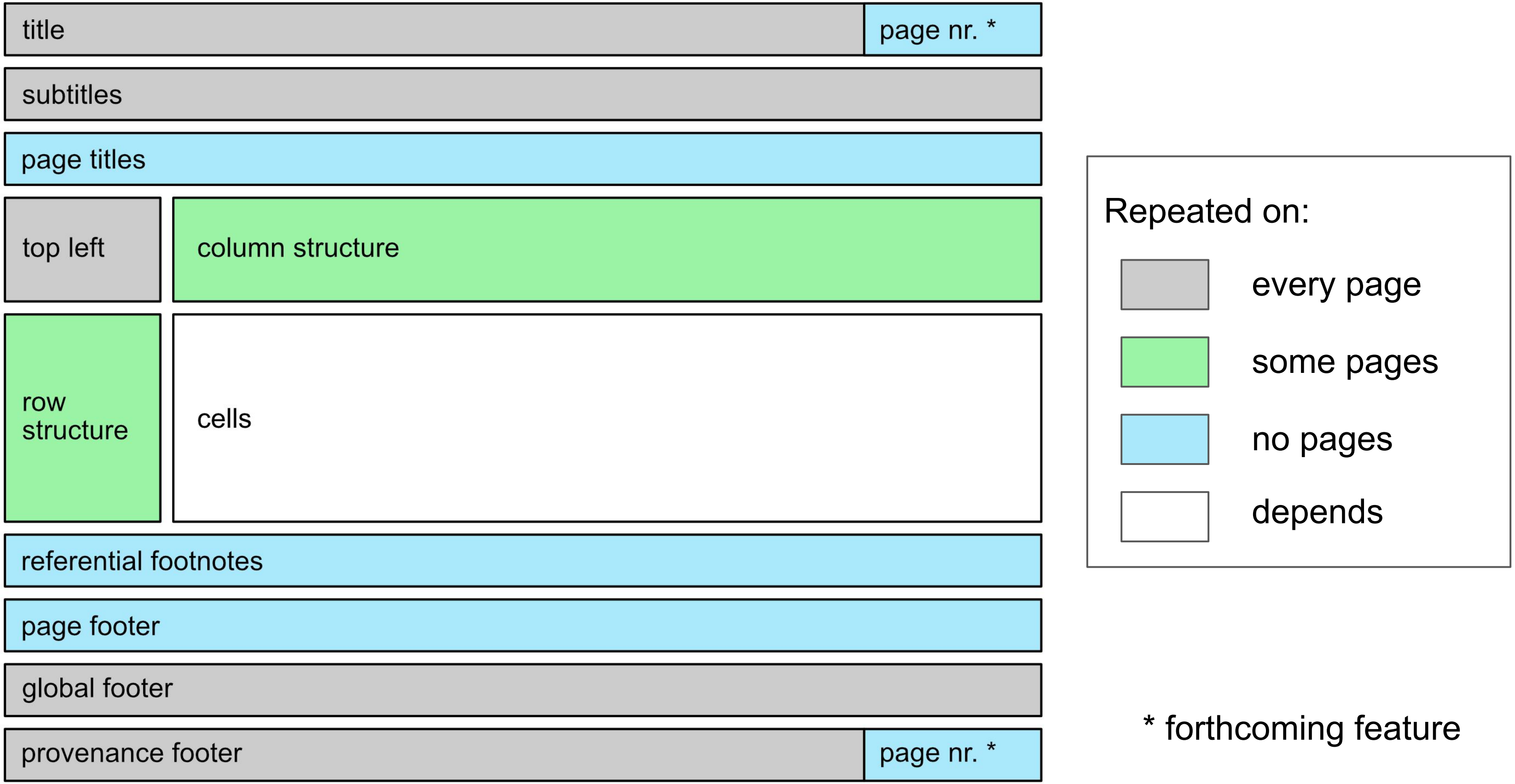

Create production table
Table has 9 columns, 200 rows
Paper is 8.5 x 11
Paper is 8.5 x 11
imgflip.com

Pagination Problem



Note some rows, columns and row names need to be repeated for context.

Context-Preserving Pagination



title

page nr. *

subtitles

page titles

top left

column structure

row structure

cells

referential footnotes

page footer

global footer

provenance footer

page nr. *

Repeated on:

▨ every page

🟩 some pages

🟦 no pages

⬜ depends

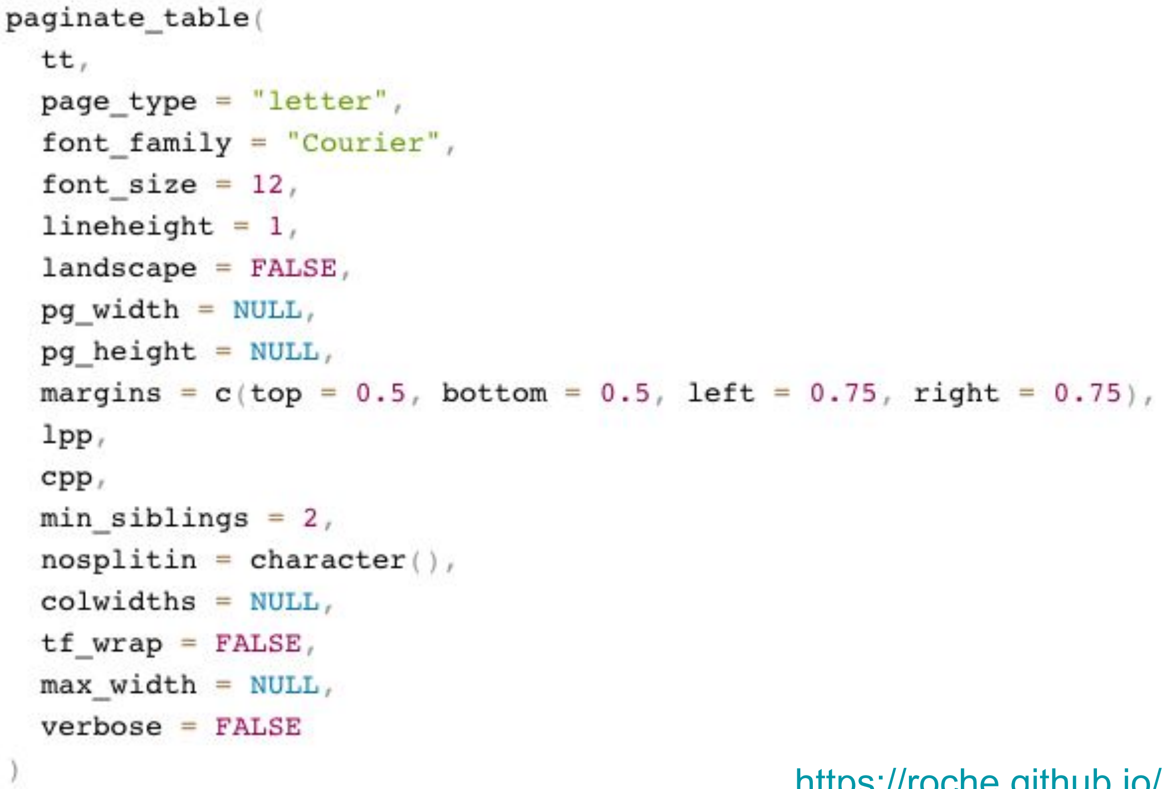
* forthcoming feature

Paginating tables

Pagination is built into the formal exporters (`export_as_txt`, `export_as_pdf`, `export_as_rtf`) and can also be invoked manually via `paginate_table.`

- Vertical and horizontal pagination
- Page & fontsize specification
- Automatic repetition of context rows (labels/summaries)
- Word wrapping of title and footer text

Pagination parametrization



```r
paginate_table(
  tt,
  page_type = "letter",
  font_family = "Courier",
  font_size = 12,
  lineheight = 1,
  landscape = FALSE,
  pg_width = NULL,
  pg_height = NULL,
  margins = c(top = 0.5, bottom = 0.5, left = 0.75, right = 0.75),
  lpp,
  cpp,
  min_siblings = 2,
  nosplitin = character(),
  colwidths = NULL,
  tf_wrap = FALSE,
  max_width = NULL,
  verbose = FALSE
)
```

https://roche.github.io/rtables/main/reference/paginate.html

exercise 03

paginate demographic table

Render the table

Multiple formats are supported

- txt (via toString)
- pdf via toString & grid graphics
- RTF via r2rtf
- html with as_html

exercise 04

render demographic table

that's it, you can email this table to your stakeholders*!

* make sure to use real data

It gets better…

`tern` in an open source R package (from Roche) that implements analytic components of standard clinical trial reporting tables for use within the rtables framework. With `rtables` and `tern` it is possible to make most (>200) of Roche’s standard tables.

```
library(rtables)
library(tern)

vars <- c("RACE", "SEX")
var_lbls <- var_labels(ex_adsl)[vars]

lyt <- basic_table() |>
  split_cols_by(var = "ARM", split_fun = add_overall_level("All Patients")) |>
  summarize_vars(
    vars = vars,
    var_labels = var_lbls
  )

build_table(lyt, ex_adsl)
```

It gets better…

tern in an open source R package (from Roche) that implements analytic components of standard clinical trial reporting tables for use within the rtables framework. With `rtables` and `tern` it is possible to make most (>200) of Roche’s standard tables.

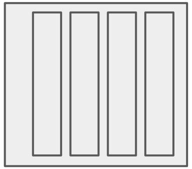
```
library(rtables)
library(tern)

vars <- c("RACE", "SEX")
var_lbls <- var_labels(ex_adsl)[vars]

lyt <- basic_table() |>
  split_cols_by(var = "ARM", split_fun = add_overall_level("All Patients")) |>
  summarize_vars(
    vars = vars,
    var_labels = var_lbls
  )

build_table(lyt, ex_adsl)
```

exercise 05

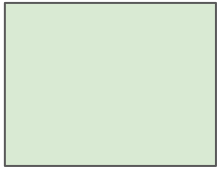
demographic table revisited

now let’s get cracking

So whats next…



data.frame
Unaggregated data
(variables + obs)



layout
faceting, value
derivation, formatting
instructions

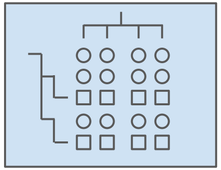
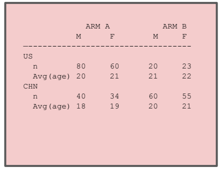


table object
cell values,
table structure,
format instructions,
paths



```
              ARM A            ARM B
            M      F        M      F
-------------------------------------
US
  n         80     60       20     23
  Avg(age)  20     21       21     22
CHN
  n         40     34       60     55
  Avg(age)  18     19       20     21
```

formatted table
such as ASCII, pdf,
rtf

Layouts declare tables (pre-data)

Table layouts

- are declared pre-data (symbolically describe the table structure)
- declare data facets
- contain the cell value derivation
  - Via `analyze()` and `summarize_row_groups()`
- every layout starts with `basic_table()`

exercise 10

create a table from basic_table() layout

Deriving cell values with `analyze()`

```
lyt <- basic_table() |>
   analyze("AGE")

build_table(lyt, ex_adsl)
```

```
        all obs
───────────────
Mean     34.88
```

```
fivenum_afun <- function(x) {
  in_rows(n = sum(!is.na(x)),
          "mean (sd)" = c(mean(x), sd(x)),
          median = median(x),
          "min - max" = range(x),
          .formats = c(n = "xx",
                       "mean (sd)" = "xx.x (xx.x)",
                       median = "xx.x",
                       "min - max" = "xx.x - xx.x"))
}

lyt2 <- basic_table() %>% analyze("AGE", fivenum_afun)

build_table(lyt2, ex_adsl)
```

```
                all obs
───────────────────────────
n                 400
mean (sd)     34.9 (7.4)
median           34.0
min - max     20.0 - 69.0
```

exercise 06

analyze() basics

Analysis - cell value derivation

So far we have seen how layouts are used to define facets.

- Analyses define how the data facet should be summarized and displayed
- The two main analyses functions are
  - analyze
  - summarize_row_groups
- An analysis can return cells for multiple rows with in_rows()
- Cell value formatting can be done with rcell, and the various format arguments

exercise 07

Cell Value Formatting

Analyze Calls

- analyze calls can be called sequentially
- analyze can be applied to multiple variables

They are equivalent.

exercise 08

multiple analyze() calls

Analysis Functions - Additional Arguments

- When deriving count & percentages one needs access to the column population N

- Analysis functions can optionally accept a number of arguments:
  - `.N_col` for column count
  - `.N_total` for total count
  - `.spl_context` for row-faceting context (see `?spl_context`)
  - `.var` for the name of the variable being analyzed
  - And others (see `?analyze`)

very advanced topic

Percentages

```
pct_afun <- function(x, .N_col) {
  rcell(
     sum(!is.na(x)) * c(1, 1/.N_col),
     format = "xx (xx.x%)"
  )
}

lyt <- basic_table() |>
    analyze("AGE", pct_afun)

build_table(lyt, ex_adsl)
```

```
               all obs
------------------------
pct_afun   400 (100.0%)
```

exercise 09

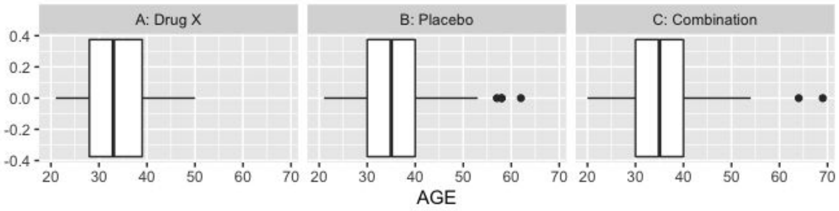
```
length(x) * c(1, 1/N)
```

Declaring Facets

- `split_rows_by()` (and siblings) add row faceting structure
- `split_cols_by()` (and siblings) add column faceting structure
- Column and row facet structure declared independently
  - As in `facet_grid(rows = , cols = )`

Column Faceting - ggplot2 and rtables

```
ggplot(ex_adsl, mapping = aes(x = AGE)) +
  geom_boxplot() +
  facet_grid(cols = vars(ARM))
```

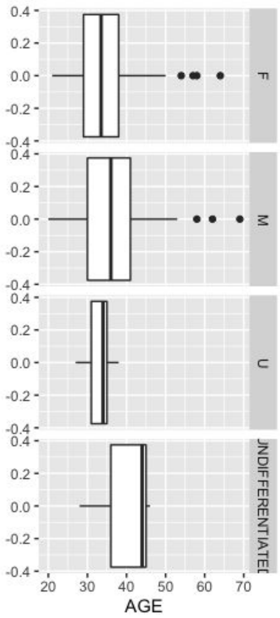


```
lyt <- basic_table() |>
  split_cols_by("ARM") |>
  analyze("AGE", range, format = "xx.xx - xx.xx")

build_table(lyt, ex_adsl)
```

```
               A: Drug X       B: Placebo     C: Combination
------------------------------------------------------------
range       21.00 - 50.00    21.00 - 62.00    20.00 - 69.00
```

Row Faceting - ggplot2 and rtables

```
ggplot(ex_adsl, mapping = aes(x = AGE)) +
  geom_boxplot() +
  facet_grid(rows = vars(SEX))
```

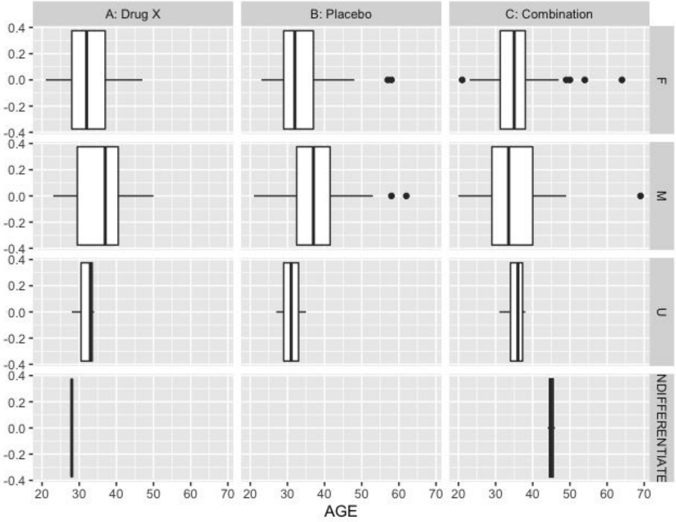


```
lyt2 <- basic_table() |>
  split_rows_by("SEX") |>
  analyze("AGE", range,
     format = "xx.xx - xx.xx")

build_table(lyt2, ex_adsl)
```

```
                           all obs
————————————————————————————————————
F
  range                21.00 - 64.00
M
  range                20.00 - 69.00
U
  range                27.00 - 38.00
UNDIFFERENTIATED
  range                28.00 - 46.00
```

Grid Faceting - `ggplot2` and `rtables`

```
ggplot(ex_adsl, mapping = aes(x = AGE)) +
    geom_boxplot() +
  facet_grid(rows = vars(SEX),
             cols = vars(ARM))
```



```
lyt3 <- basic_table() |>
  split_cols_by("ARM") |>
  split_rows_by("SEX") |>
  analyze("AGE", range, format = "xx.xx - xx.xx")

build_table(lyt3, ex_adsl)
```

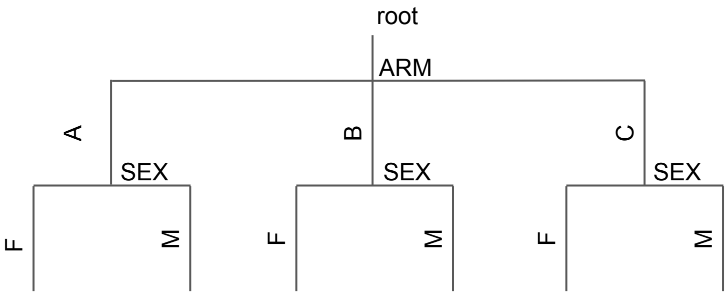
```
                   A: Drug X       B: Placebo     C: Combination
————————————————————————————————————————————————————————————————
F
  range          21.00 - 47.00   23.00 - 58.00   21.00 - 64.00
M
  range          23.00 - 50.00   21.00 - 62.00   20.00 - 69.00
U
  range          28.00 - 34.00   27.00 - 35.00   31.00 - 38.00
UNDIFFERENTIATED
  range          28.00 - 28.00    Inf - -Inf     44.00 - 46.00
```

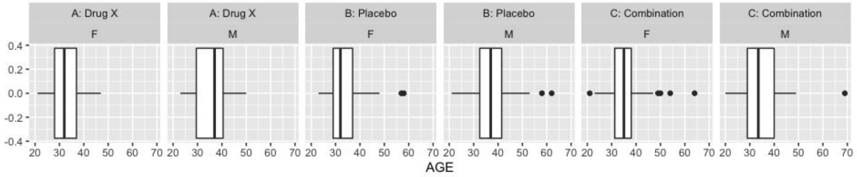
exercise 11

faceting data in ggplot2 and rtables

Nested Faceting Structure

Consecutive splits give nested facet structure, same as giving multiple variables in one dim to `facet_grid()`





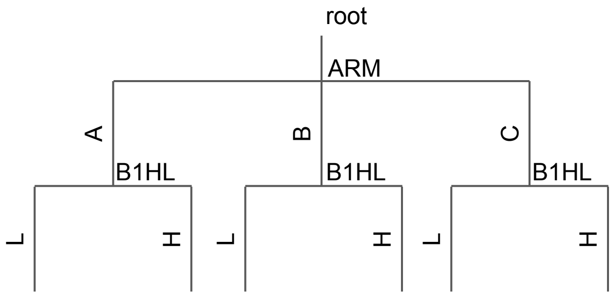
```
                A: Drug X                   B: Placebo                 C: Combination
                F             M             F             M             F             M
———————————————————————————————————————————————————————————————————————————————————————————
range   21.0 - 47.0   23.0 - 50.0   23.0 - 58.0   21.0 - 62.0   21.0 - 64.0   20.0 - 69.0
```

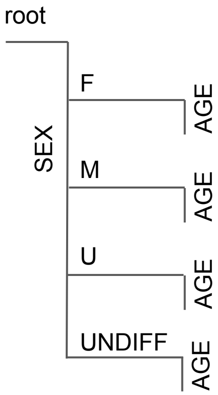
exercise 12

Nested Splits

Sneak peak into table objects

```
lyt <- basic_table() |>
  split_cols_by("ARM") |>
  split_cols_by("B1HL") |>
  split_rows_by("SEX") |>
  analyze("AGE", function(x) "")

tbl <- build_table(lyt, ex_adsl3)

col_paths_summary(tbl)
row_paths_summary(tbl)
```





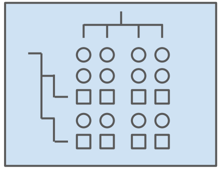


table object
numbers, strings
paths

exercise 13

Table Structure

A sneak peak into

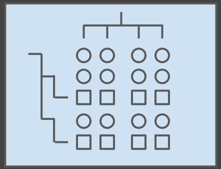


table object
numbers, strings
paths

Split Functions - Generalizing Faceting

- By default split_*_by generate faceting which partitions data based on a categorical variable
  - Same as faceting in `ggplot2`, `lattice`
- We can control which facet panes are generated via *split functions*
  - Split functions: `drop_split_levels`
  - Split function Factories: `remove_split_levels(excl=)`, `trim_levels_in_group(innervar =)`, `add_combo_levels(combosdf =)`, **`add_overall_col(valname =)`**

exercise 14

Split Functions

Custom Split Functions

Facets: We Are Partitions of Data Based on Cat. Variables

`rtables:`

Custom Split Functions

Custom split functions give us full control over the faceting process during table creation

Custom split functions can do 3 things:

- Preprocess incoming (parent) facet data
  - `drop_split_levels()` - drops empty factor levels in var *before splitting*
- Override the core data -> split data process
  - We almost never need to do this, and it’s currently only fully supported in row space.
- Post-process the set of generated groups
  - `add_combo_levels()` - does normal splitting then combines the resulting groups to form additional new facets
- See documentation for `custom_split_funs`

drop_split_levels()

```
drop_split_levels
function (df, spl, vals = NULL, labels = NULL, trim = FALSE)
{
    var <- spl_payload(spl)
    df2 <- df
    df2[[var]] <- factor(df[[var]])
    .apply_split_inner(spl, df2, vals = vals, labels = labels,
        trim = trim)
}
```

Preprocess incoming data

Now split as normal

double_trouble() split function*

```
double_trouble <- function(df, spl, vals, labels, trim) {
    ret <- do_base_split(spl, df, vals = vals,
                         labels = labels, trim = trim)
    ret$datasplit <- c(ret$datasplit, tail(ret$datasplit, 1))
    ret$values <- c(ret$values, tail(ret$values, 1))
    ret$labels <- c(ret$labels, tail(ret$labels, 1))
    ret$extras <- c(ret$extras, tail(ret$extras, 1))
    ret
}
```

First split as normal

Now add the last column again

```
lyt <- basic_table() |>
   split_cols_by("ARM", split_fun = double_trouble) |>
   analyze("AGE")

build_table(lyt, ex_adsl)
```

```
       A: Drug X   B: Placebo   C: Combination   C: Combination
———————————————————————————————————————————————————————————————
Mean     33.77       35.43          35.43            35.43
```

* Note this is a very silly custom split function …

exercise 15

Custom Split Functions

Various layout topics

- Show patient population per column (N=xx)
- Names vs Labels
  - Variable names vs variable labels
  - e.g. ARMCD & ARM, PARAMCD & PARAM

exercise 16

Various Layout Topics

Multivariable Splits

Certain table types call for columns to be variables, rather than groups of data.

- We declare this facet structure via `split_cols_by_multivar(vars)`
    - NOTE: Unlike data-based variable splits, we know exactly how many facets this will generate at declaration time
- We declare cell-value derivations in this case via `analyze_colvars`
    - Here we (can) give a **list** of analysis functions (one for each var column)

```
lyt <- basic_table() %>%
   split_cols_by_multivar(c("RACE", "AGE"), varlabels = c("Ethn. Present", "Ave. Age")) %>%
    analyze_colvars(afun = list(function(x) length(unique(x)),
                                function(x) rcell(mean(x), format = "xx.x")))
```

```
   Ethn. Present   Ave. Age
---------------------------
         6           34.9
```

exercise 17

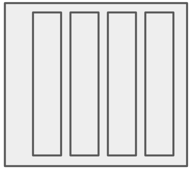
Multivar splits

The main points to take away by now are …

- rables is a sophisticated end 2 end table framework
- tables are faceted visualizations
- rtables tables are created with layouts and data
- layouts declare facets (split_* functions), analyses (analyze function)
- you can read the following code and predict the table structure:

```
lyt <- basic_table() |>
  split_cols_by("ARM") |>
  split_cols_by("SEX") |>
  split_rows_by("STRATA1") |>
  analyze("AGE", range, format = "xx.xx - xx.xx")

build_table(lyt, ex_adsl)
```

So whats next…



data.frame
Unaggregated data
(variables + obs)

layout
faceting, value
derivation, formatting
instructions

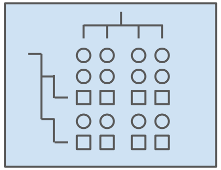
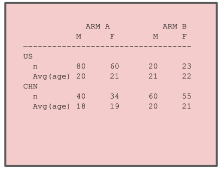


table object
cell values,
table structure,
format instructions,
paths



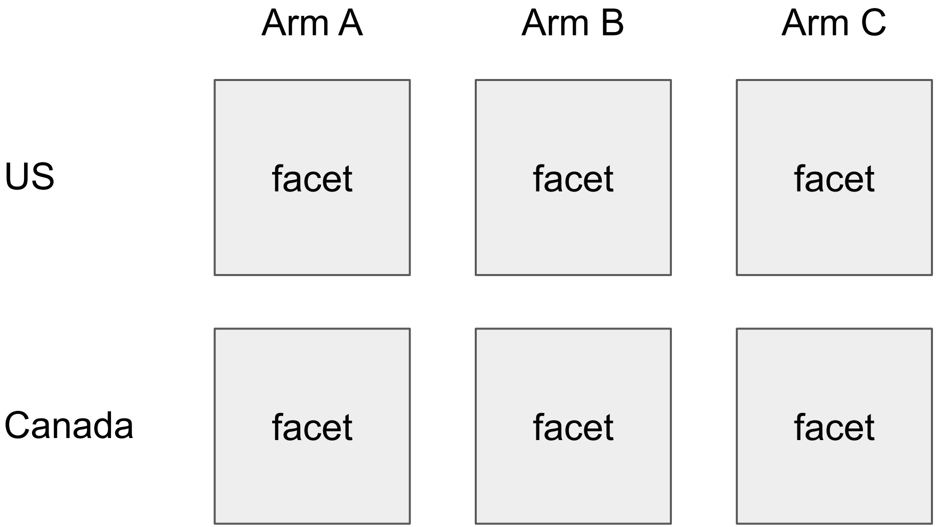
```
             ARM A              ARM B
           M        F        M        F
--------------------------------------------
US
  n        80       60       20       23
  Avg(age) 20       21       21       22
CHN
  n        40       34       60       55
  Avg(age) 18       19       20       21
```

formatted table
such as ASCII, pdf,
rtf

We have one more topic to cover for the layouts section

Analyze revisited

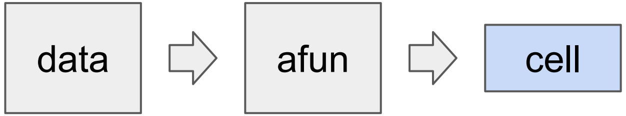
- The analysis function assigned to the argument **`afun`** in **`analyze()`** takes as input facet data and is expected to return **one or more cell values** via **`in_rows()`**

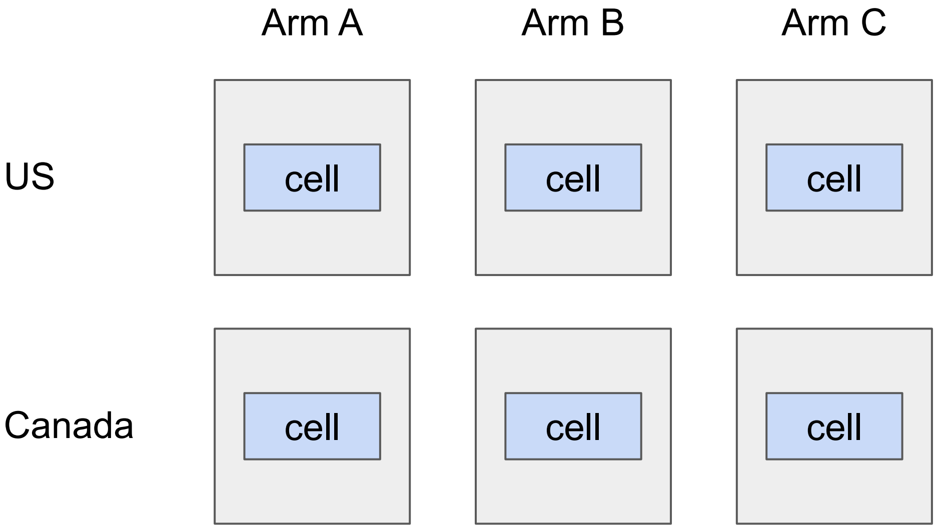


	Arm A	Arm B	Arm C
US	facet	facet	facet
Canada	facet	facet	facet

Analyze revisited

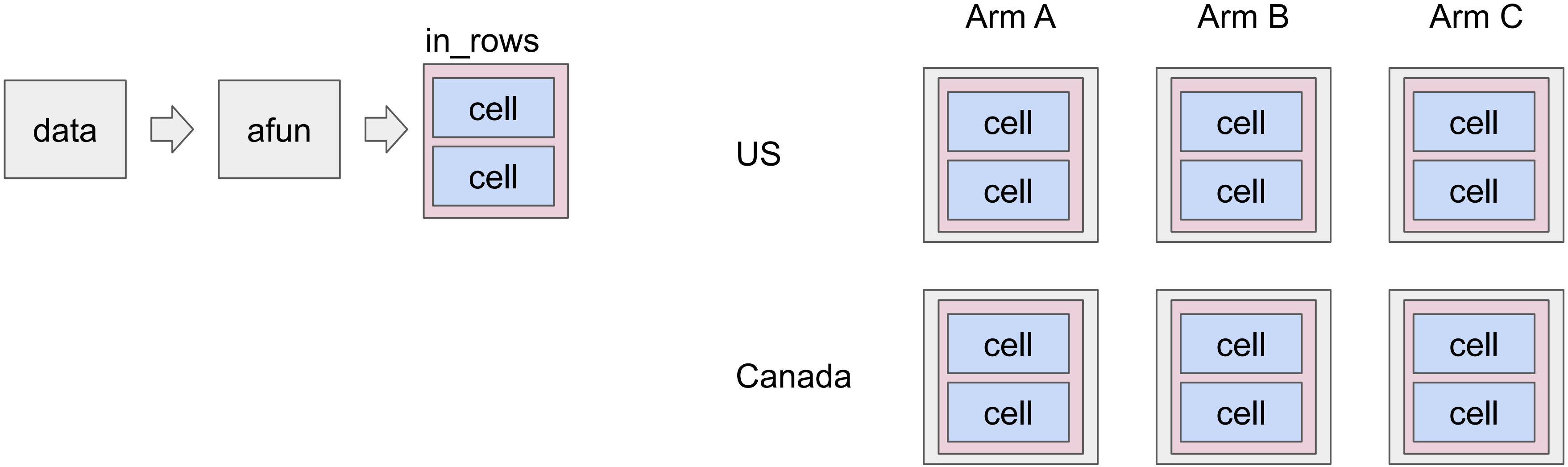
- The analysis function assigned to the argument **`afun`** in **`analyze()`** takes as input facet data and is expected to return **one or more cell values** via **`in_rows()`**





Analyze revisited

- The analysis function assigned to the argument `afun` in `analyze()` takes as input facet data and is expected to return **one or more cell values** via `in_rows()`



exercise 18

analyze, cell groups, rows

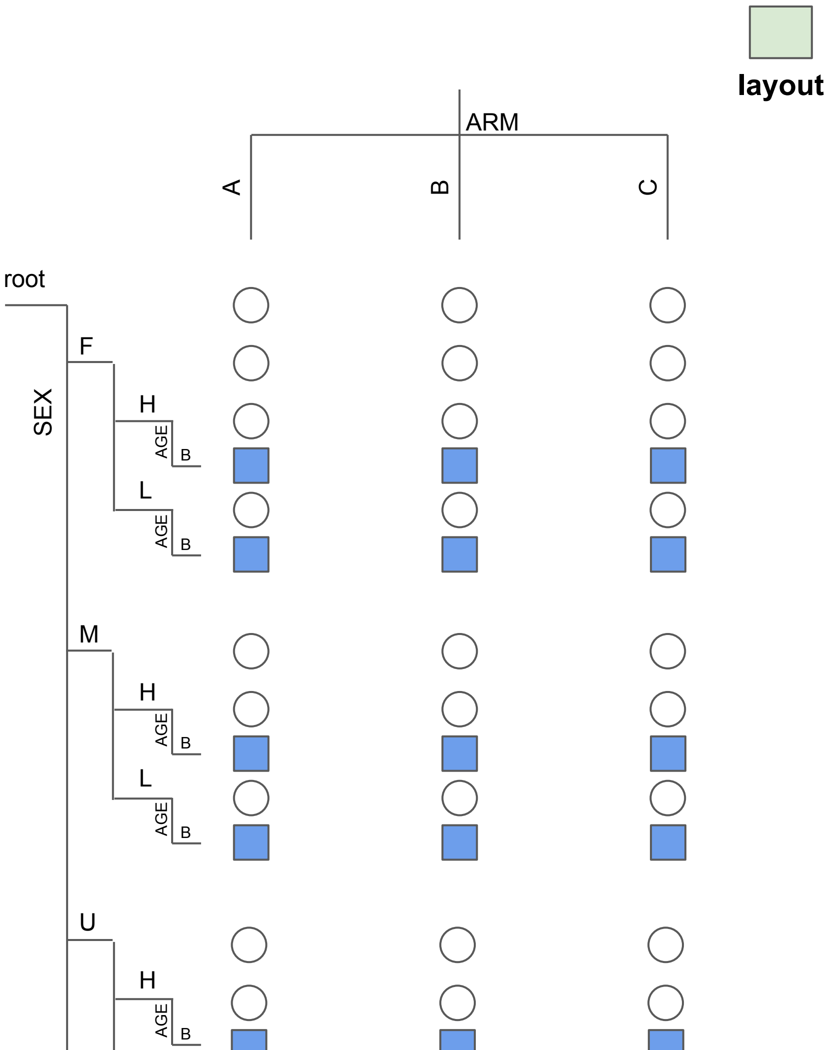
🟩 **layout**

Group summaries

```
lyt <- basic_table() |>
  split_cols_by("ARM") |>
  split_rows_by("SEX") |>
  split_rows_by("B1HL") |>
  analyze("AGE", \(x) list(B = "a"))

build_table(lyt, ex_adsl3)
```

Group summaries

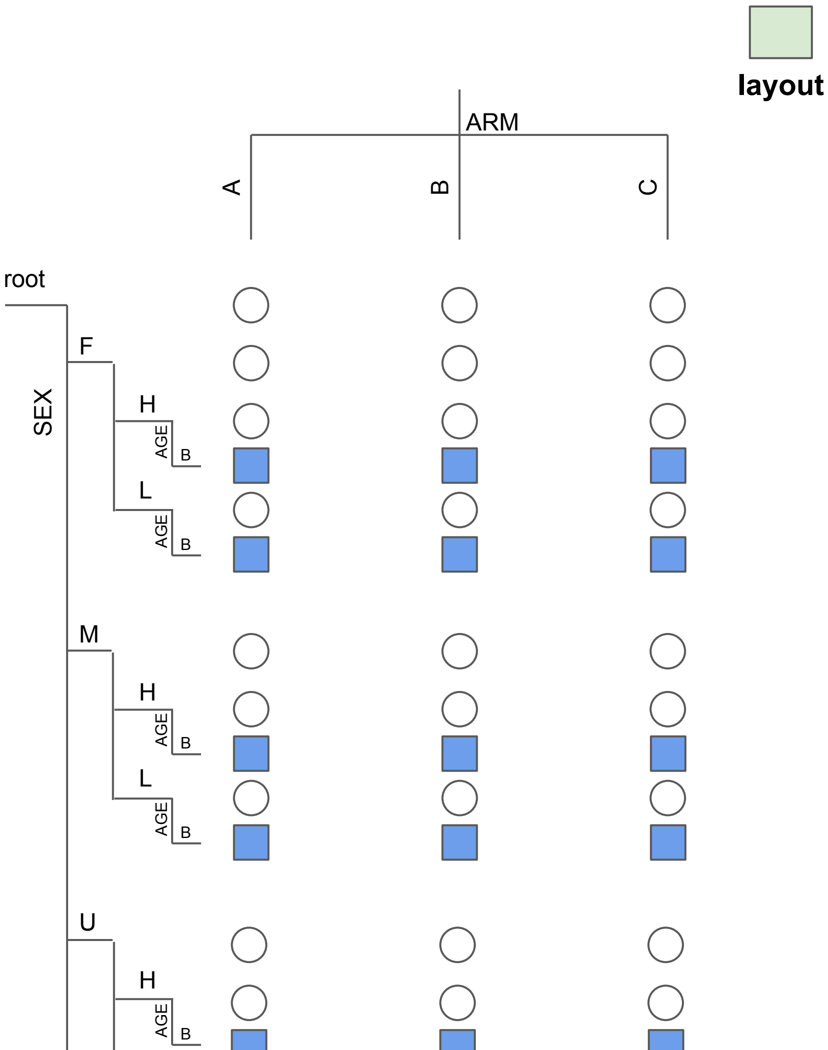
```
lyt <- basic_table() |>
  split_cols_by("ARM") |>
  split_rows_by("SEX") |>
  split_rows_by("B1HL") |>
  analyze("AGE", \(x) list(B = "a"))

build_table(lyt, ex_adsl3)
```



Group summaries

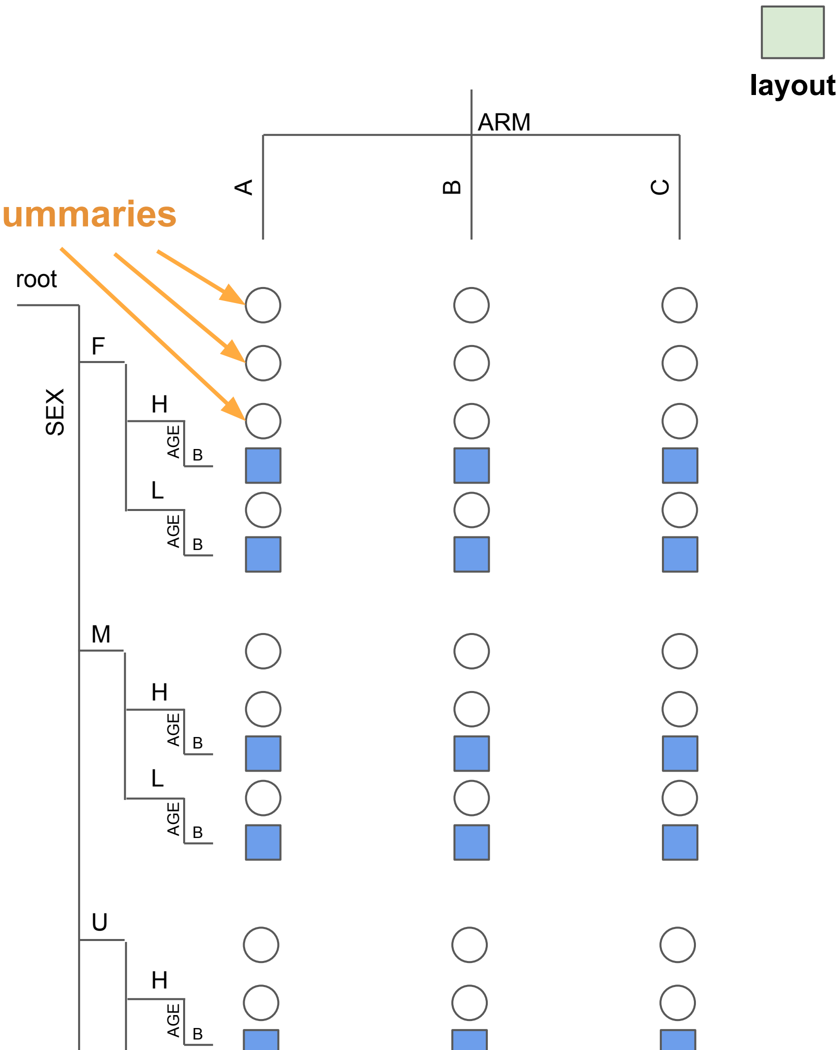
```
lyt <- basic_table() |>
  split_cols_by("ARM") |>
  split_rows_by("SEX") |>
  split_rows_by("B1HL") |>
  analyze("AGE", \(x) list(B = "a"))

build_table(lyt, ex_adsl3)
```

Note, `analyze` can return multiple rows with `in_rows()`



Group summaries

```
lyt <- basic_table() |>
  split_cols_by("ARM") |>
  split_rows_by("SEX") |>
  split_rows_by("B1HL") |>
  analyze("AGE", \(x) list(B = "a"))

build_table(lyt, ex_adsl3)
```

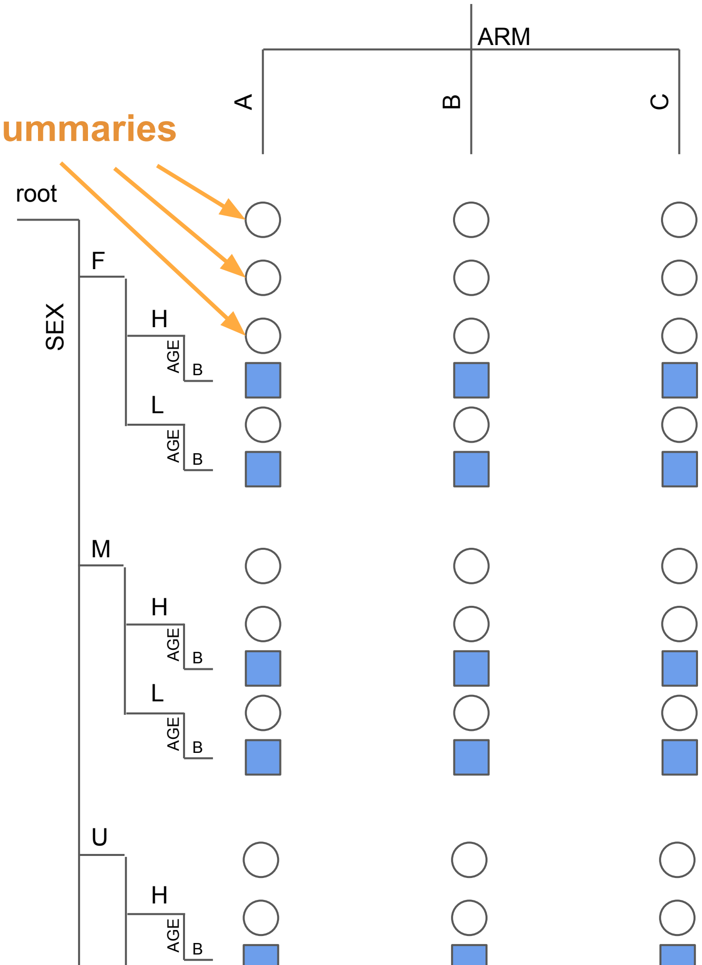


🟩 **layout**

Group summaries

3 levels of group summaries

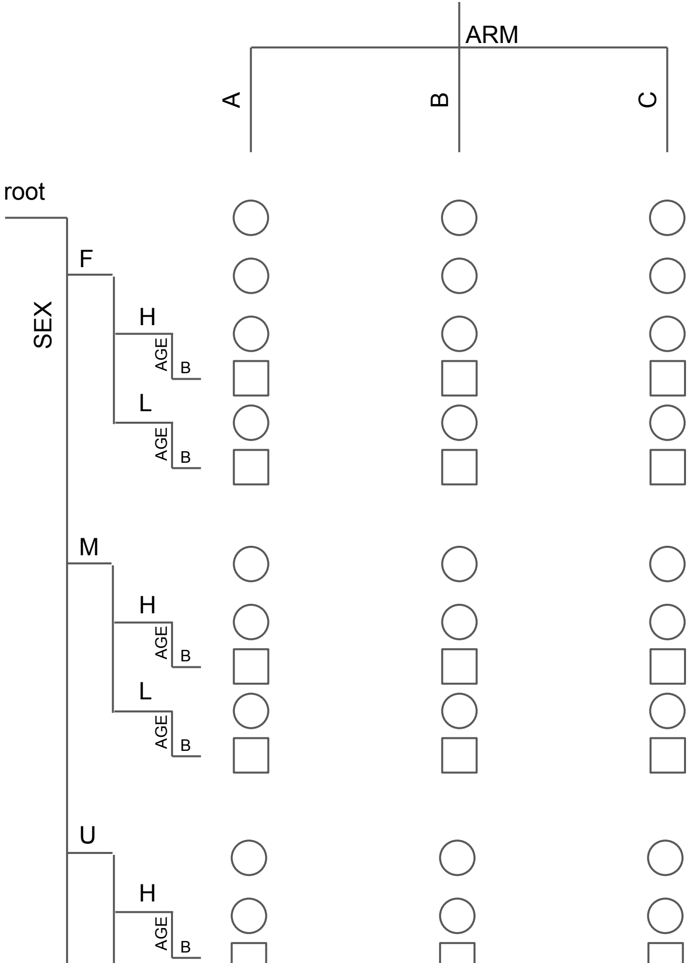
```
lyt <- basic_table() |>
  split_cols_by("ARM") |>
  split_rows_by("SEX") |>
  split_rows_by("B1HL") |>
  analyze("AGE", \(x) list(B = "a"))

build_table(lyt, ex_adsl3)
```



🟩 layout

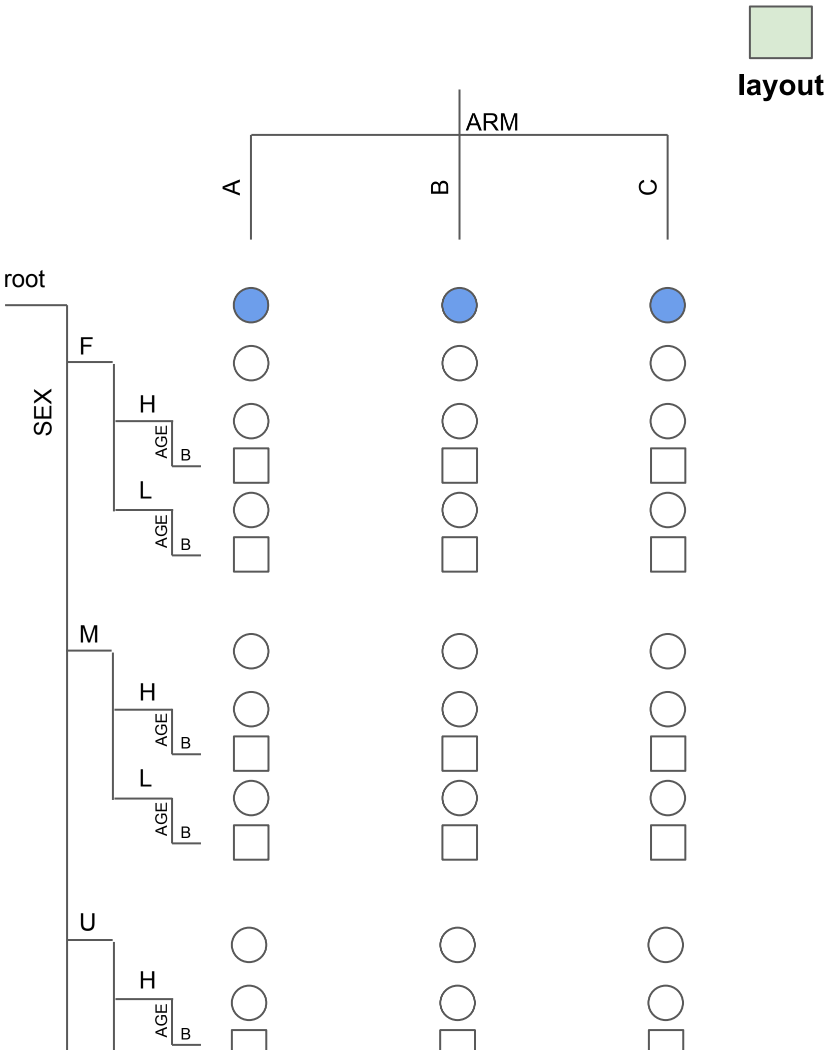
Group summaries

```
lyt <- basic_table() |>
  split_cols_by("ARM") |>
  split_rows_by("SEX") |>
  split_rows_by("B1HL") |>
  analyze("AGE", \(x) list(B = "a"))

build_table(lyt, ex_adsl3)
```



Group summaries

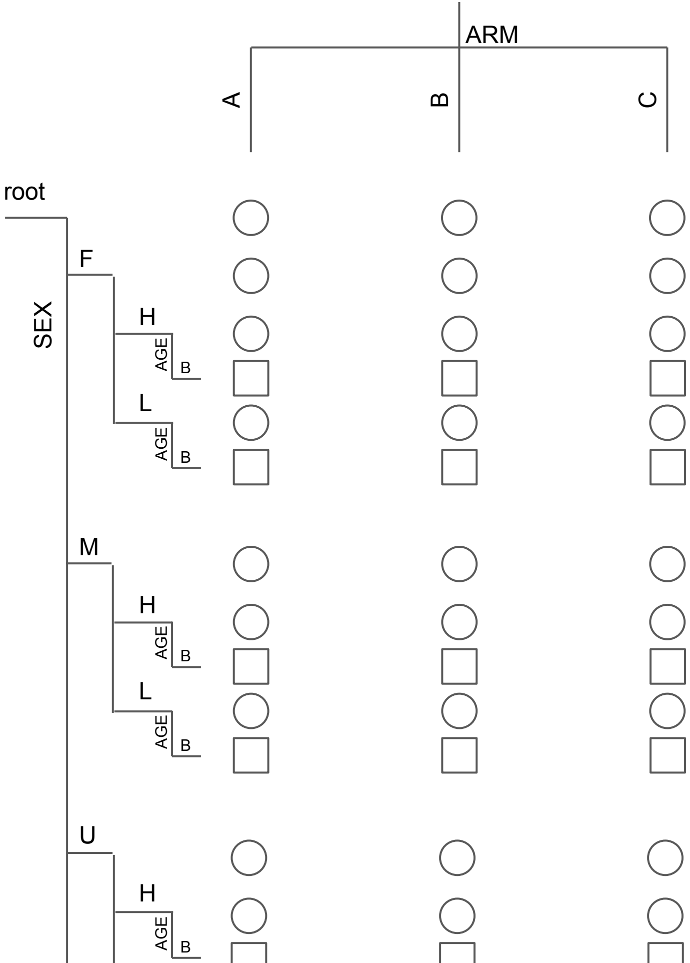
```
lyt <- basic_table() |>
  split_cols_by("ARM") |>
  summarize_row_groups() |>
  split_rows_by("SEX") |>
  split_rows_by("B1HL") |>
  analyze("AGE", \(x) list(B = "a"))

build_table(lyt, ex_adsl3)
```



🟩 **layout**

Group summaries

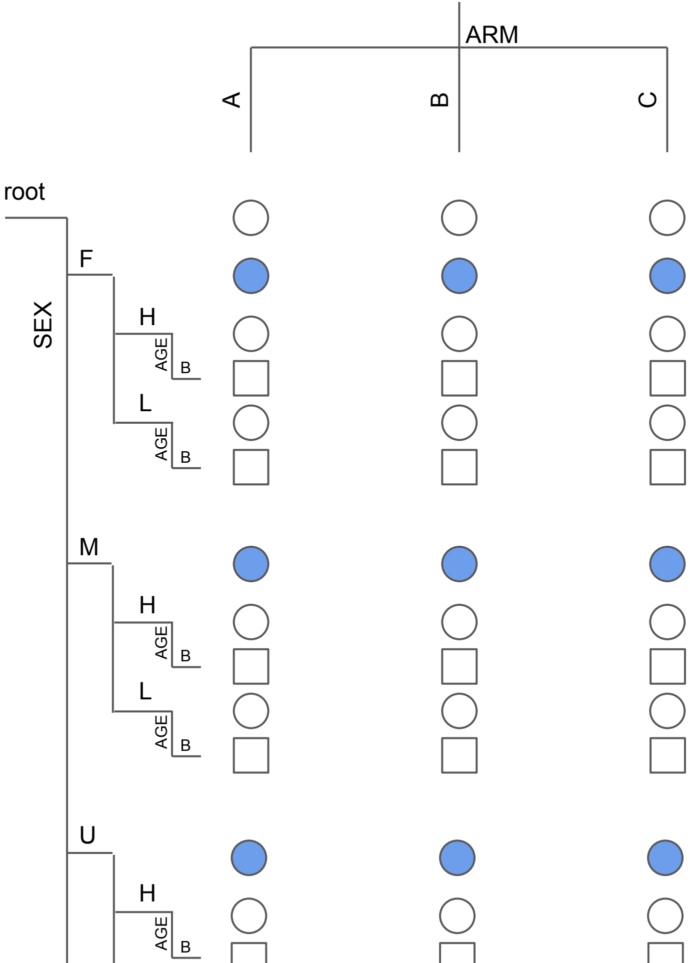
```
lyt <- basic_table() |>
  split_cols_by("ARM") |>
  split_rows_by("SEX") |>
  split_rows_by("B1HL") |>
  analyze("AGE", \(x) list(B = "a"))

build_table(lyt, ex_adsl3)
```



Group summaries

```
lyt <- basic_table() |>
  split_cols_by("ARM") |>
  split_rows_by("SEX") |>
  summarize_row_groups() |>
  split_rows_by("B1HL") |>
  analyze("AGE", \(x) list(B = "a"))

build_table(lyt, ex_adsl3)
```

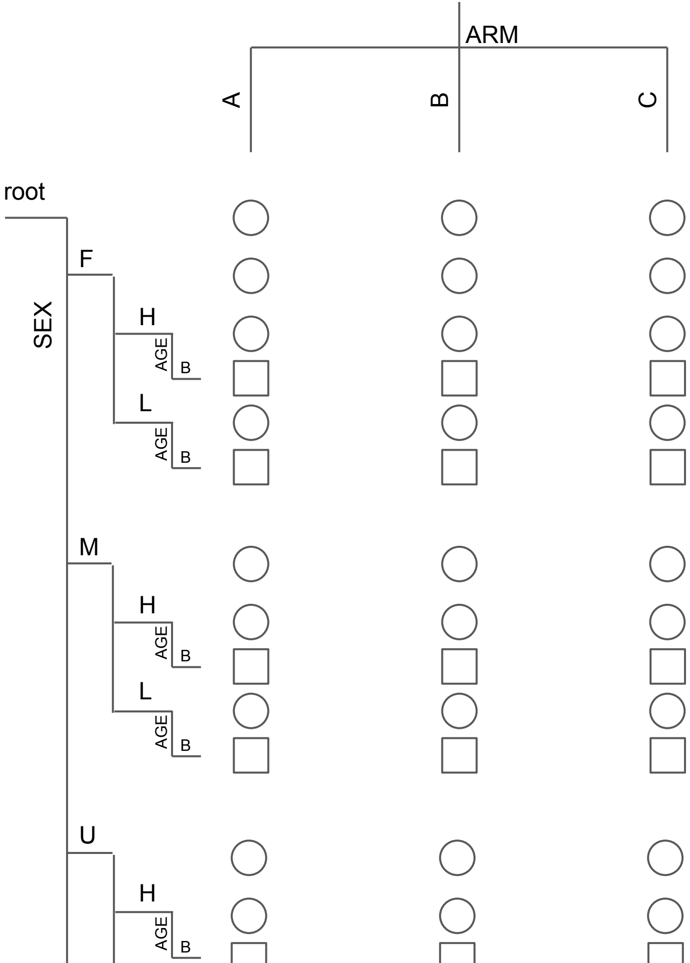
🟩 **layout**



🟩 **layout**

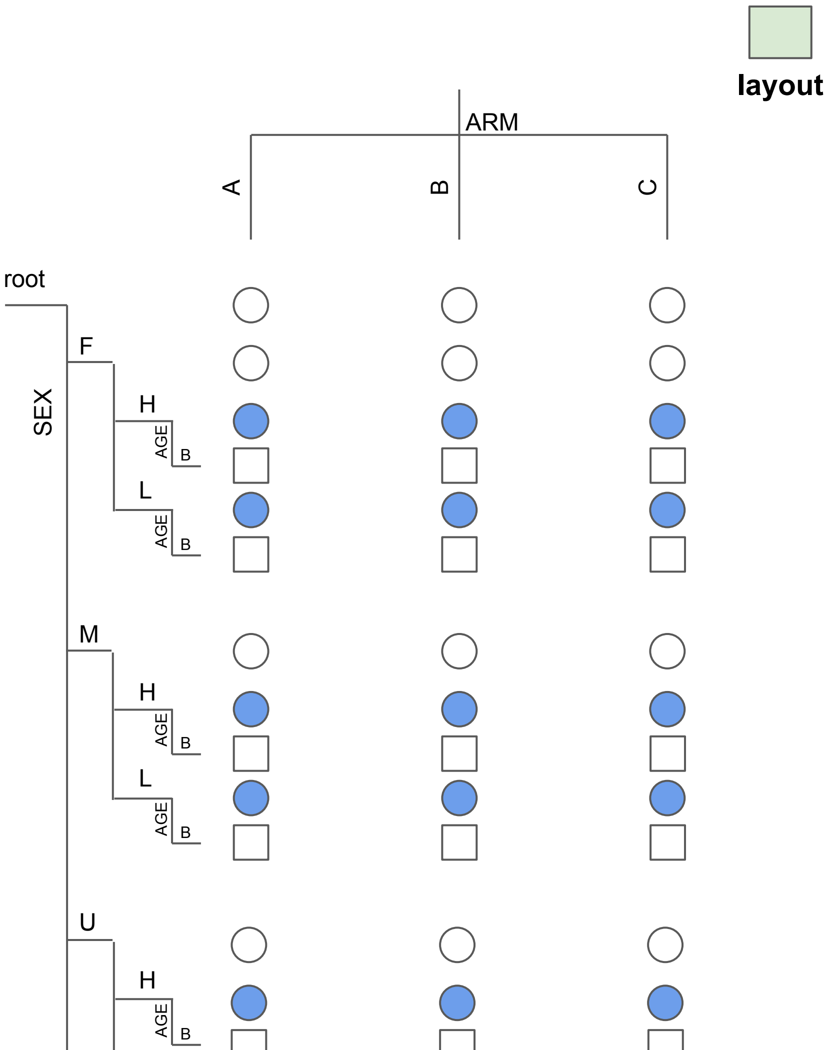
Group summaries

```
lyt <- basic_table() |>
  split_cols_by("ARM") |>
  split_rows_by("SEX") |>
  split_rows_by("B1HL") |>
  analyze("AGE", \(x) list(B = "a"))

build_table(lyt, ex_adsl3)
```



Group summaries

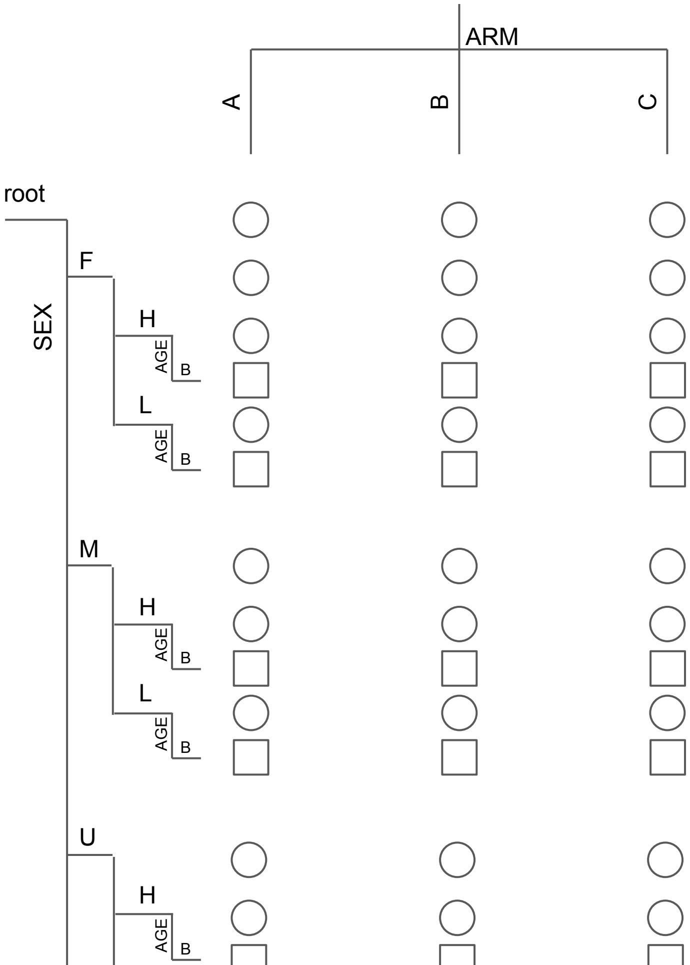
```
lyt <- basic_table() |>
  split_cols_by("ARM") |>
  split_rows_by("SEX") |>
  split_rows_by("B1HL") |>
  summarize_row_groups() |>
  analyze("AGE", \(x) list(B = "a"))

build_table(lyt, ex_adsl3)
```



🟩 **layout**

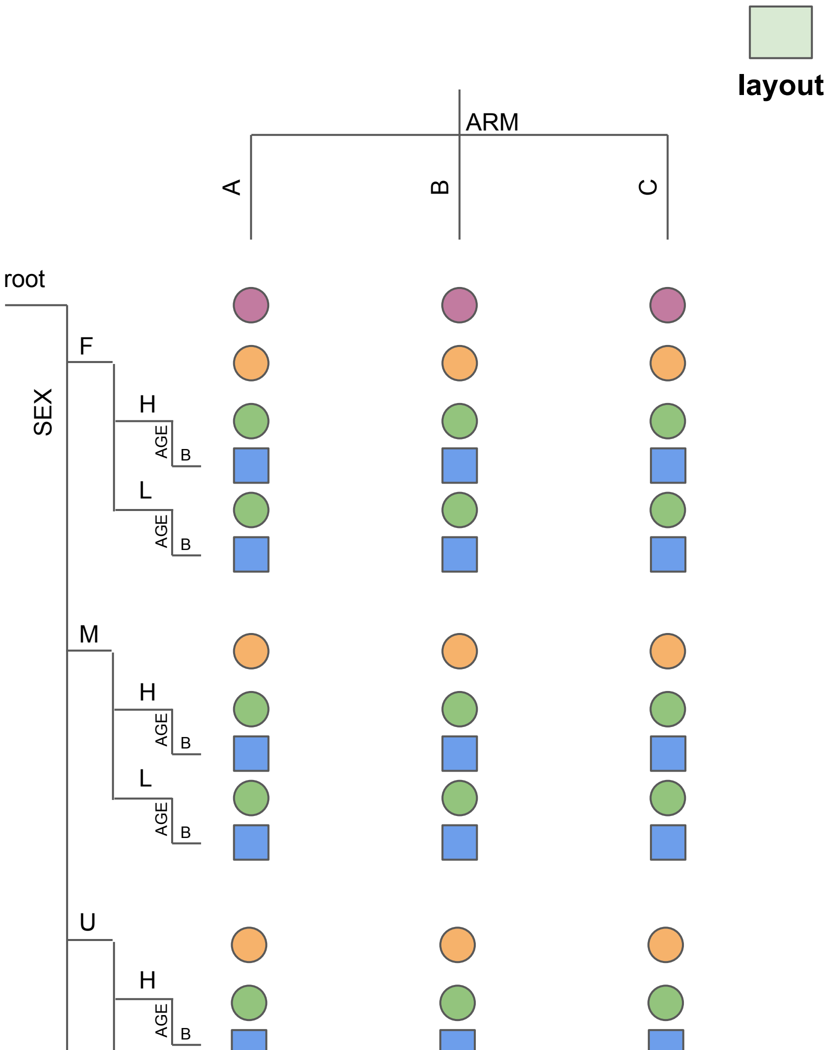
Group summaries

```
lyt <- basic_table() |>
  split_cols_by("ARM") |>
  split_rows_by("SEX") |>
  split_rows_by("B1HL") |>
  analyze("AGE", \(x) list(B = "a"))

build_table(lyt, ex_adsl3)
```



Group summaries

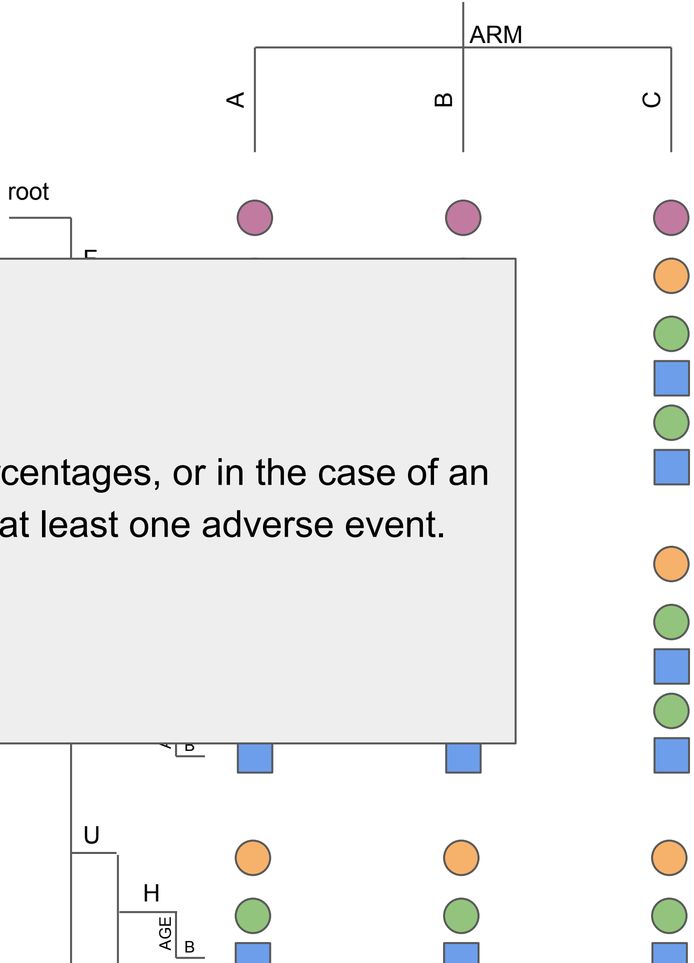
```
lyt <- basic_table() |>
  split_cols_by("ARM") |>
  summarize_row_groups() |>
  split_rows_by("SEX") |>
  summarize_row_groups() |>
  split_rows_by("B1HL") |>
  summarize_row_groups() |>
  analyze("AGE", afun = \(x) list(B = "a"))

build_table(lyt, ex_adsl3)
```



🟩 layout

Group summaries

```
lyt <- basic_table() |>
  split_cols_by("ARM") |>
  summarize_row_groups() |>
  split_rows
  summarize_
  split_rows
  summarize_
  analyze("A

build_table(
```

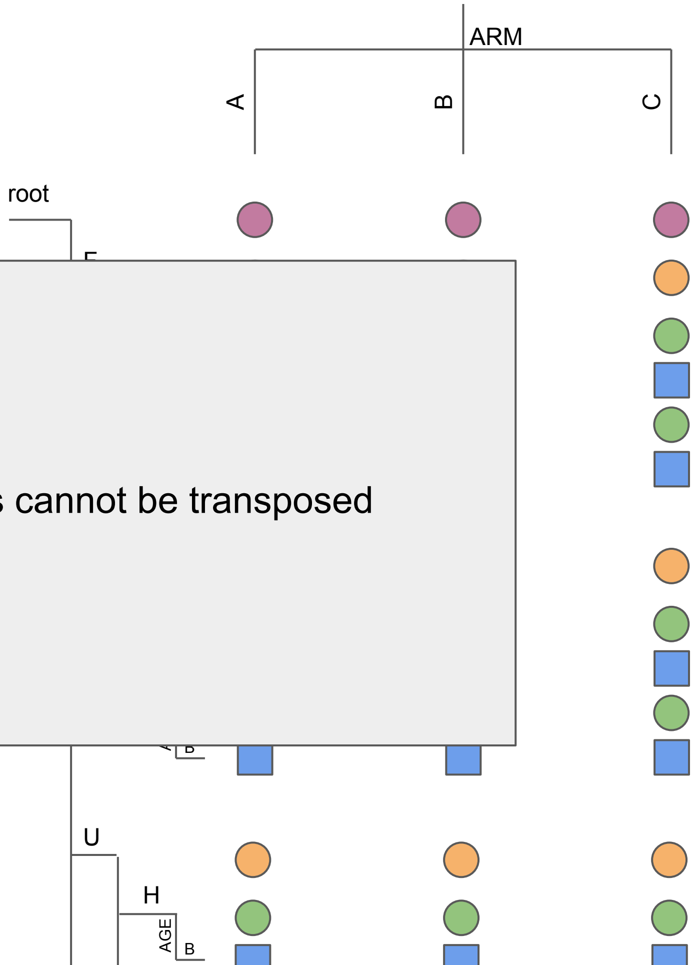


Usually group summaries hold counts, percentages, or in the case of an adverse events table unique patients with at least one adverse event.

Group summaries

```
lyt <- basic_table() |>
  split_cols_by("ARM") |>
  summarize_row_groups() |>
  split_row
  summarize
  split_row
  summarize
  analyze("
"a"))

build_table
```

Note: this is one reason why rtables tables cannot be transposed



🟩 layout

exercise 19

Group Summaries

Basic Adverse Events Table

Let’s apply our knowledge to create an

- adverse events table
- disposition table

exercise 20

Adverse Events Table

note the usage of indent_mod

exercise 21

Disposition Table

Same table for different subsets…
is a breeze with layouts

```r
library(dplyr)

vars <- c("RACE", "SEX", "STRATA1", "STRATA2")
var_lbls <- var_labels(ex_adsl)[vars]

lyt <- basic_table(show_colcounts = TRUE) |>
  split_cols_by(var = "ARM", split_fun = add_overall_level("All Patients")) |>
  summarize_vars(vars, var_labels = var_lbls)

build_table(lyt, ex_adsl)

ex_adsl_USA <- ex_adsl |>
  filter(COUNTRY == "USA")

build_table(lyt, ex_adsl_USA)
```

exercise 22

reuse Layouts

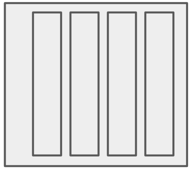
Layouts topics left to the reader

The following topics are important for regulatory table generation but we don’t have sufficient time to discuss them here:

- comparison to baseline (doc)
    - e.g make ARM A the reference column
- trim_levels_to_map (doc)
- section dividers (doc)
- indent modifications (doc)
- table_inset, (?table_inset)
- page_by (doc)

Luckily we have extensive documentation (roche.github.io/rtables) 👍

So whats next…



data.frame
Unaggregated data
(variables + obs)

layout
faceting, value
derivation, formatting
instructions

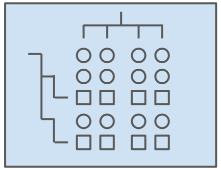
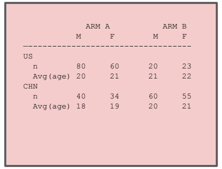


table object
cell values,
table structure,
format instructions,
paths

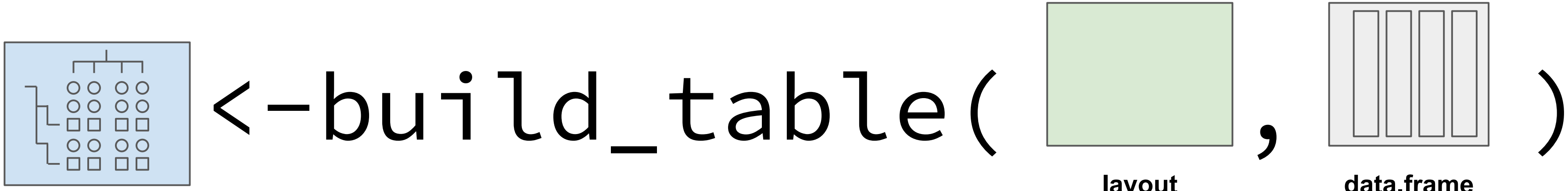


```
                ARM A            ARM B
              M      F        M      F
-----------------------------------
US
  n          80     60       20     23
  Avg(age)   20     21       21     22
CHN
  n          40     34       60     55
  Avg(age)   18     19       20     21
```

formatted table
such as ASCII, pdf,
rtf

this topic is will take less time, promised!

Recap 😀



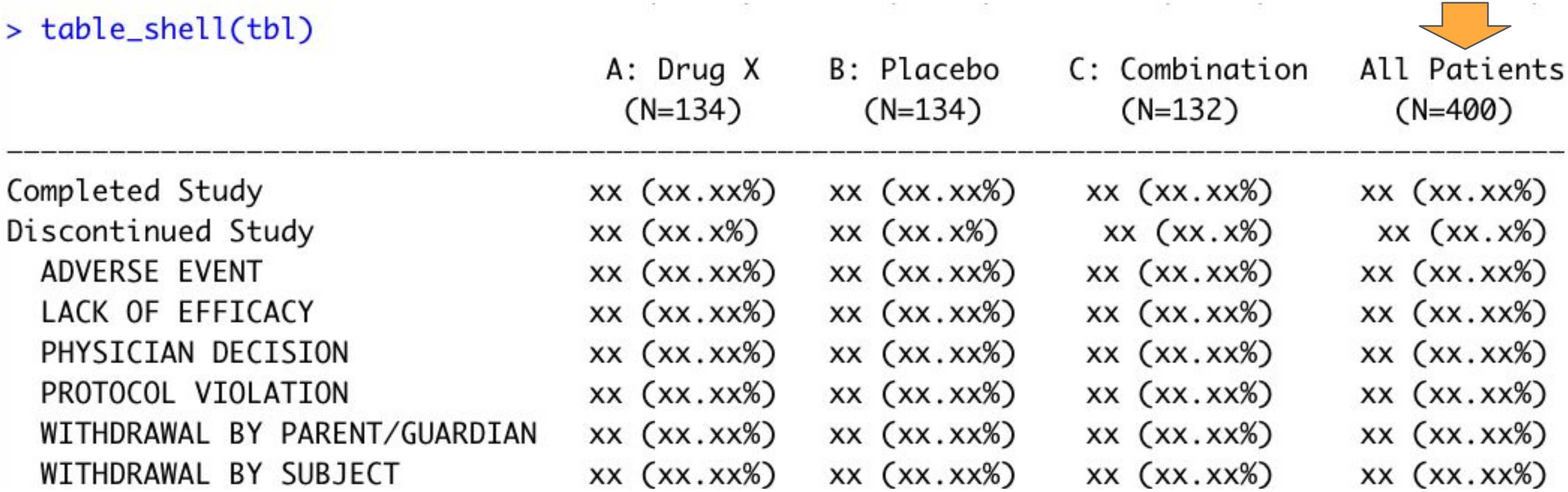
`<-build_table( , )`

table object
cell values,
table structure,
format instructions,
paths

layout
faceting, value
derivation, formatting
instructions

data.frame
Unaggregated data
(variables + obs)

Let’s start simple



known issue
we will fix it

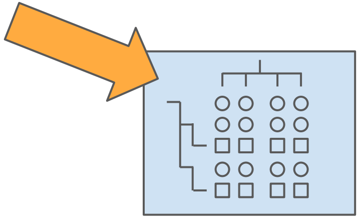
```
> table_shell(tbl)
                                  A: Drug X      B: Placebo     C: Combination    All Patients
                                   (N=134)        (N=134)          (N=132)          (N=400)
----------------------------------------------------------------------------------------------
Completed Study                  xx (xx.xx%)    xx (xx.xx%)     xx (xx.xx%)      xx (xx.xx%)
Discontinued Study               xx (xx.x%)     xx (xx.x%)       xx (xx.x%)       xx (xx.x%)
  ADVERSE EVENT                  xx (xx.xx%)    xx (xx.xx%)     xx (xx.xx%)      xx (xx.xx%)
  LACK OF EFFICACY               xx (xx.xx%)    xx (xx.xx%)     xx (xx.xx%)      xx (xx.xx%)
  PHYSICIAN DECISION             xx (xx.xx%)    xx (xx.xx%)     xx (xx.xx%)      xx (xx.xx%)
  PROTOCOL VIOLATION             xx (xx.xx%)    xx (xx.xx%)     xx (xx.xx%)      xx (xx.xx%)
  WITHDRAWAL BY PARENT/GUARDIAN  xx (xx.xx%)    xx (xx.xx%)     xx (xx.xx%)      xx (xx.xx%)
  WITHDRAWAL BY SUBJECT          xx (xx.xx%)    xx (xx.xx%)     xx (xx.xx%)      xx (xx.xx%)
```

Table objects

- rtables table objects are implemented with a tree data structure

Table objects

- rtables table objects are implemented with a tree data structure



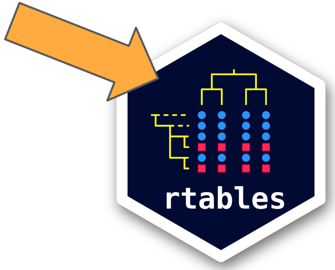
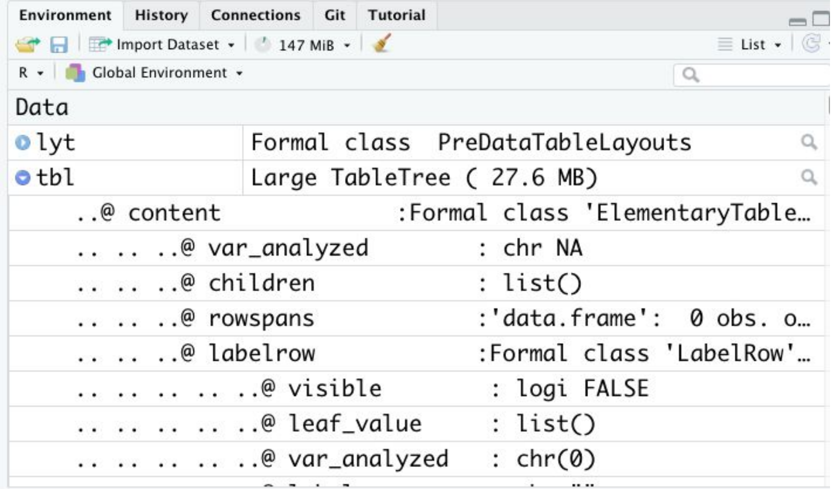


Table objects

- rtables table objects are implemented with a tree data structure

- you rarely (likely never) have to walk the tree



```
Data
lyt                        Formal class  PreDataTableLayouts
tbl                        Large TableTree ( 27.6 MB)
    ..@ content                     :Formal class 'ElementaryTable…
    .. .. ..@ var_analyzed                 : chr NA
    .. .. ..@ children                     : list()
    .. .. ..@ rowspans                     :'data.frame':  0 obs. o…
    .. .. ..@ labelrow                     :Formal class 'LabelRow'…
    .. .. .. .. ..@ visible          : logi FALSE
    .. .. .. .. ..@ leaf_value       : list()
    .. .. .. .. ..@ var_analyzed     : chr(0)
```

Table objects

- rtables table objects are implemented with a tree data structure

- you rarely (likely never) have to walk the tree

- trees retain lots of information about the table construction
- trees are also useful for subsetting & pagination

Table objects

- To learn about the structure of a table object:
  - `table_structure()`, `make_row_df()`, `make_col_df`, `row_paths`, `col_paths`
  - `dim()`, `nrow()`, `ncol()`
  - **`NOT str()`** (I mean it, it will not help you)
- rtables table objects are implemented with a tree data structure
  - You won’t need to know this beyond understanding pathing
  - Useful for lots of functionality internally
    - Pagination
    - subsetting

You can subset tables

We support 2 subsetting methods

- Absolut index based subsetting
  - tbl[1:30, 3:3]
  - Possible to make non-meaningful table
- Path based subsetting that retains its context
  - tbl[]
  - Cell values are in context of their group

exercise 24

subsetting table

Pagination

The pagination algorithm relies heavily on the tree structure

- how many leafs per page must be grouped
- repeated summary information

.We have discussed pagination in more detail earlier in the course.

rbind/cbind

Tables from the rtables package support cbind and rbind in order to join two or more tables.

- This is rarely the best solution, try to use sequential analyze and non-nested splits instead

Compare rtables

The `compare_rtables()` function lets you compare two rtables objects.

To compare two tables visually you can use the Viewer() function.

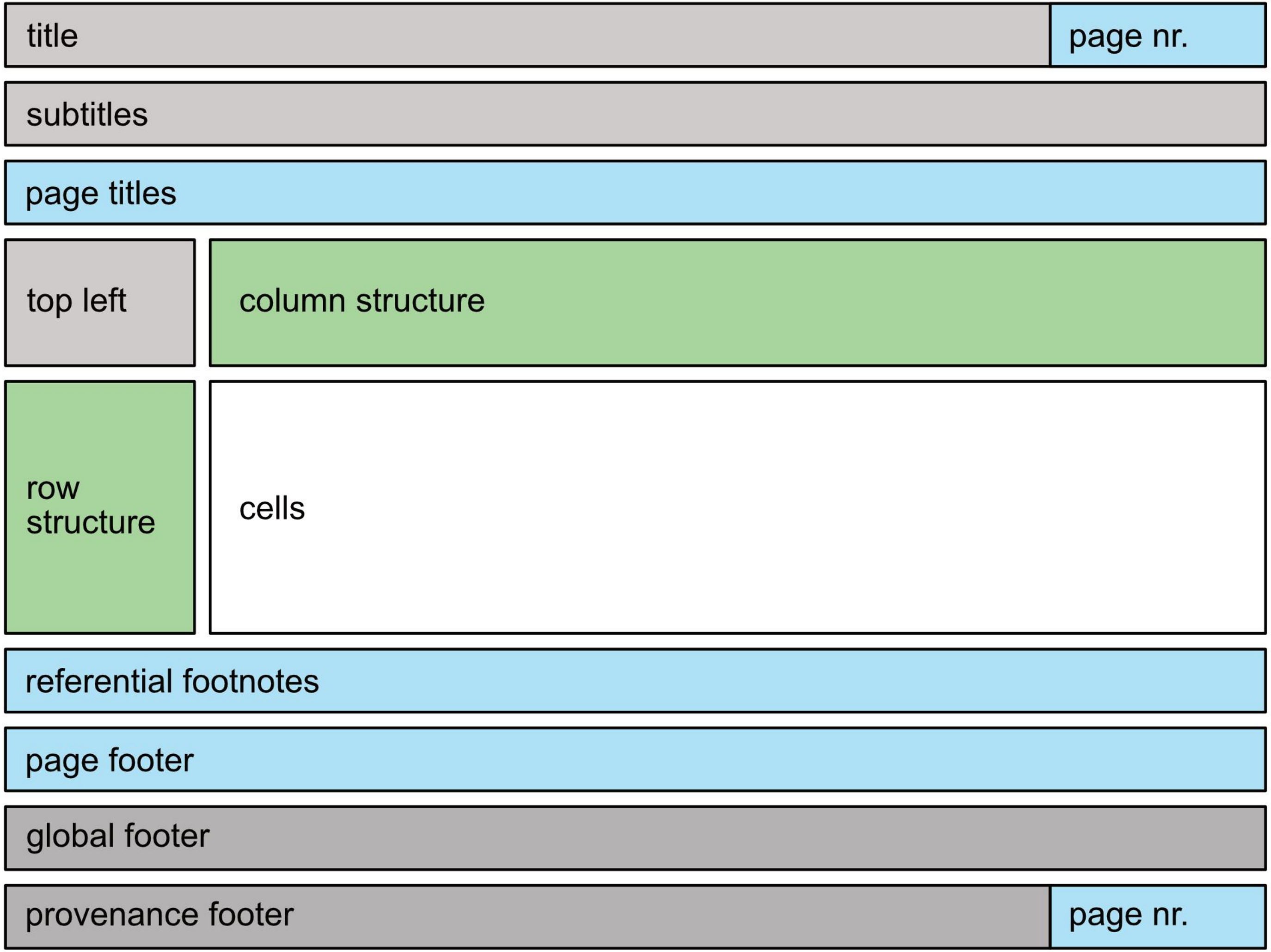
exercise 26

compare rtables

Referential footnotes

```
Analysis DRAFT
[subtitles] Table XXX.X - Silly Age Analysis Where We Compare Drugs
That Aren't Real To Placebo

    ———————————————————————————————————————————————————————————————————————————————
                                       A: Drug X {1}   B: Placebo   C: Combination
    ———————————————————————————————————————————————————————————————————————————————
    WHITE {2}                           14 (11.6%)     14 (13.2%)   18 (14.0%) {3}
      mean                                 39.4         36.9 {4}         35.1
    BLACK OR AFRICAN AMERICAN           28 (23.1%)     24 (22.6%)     27 (20.9%)
      mean                                 34.7           31.7           34.0
    ASIAN                               79 (65.3%)     68 (64.2%)   84 (65.1%) {5}
      mean                                 34.2           32.7           34.6
    AMERICAN INDIAN OR ALASKA NATIVE     0 (0.0%)       0 (0.0%)       0 (0.0%)
      mean                                   NA            NA             NA
    MULTIPLE {6, 7}                      0 (0.0%)       0 (0.0%)       0 (0.0%)
      mean                                   NA            NA             NA
    ———————————————————————————————————————————————————————————————————————————————

    {1} - drug x is a fake drug that isn't real
    {2} - didn't specify content
    {3} - cell fnote didn't specify content
    {4} - white arm b mean
    {5} - asian arm c content is where the group summary for asian
    patients given the combination lives
    {6} - race multiple row fn 1
    {7} - race multiple row fn 2
    ———————————————————————————————————————————————————————————————————————————————

    [Main Footer] Funded By Company X, Administered by Company Y, Some
    other companies also paid attention too

[Prov Footer] Study SuperAwesomeStudy - File:
/path/to/study/data/superawesome/superawesome_dm.csv
   Data Snapshot: 12/12/2012 - Hash: 5e338704a8e069ebd8b38ca71991cf94
```

see docs.



title

page nr.

subtitles

page titles

top left

column structure

row structure

cells

referential footnotes

page footer

global footer

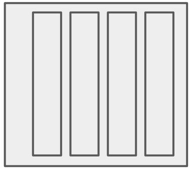
provenance footer

page nr.

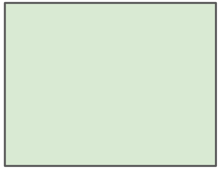
Colors indicate how the content changes when paginating tables

* page nr. is not implemented yet, i.e. user have to add themselves

So whats next…



data.frame
Unaggregated data
(variables + obs)



layout
faceting, value
derivation, formatting
instructions

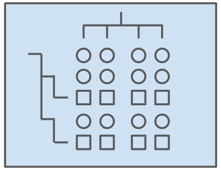
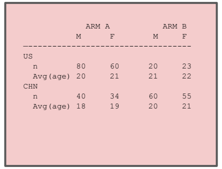


table object
cell values,
table structure,
format instructions,
paths



```
              ARM A           ARM B
            M      F        M      F
-----------------------------------
US
  n         80     60       20     23
  Avg(age)  20     21       21     22
CHN
  n         40     34       60     55
  Avg(age)  18     19       20     21
```

formatted table
such as ASCII, pdf,
rtf

Luckily we have already covered this in our example at the beginning.

And That’s `rtables` in ~2.5 Hours

You are now an advanced rtables user. Practice makes perfect!

One more thing…

Please take time studying the tern documentation:

- https://insightsengineering.github.io/tern

there is a good chance that tern will help you making most of your clinical trials analysis tables relatively fast.

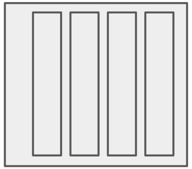
We would like to hear from you

- Please file an issue on https://github.com/Roche/rtables/issues if
    - You have a technical issue
    - Need a particular feature
    - Have general feedback
    - Need help

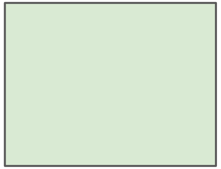
rtables

thank you !

So whats next…



data.frame
Unaggregated data
(variables + obs)



layout
faceting, value
derivation, formatting
instructions

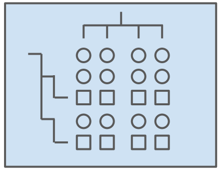
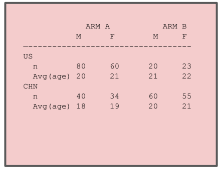


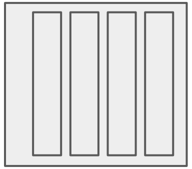
table object
cell values,
table structure,
format instructions,
paths



```
             ARM A            ARM B
           M       F        M     F
------------------------------------
US
  n        80      60      20     23
  Avg(age) 20      21      21     22
CHN
  n        40      34      60     55
  Avg(age) 18      19      20     21
```

formatted table
such as ASCII, pdf,
rtf

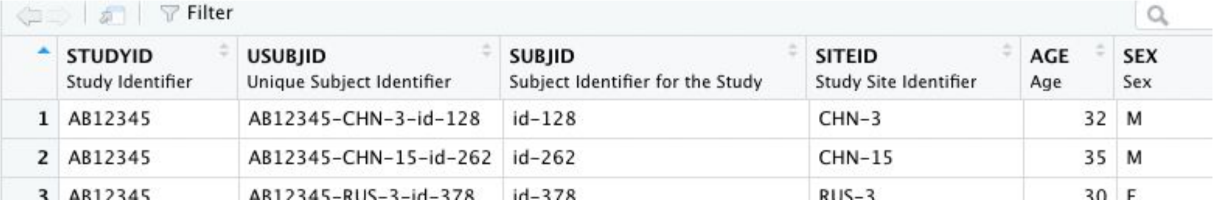
Data



data.frame
rows and columns
(variables)

Data frames or tibbles are supported. You may derive them with dplyr (in general) or with admiral (ADaM, CDISC).

- Variables in clinical trials have a name and labels



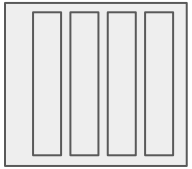
	STUDYID	USUBJID	SUBJID	SITEID	AGE	SEX
	Study Identifier	Unique Subject Identifier	Subject Identifier for the Study	Study Site Identifier	Age	Sex
1	AB12345	AB12345-CHN-3-id-128	id-128	CHN-3	32	M
2	AB12345	AB12345-CHN-15-id-262	id-262	CHN-15	35	M
3	AB12345	AB12345-RUS-3-id-378	id-378	RUS-3	30	F

- the formatters R package (which comes with rtables) has a number of synthetic ADaM datasets
  - ex_adsl is a wide dataset
  - the others are long datasets

exercise A1

data

Data - factors



data.frame
rows and columns
(variables)

Factors in R implement a data structure that is useful to represent categorical (ordered or unordered) data.

```
df <- data.frame(
  ARM = factor(c("A", "B", "A"), levels = c("A", "B", "C")),
  AGE = c(21, 23, 14)
)
```

The order of the factor levels determines the order of the columns/rows

exercise A2

factors in rtables

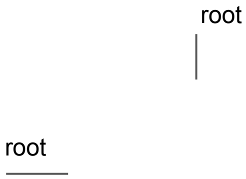
It’s complete code

Row Structure

- So far we have focused on controlling column structure.
- row structure is also defined via layouting instructions
- row and column structure define the facets
- rtables table model does not support the transpose operation
    - Concepts are similar
    - use split_rows_* layouting functions
    - Split function usage are the same

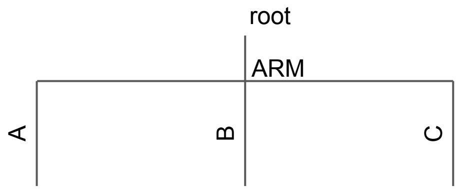
Row structure

```
lyt <- basic_table()

build_table(lyt, ex_adsl3)
```



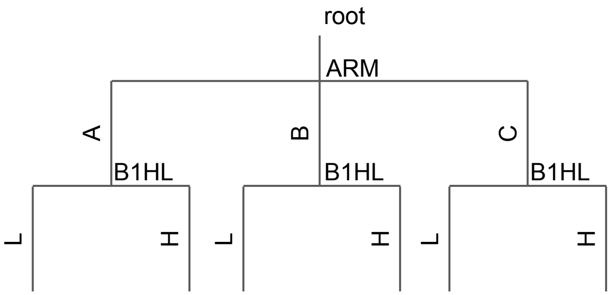
Row structure

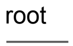
```
lyt <- basic_table() |>
  split_cols_by("ARM")

build_table(lyt, ex_adsl3)
```



Row structure

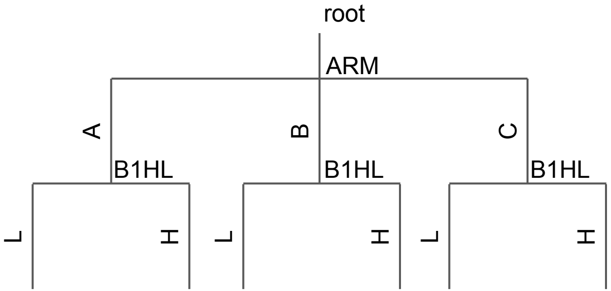
```
lyt <- basic_table() |>
  split_cols_by("ARM") |>
  split_cols_by("B1HL")

build_table(lyt, ex_adsl3)
```

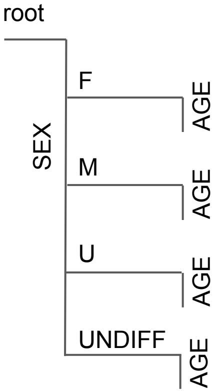




Row structure

```
lyt <- basic_table() |>
  split_cols_by("ARM") |>
  split_cols_by("B1HL") |>
  split_rows_by("SEX") |>
  analyze("AGE", function(x) "")

build_table(lyt, ex_adsl3)
```





exercise A3

Row Structure