rtables For Power-Users Part 1

Nest Team Training

Jan 31, 2023

Gabriel Becker, Adrian Waddell

rtables

What you will learn

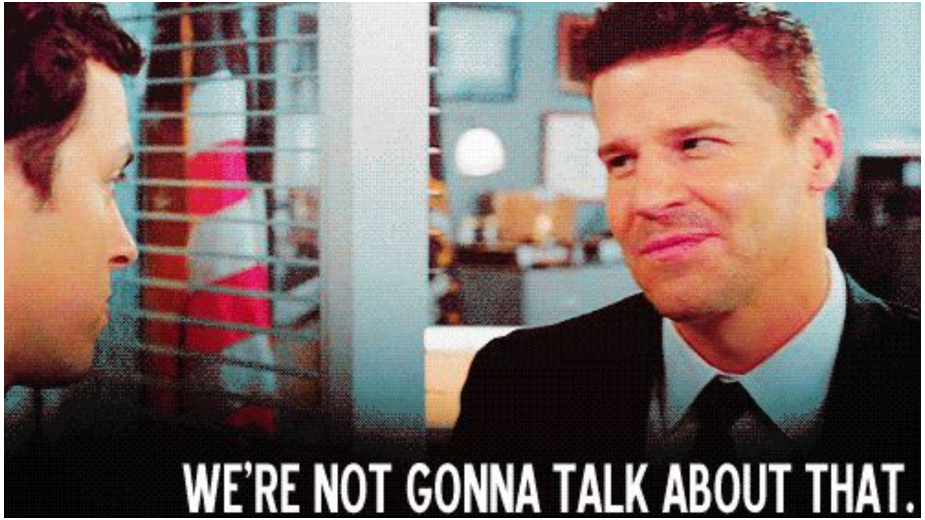
- rtables fundamentals
  - Conceptual model and motivations
  - Basic layout construction
- Table Structure
  - Pathing
  - Subsetting
  - Post-processing and mitigation
- (Semi-) Advanced Features
  - Specialized splitting
  - Appearance customization
  - Pagination

What Will Wait For Part 2

- (Semi-) Advanced Topics
  - Whatever we don’t have time to get to today
- Advanced customization
  - Custom-written split functions
  - Advanced features of analysis function writing
- Tying it all together
  - “Thinking in `rtables`”
  - Reasoning about workarounds

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!

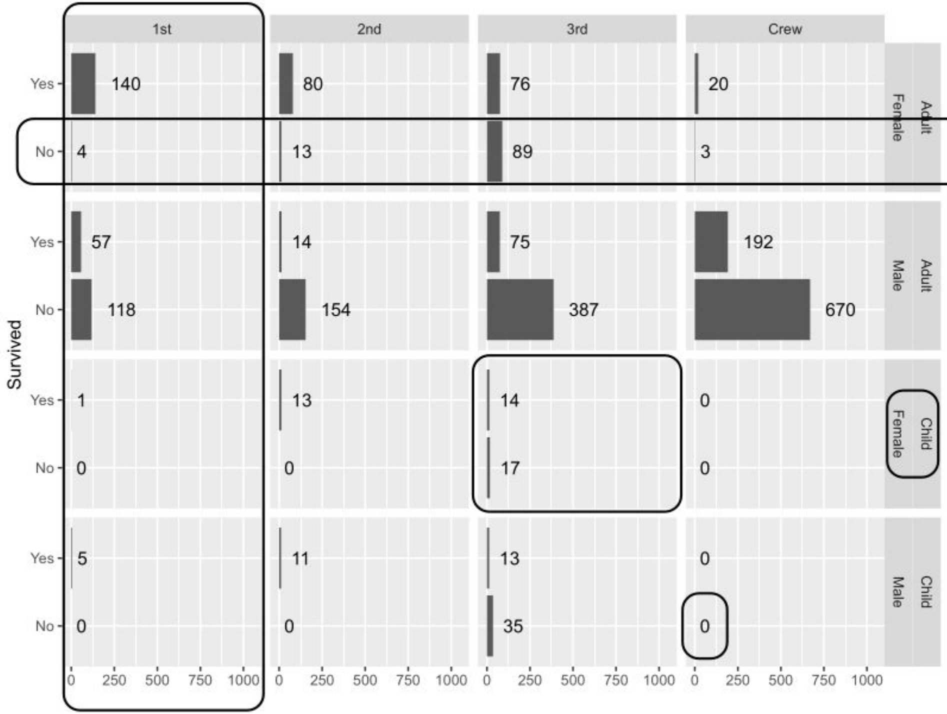
The First Step In Creating a Table

Calculating cell values, right?

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

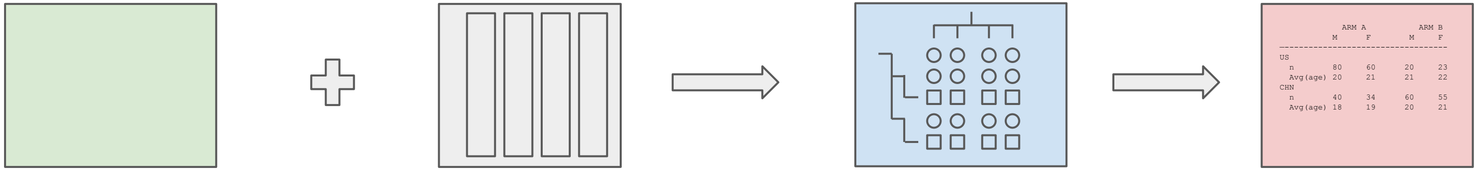
Imagine Manually Subsetting Facet Data When Using `ggplot2` (or `lattice`)

Subsetting data and calculating facet statistics

Humans

Computers

Basics of rtables



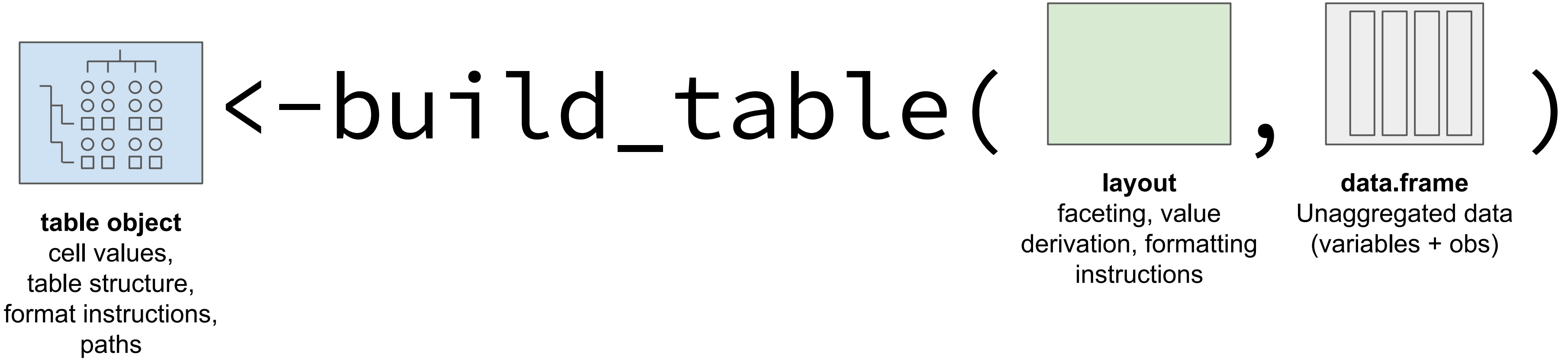
layout
faceting, value derivation, formatting instructions

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



Basics of rtables

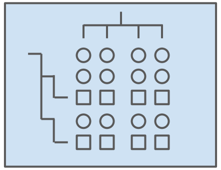
`print(`  `)`

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

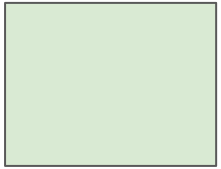
```
             ARM A            ARM B
           M       F        M      F
--------------------------------------
US
  n         80      60       20     23
  Avg(age)  20      21       21     22
CHN
  n         40      34       60     55
  Avg(age)  18      19       20     21
```

formatted table
here ASCII

Basics of rtables

```
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



layout
faceting, value derivation, formatting instructions

```
library(rtables)

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

tbl <- build_table(lyt, ex_adsl)

print(tbl)
```

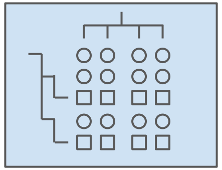
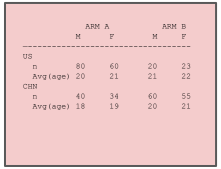
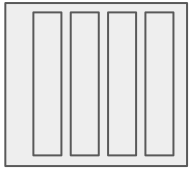


table object
cell values,
table structure,
format instructions,
paths



```
              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



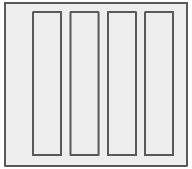
data.frame
Unaggregated data
(variables + obs)

Basics of rtables

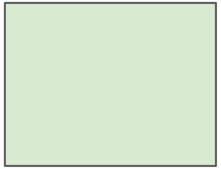
```
> library(rtables)
Loading required package: magrittr
Loading required package: formatters
>
> lyt <- basic_table() |>
+   split_cols_by("ARM") |>
+   analyze("AGE", mean, format = "xx.xx")
>
> tbl <- build_table(lyt, ex_adsl)
>
> print(tbl)
       A: Drug X   B: Placebo   C: Combination
———————————————————————————————————————————————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

now let's get cracking

So whats next…



data.frame
Unaggregated data
(variables + obs)



layout
faceting, value
derivation, formatting
instructions

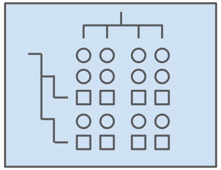
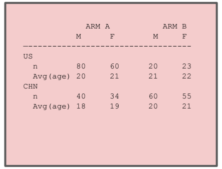


table object
cell values,
table structure,
format instructions,
paths



	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)

- Faceting (row and column)
  - `split_rows_by()` (and sibling funs), `split_cols_by()` (and sibling funs)
- cell value derivation
  - Via `analyze()` and `summarize_row_groups()`
- every layout starts with `basic_table()`
  - Metadata (titles, footer)
  - Display of column counts

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)1
```

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

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
```

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

Analyzing More Than One Variable Within a Facet

- `analyze` calls can be called sequentially
  - `nested = TRUE` (the default) combines them within facets
  - `nested = FALSE` turns this off and generates a new top-level subtable
- `analyze` can be applied to multiple variables
  - `analyze` accepts a list of analysis functions in this case

They are equivalent (by default).

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`)

advanced topic, covered in pt 2

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)
```

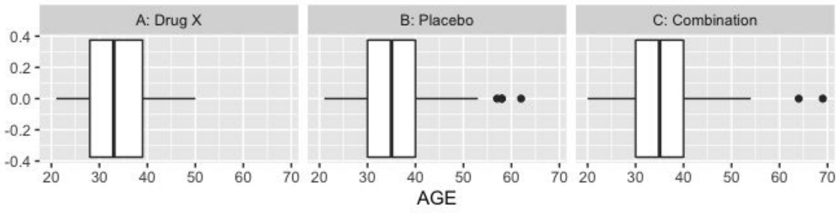
```
             all obs
————————————————————————
pct_afun   400 (100.0%)
```

Declaring Facets

- `split_rows_by()` (and sibling funcs) add row faceting structure
- `split_cols_by()` (and sibling funcs) 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))
```

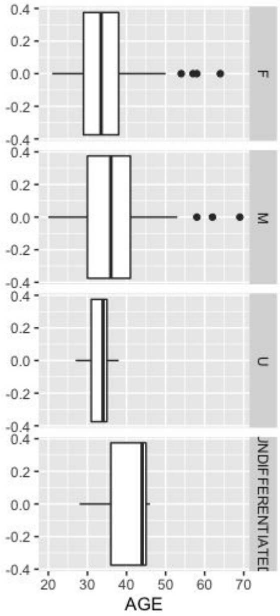



```
lyt <- basic_table() |>
  split_cols_by("ARM") |>
  analyze("AGE", range, format = "xx.xx - xx.xx")

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))
```

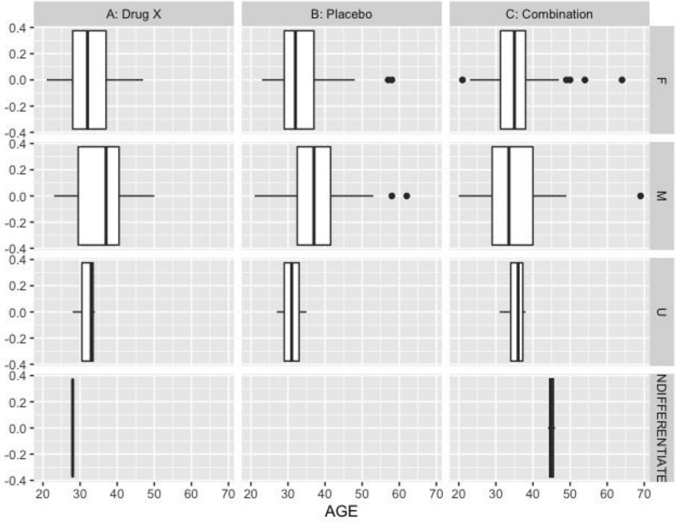



```
lyt2 <- basic_table() |>
  split_rows_by("SEX") |>
  analyze("AGE", range,
     format = "xx.xx - xx.xx")

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))
```

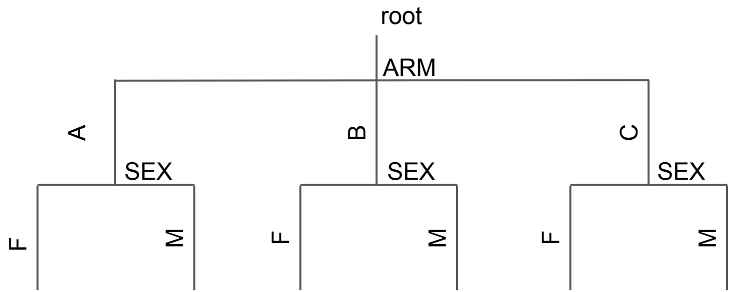


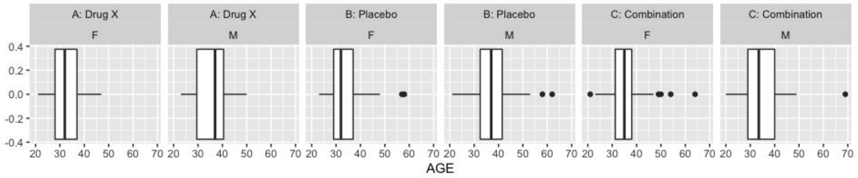

```
lyt3 <- basic_table() |>
  split_cols_by("ARM") |>
  split_rows_by("SEX") |>
  analyze("AGE", range, format = "xx.xx - xx.xx")

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
```

Nested Faceting Structure

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

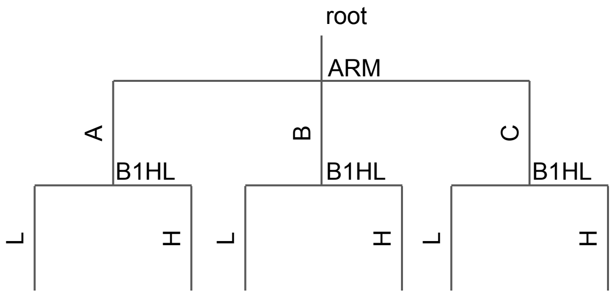


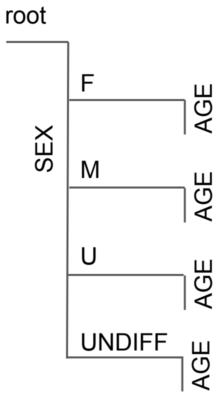



```
                 A: Drug X                   B: Placebo                  C: Combination
              F              M             F              M             F              M
————————————————————————————————————————————————————————————————————————————————————————————
range    21.0 - 47.0    23.0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

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)
```





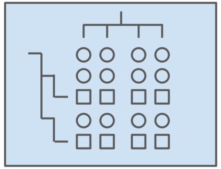


table object
numbers, strings
paths

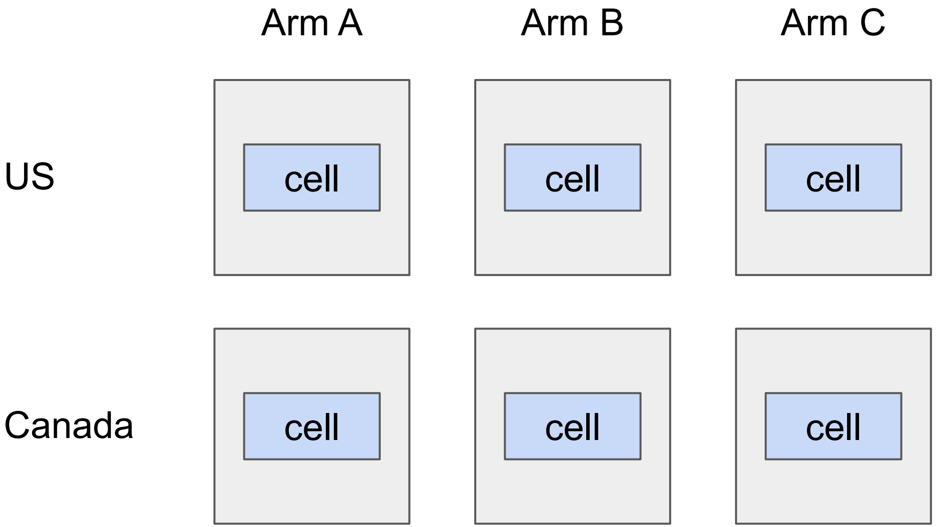
Analyze revisited

- The analysis function is applied ***within each facet*** declared by the splitting it is nested within, generating **one or more cell values** via **`in_rows()`**

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

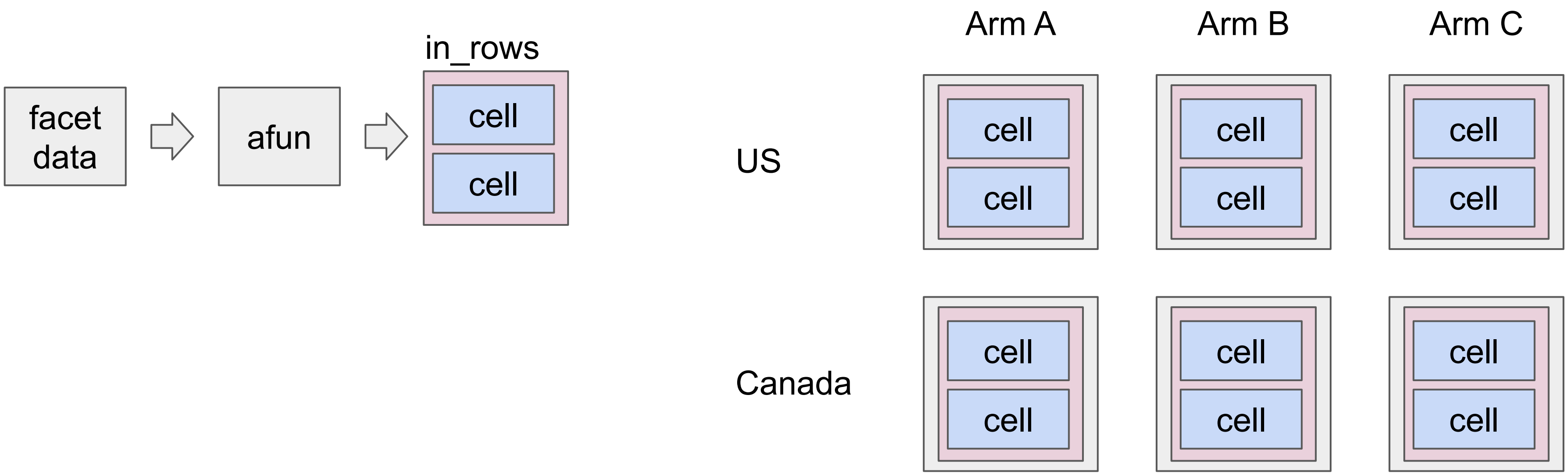
Analyze revisited

- The analysis function is applied ***within each facet*** declared by the splitting it is nested within, generating **one or more cell values** via **`in_rows()`**



Analyze revisited

- The analysis function is applied ***within each facet*** declared by the splitting it is nested within, generating **one or more cell values** via `in_rows()`



Analyze revisited

- The analysis function is applied ***within each facet*** declared by the splitting it is nested within, generating **one or more cell values** via **`in_rows()`**

```
lyt2 <- basic_table() %>%
    split_cols_by("ARM") %>%
    split_rows_by("STRATA1") %>%
    analyze("AGE", 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"))
    })

build_table(lyt2, ex_adsl)
```

```
> build_table(lyt2, ex_adsl)
                A: Drug X    B: Placebo    C: Combination
—————————————————————————————————————————————————————————
A
  n                38            44              40
  mean (sd)    33.1 (5.7)    35.1 (7.9)     34.2 (6.2)
  median          32.0          32.5            35.0
  min - max    24.0 - 46.0   23.0 - 62.0    20.0 - 47.0
B
  n                47            45              43
  mean (sd)    33.9 (7.2)    36.0 (9.1)     36.3 (8.4)
  median          33.0          36.0            35.0
  min - max    23.0 - 48.0   21.0 - 58.0    21.0 - 64.0
C
  n                49            45              49
  mean (sd)    34.2 (6.6)    35.2 (6.6)     35.6 (8.2)
  median          34.0          35.0            35.0
  min - max    21.0 - 50.0   23.0 - 51.0    24.0 - 69.0
```

Analyze revisited

- The analysis function is applied ***within each facet*** declared by the splitting it is nested within, generating **one or more cell values** via **`in_rows()`**

```
lyt2 <- basic_table() %>%
    split_cols_by("ARM") %>%
    split_rows_by("STRATA1") %>%
    analyze("AGE", 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"))
    })

build_table(lyt2, ex_adsl)
```

```
> build_table(lyt2, ex_adsl)
               A: Drug X     B: Placebo    C: Combination
—————————————————————————————————————————————————————————
A
  n                38            44              40
  mean (sd)    33.1 (5.7)    35.1 (7.9)      34.2 (6.2)
  median          32.0          32.5            35.0
  min - max    24.0 - 46.0   23.0 - 62.0     20.0 - 47.0
B
  n                47            45              43
  mean (sd)    33.9 (7.2)    36.0 (9.1)      36.3 (8.4)
  median          33.0          36.0            35.0
  min - max    23.0 - 48.0   21.0 - 58.0     21.0 - 64.0
C
  n                49            45              49
  mean (sd)    34.2 (6.6)    35.2 (6.6)      35.6 (8.2)
  median          34.0          35.0            35.0
  min - max    21.0 - 50.0   23.0 - 51.0     24.0 - 69.0
```

Core Types Of Split

Variable Split

`split_rows_by("<var>")`, `split_cols_by("<var>")`

- Used for *categorical variables*
- One facet per ***variable level***
    - Including empty levels for factors
- Facet data is incoming data subset to `<var> == <single level>`
- Most common, basic split

Multivar Split

```
split_rows_by_multivar(<vector of varnames>),
split_cols_by_multivar(<vector of varnames)
```

- One facet per ***variable***
- **Facet data is full incoming data**
- Most common/useful in *column space*
- Useful when
  - Incoming data has precalculated statistics in columns
    - E.g., *model summary display*
  - Incoming data is in (very) wide form
    - E.g., different columns for measurements from Visit 1, Visit 2, …
- Must use `analyze_colvars()` for cell content derivation
  - Analysis function can be different for different variables (passed as list)

Multivar Split Example

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

```
> build_table(lyt, DM)
   Ethn. Present   Ave. Age
---------------------------
         3            34.2
```

Static Cut Splits

```
split_rows_by_cuts("<var>", cuts = <>, cumulative = <>),
split_cols_by_cuts("<var>", cuts = <>, cumulative = <>)
```

- Used to split on values of *numeric/continuous variables*
- One facet per ***discretized value*** of <var>
- Cut points for discretization are *fixed, data independent*
- Categories can be cumulative *(*`cumulative = TRUE`*)*

Static Cut Split Examples

```
lyt_scut <- basic_table(show_colcounts = TRUE) %>%
    split_cols_by_cuts("AGE", cuts = c(1, 30, 40, 50, 1000),
                       cutlabels = c("<30", "30-40", "40-50", "50+")) %>%
    analyze("BMRKR1")
```

```
> build_table(lyt_scut, DM)
          <30       30-40     40-50      50+
        (N=122)   (N=178)   (N=45)    (N=11)
——————————————————————————————————————————
Mean     5.98      5.63      6.18      6.68
```

```
lyt_scut_cum <- basic_table(show_colcounts = TRUE) %>%
    split_cols_by_cuts("AGE", cuts = c(1, 30, 40, 50, 1000),
                       cutlabels = c("<30", "<40", "<50", "All"),
                       cumulative = TRUE) %>%
    analyze("BMRKR1")
```

```
> build_table(lyt_scut_cum, DM)
          <30        <40       <50        All
        (N=122)   (N=300)   (N=345)   (N=356)
————————————————————————————————————————————
Mean     5.98      5.77      5.83      5.85
```

Static Cut Split Examples

```
lyt_scut <- basic_table(show_colcounts = TRUE) %>%
    split_cols_by_cuts("AGE", cuts = c(1, 30, 40, 50, 1000),
                       cutlabels = c("<30", "30-40", "40-50", "50+")) %>%
    analyze("BMRKR1")

> build_table(lyt_scut, DM)
          <30      30-40    40-50    50+
        (N=122)  (N=178)  (N=45)   (N=11)
——————————————————————————————————————————
Mean     5.98     5.63     6.18     6.68
```

```
lyt_scut_cum <- basic_table(show_colcounts = TRUE) %>%
    split_cols_by_cuts("AGE", cuts = c(1, 30, 40, 50, 1000),
                       cutlabels = c("<30", "<40", "<50", "All"),
                       cumulative = TRUE) %>%
    analyze("BMRKR1")

> build_table(lyt_scut_cum, DM)
          <30       <40      <50       All
        (N=122)  (N=300)  (N=345)   (N=356)
———————————————————————————————————————————
Mean     5.98     5.77     5.83      5.85
```

Static Cut Split Examples

```
lyt_scut <- basic_table(show_colcounts = TRUE) %>%
    split_cols_by_cuts("AGE", cuts = c(1, 30, 40, 50, 1000),
                       cutlabels = c("<30", "30-40", "40-50", "50+")) %>%
    analyze("BMRKR1")

> build_table(lyt_scut, DM)
           <30      30-40     40-50     50+
         (N=122)   (N=178)   (N=45)   (N=11)
——————————————————————————————————————————
Mean      5.98      5.63      6.18     6.68
```

```
lyt_scut_cum <- basic_table(show_colcounts = TRUE) %>%
    split_cols_by_cuts("AGE", cuts = c(1, 30, 40, 50, 1000),
                       cutlabels = c("<30", "<40", "<50", "All"),
                       cumulative = TRUE) %>%
    analyze("BMRKR1")

> build_table(lyt_scut_cum, DM)
           <30       <40       <50       All
         (N=122)   (N=300)   (N=345)   (N=356)
————————————————————————————————————————————
Mean      5.98      5.77      5.83      5.85
```

Dynamic Cut Splits

```
split_rows_by_cutfun("<var>", cutfun = <>, cumulative = <>),
split_cols_by_cutfun("<var>", cutfun = <>, cumulative = <>)
```

- Used to split on values of *numeric/continuous variables*
- One facet per ***discretized value*** of <var>
- Cut points are *dynamic, depend on data*
  - Cut points are calculated once, by ***applying `cutfun` to data for full table***
  - Cuts identical applications of this split, ***not dependent on any splitting it is nested within***
- Can be cumulative (`cumulative = TRUE`)

Cut Functions and CutLabel Functions

Cut Function

- Accepts (full-table) data vector for variable
- Returns vector of break points (including lower and upper bound)
  - Like `quantile()`
  - Can be named, if so, labels for upper limit of each cut is used
    - Ignored if cutlabel function is specified

CutLabel Function

- Accepts output of cutfun
- Returns vector of labels to use

Dynamic Cut Examples

```
helper <- function(vec) paste(round(vec, 1),
                              collapse = " - ")

mycutfun <- function(x) {
    ret <- quantile(x, c(0, .33, .66, 1))
    forlab <- floor(ret)
    names(ret) <- c("",
                    helper(ret[1:2]),
                    helper(ret[2:3]),
                    helper(ret[3:4]))
    ret
}
```

```
mycumlabfun <- function(x) c(helper(x[1:2]),
                             helper(x[c(1,3)]),
                             "All")
```

```
lyt_dyncut<- basic_table(show_colcounts = TRUE) %>%
    split_cols_by_cutfun("AGE", cutfun = mycutfun) %>%
    analyze("BMRKR1")
```

```
> build_table(lyt_dyncut, DM)
       20 - 30    30 - 36    36 - 60
       (N=122)    (N=114)    (N=120)
————————————————————————————————————
Mean    5.98       5.67       5.90
```

```
lyt_dyncut_cum <-  basic_table(show_colcounts = TRUE) %>%
    split_cols_by_cutfun("AGE", cutfun = mycutfun,
                         cutlabelfun = mycumlabfun,
                         cumulative = TRUE) %>%
    analyze("BMRKR1")
```

```
> build_table(lyt_dyncut_cum, DM)
       20 - 30    20 - 36      All
       (N=122)    (N=236)    (N=356)
————————————————————————————————————
Mean    5.98       5.83       5.85
```

Mix and Match

All of these split types can be

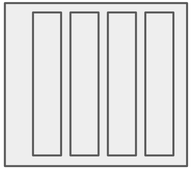
- Used in both column and row space
    - Though again, multivar doesn't make much sense in row space
- Nested within eachother
    - Though nesting splits inside a multivar doesn't make much sense

The main points to take away by now are …

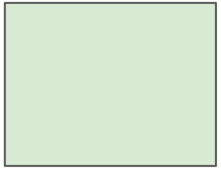
- rables is a sophisticated end 2 end table framework
- tables are faceted visualizations
- rtables tables are created with layouts and data
- layouts declare facets (split_* functions), analyses (analyze function)
- you can read the following code and predict the table structure:

```
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

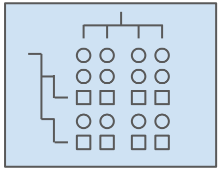
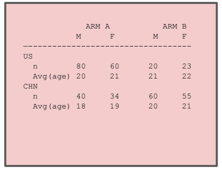


table object
cell values,
table structure,
format instructions,
paths



	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

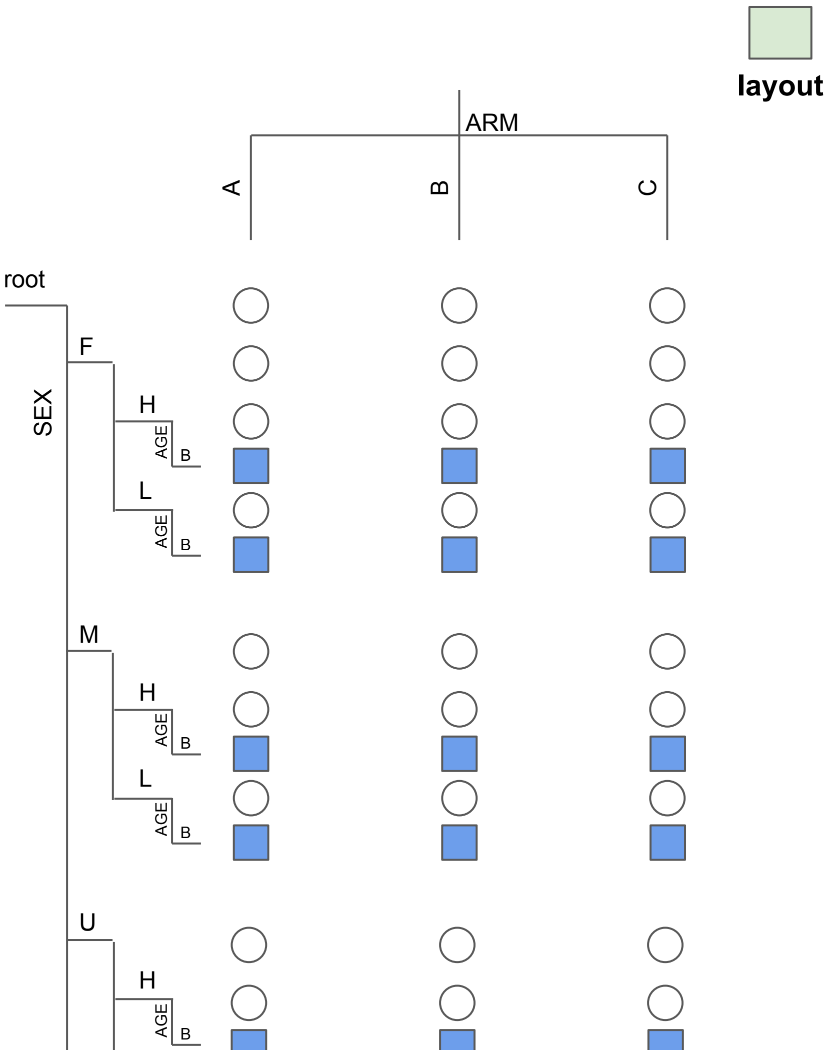
layout

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

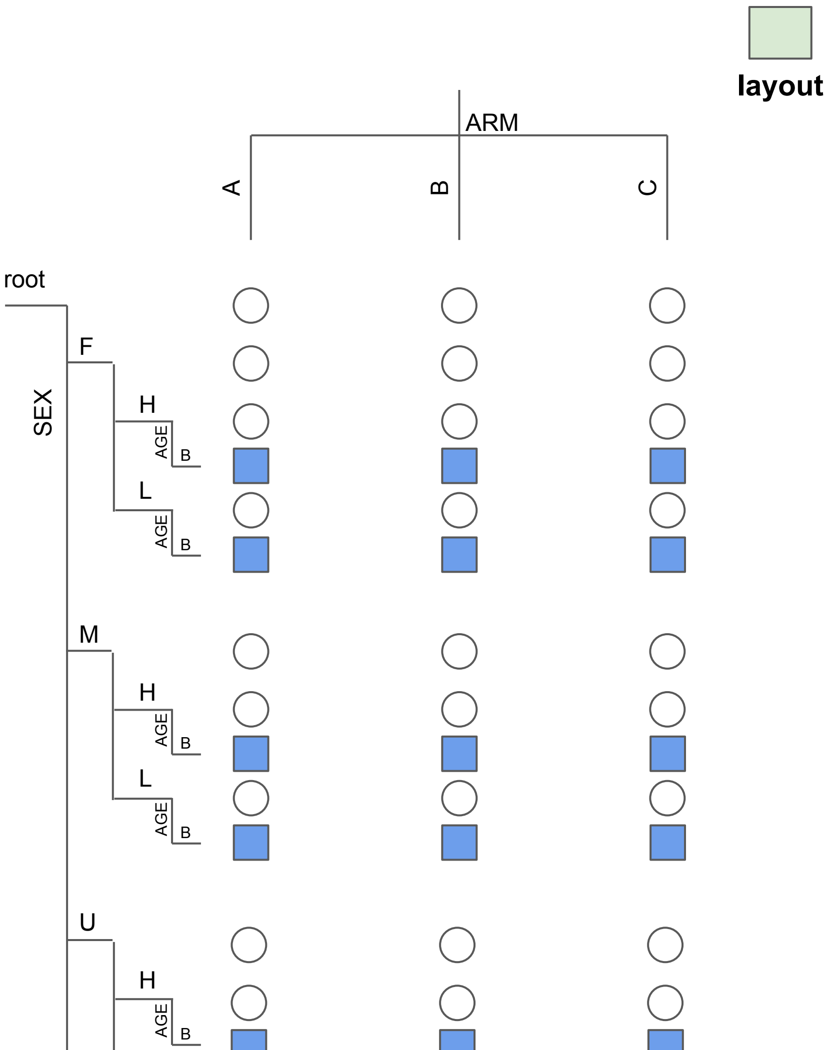
```
lyt <- basic_table() |>
  split_cols_by("ARM") |>
  split_rows_by("SEX") |>
  split_rows_by("B1HL") |>
  analyze("AGE", \(x) list(B = "a"))

build_table(lyt, ex_adsl3)
```



Group summaries

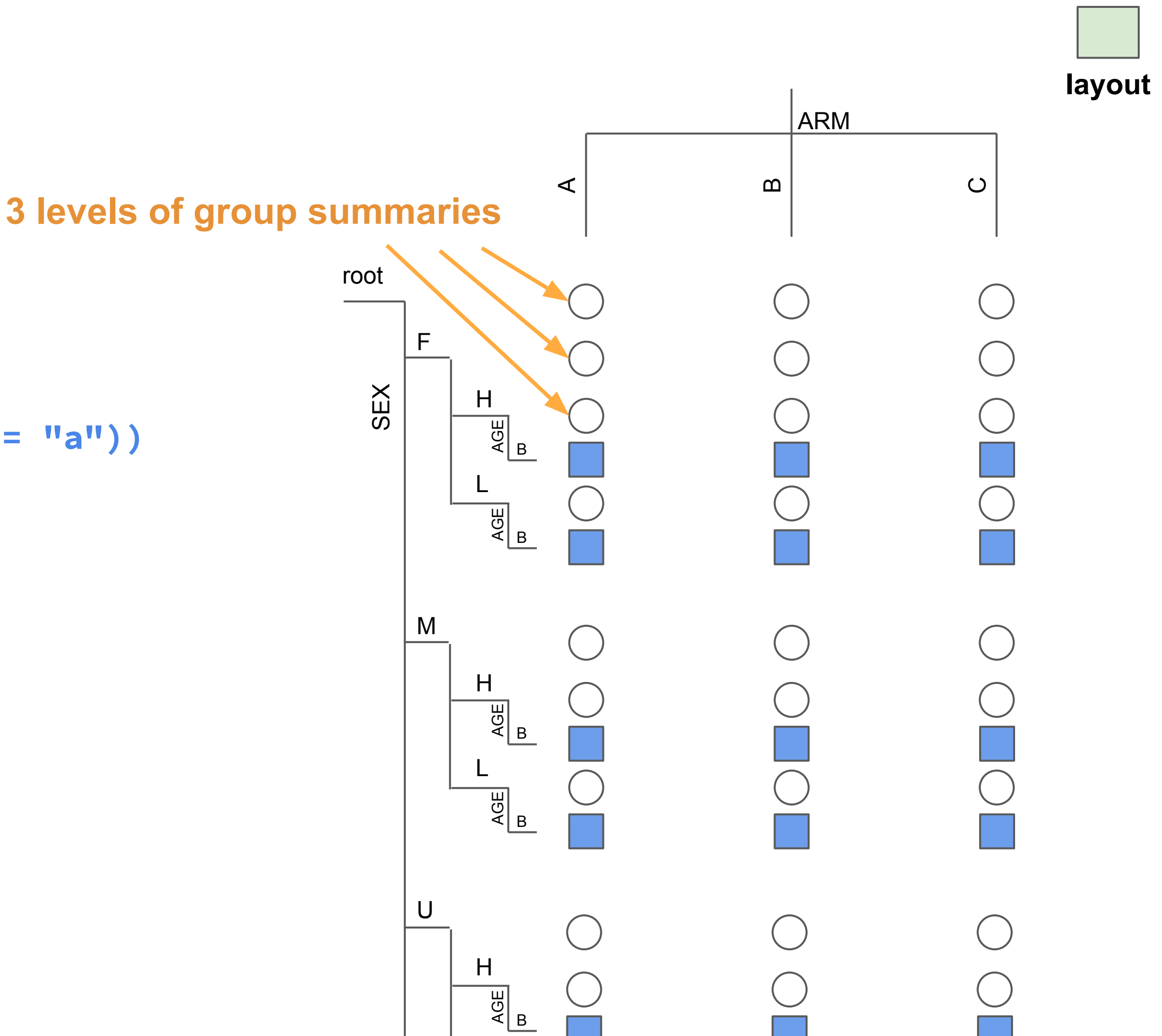
```
lyt <- basic_table() |>
  split_cols_by("ARM") |>
  split_rows_by("SEX") |>
  split_rows_by("B1HL") |>
  analyze("AGE", \(x) list(B = "a"))

build_table(lyt, ex_adsl3)
```

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



Group summaries

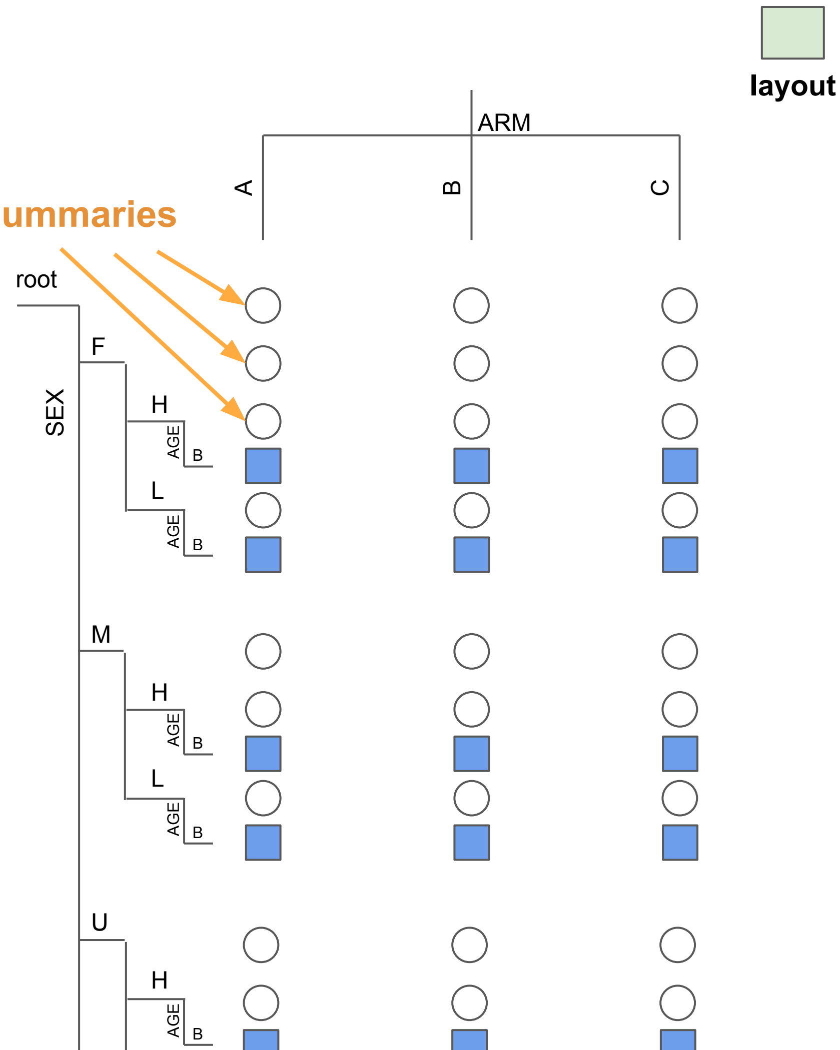
```r
lyt <- basic_table() |>
  split_cols_by("ARM") |>
  split_rows_by("SEX") |>
  split_rows_by("B1HL") |>
  analyze("AGE", \(x) list(B = "a"))

build_table(lyt, ex_adsl3)
```



Group summaries

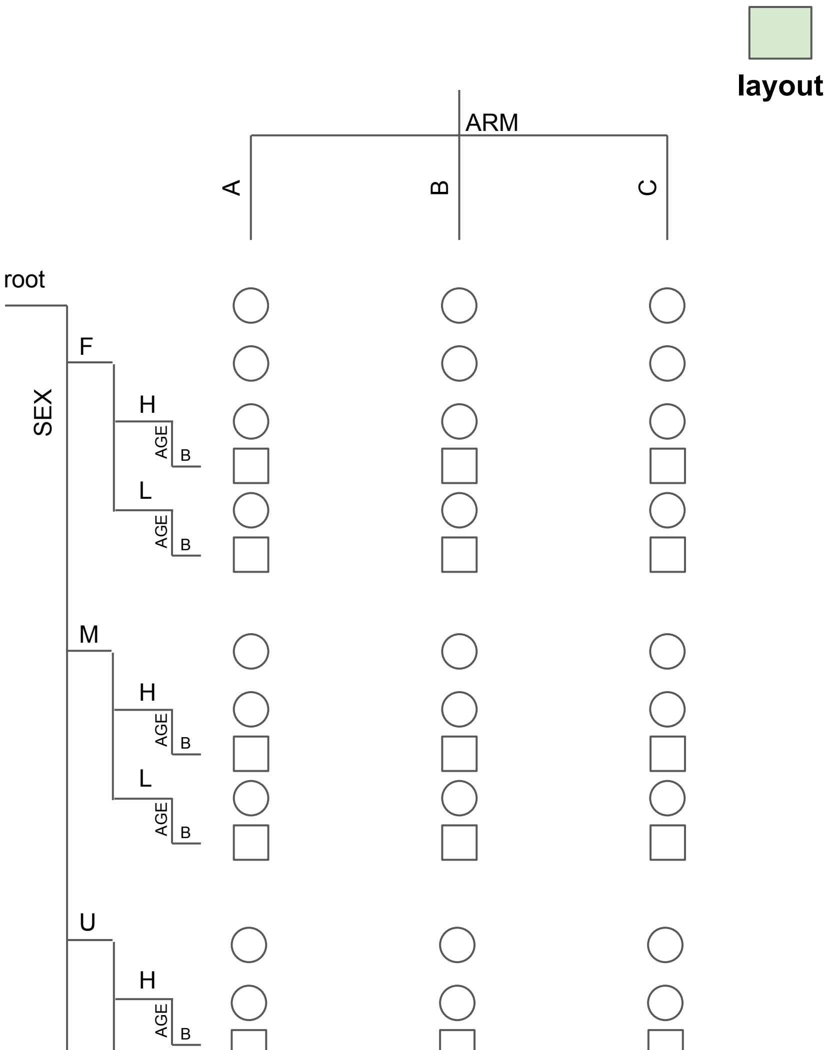
```
lyt <- basic_table() |>
  split_cols_by("ARM") |>
  split_rows_by("SEX") |>
  split_rows_by("B1HL") |>
  analyze("AGE", \(x) list(B = "a"))

build_table(lyt, ex_adsl3)
```

3 levels of group summaries



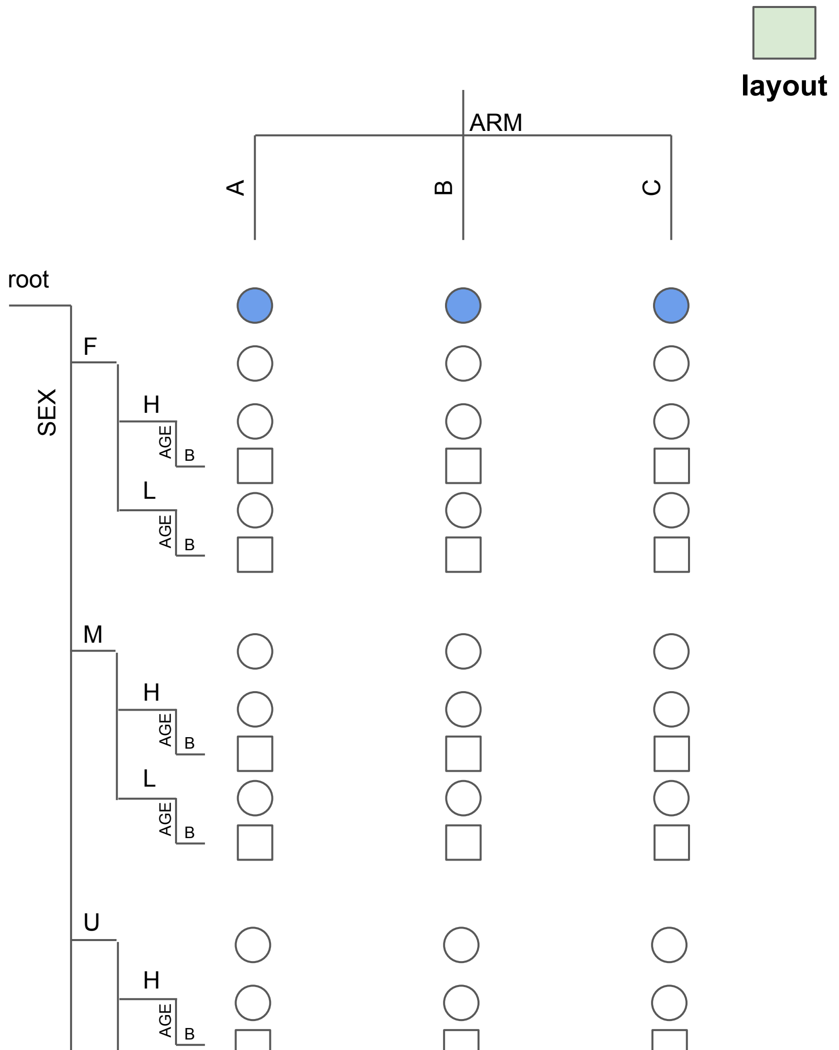
Group summaries

```
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)
```



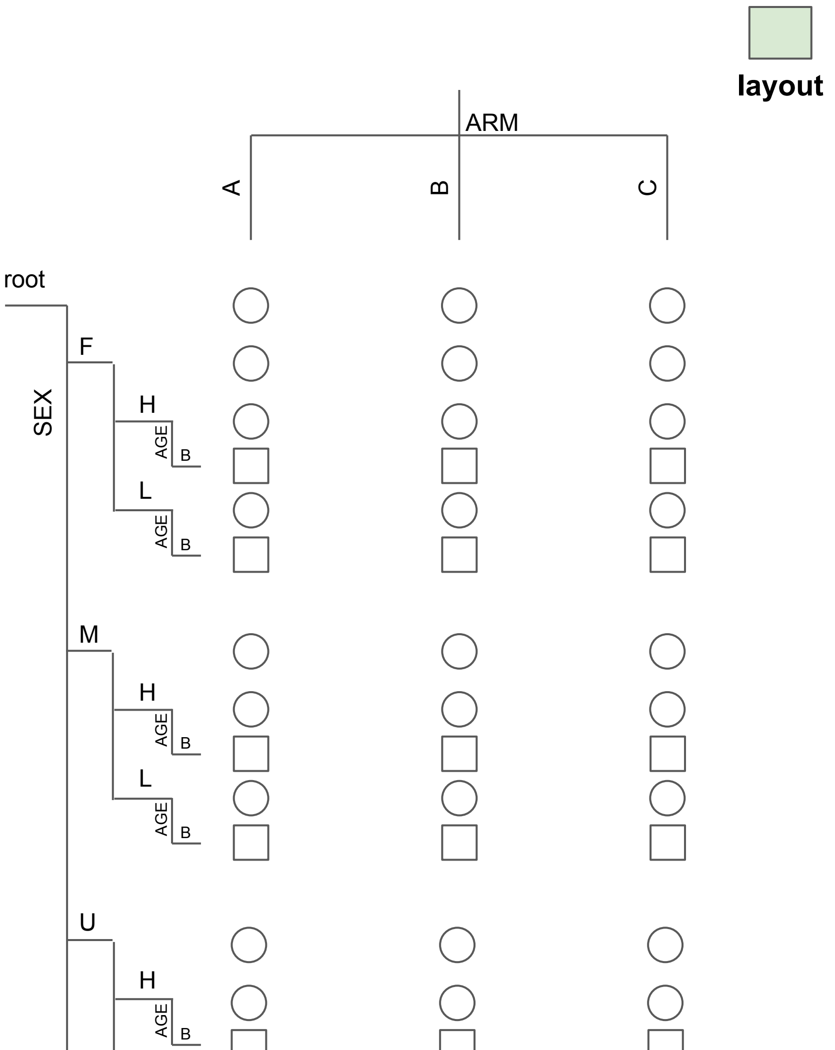
Group summaries

```
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)
```



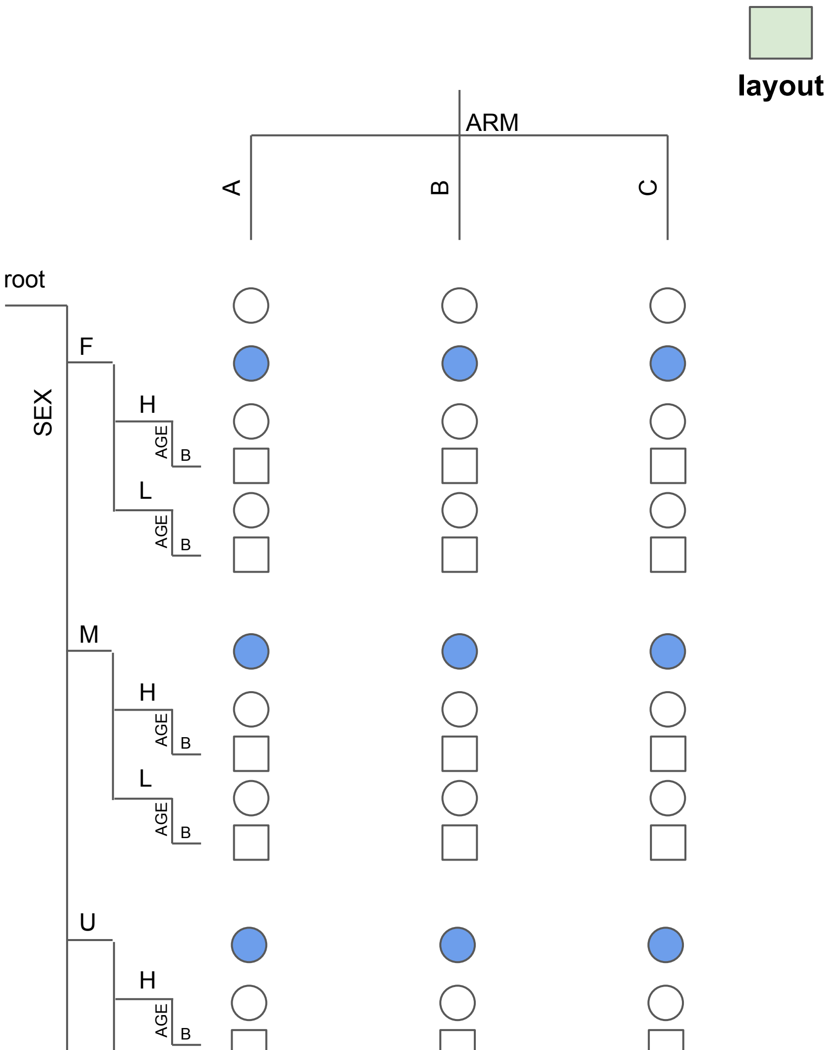
Group summaries

```
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)
```



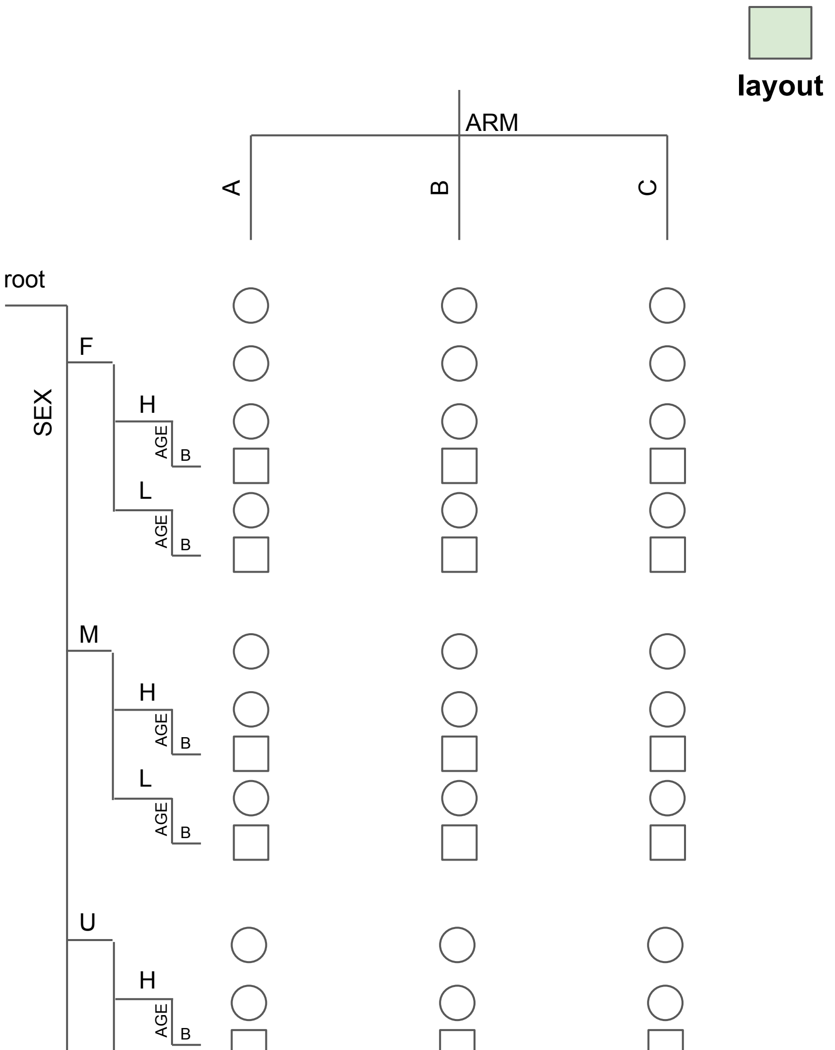
Group summaries

```
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)
```



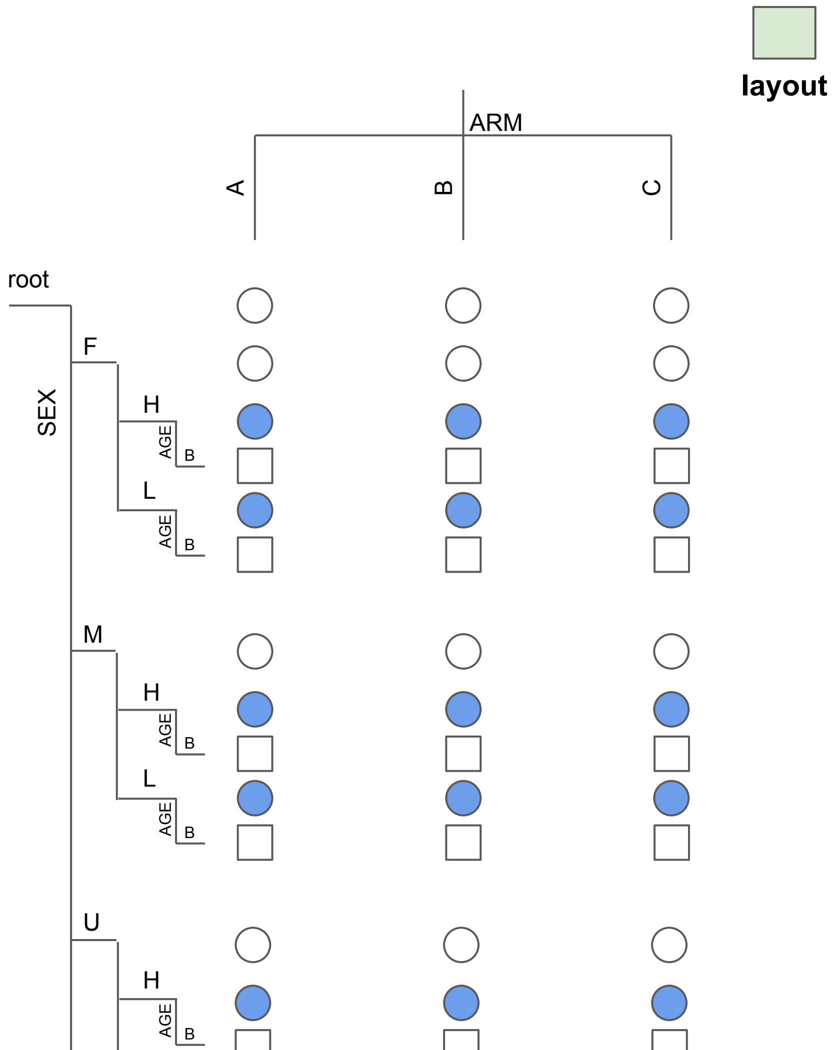
Group summaries

```
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)
```



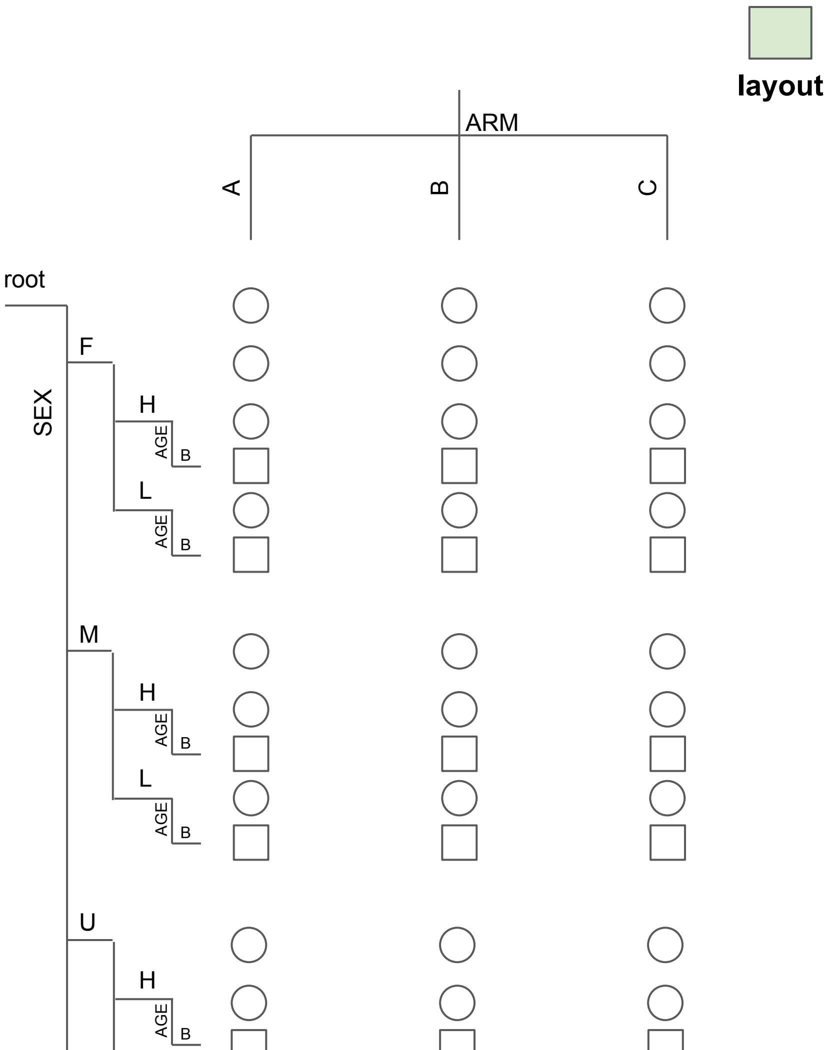
Group summaries

```
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)
```



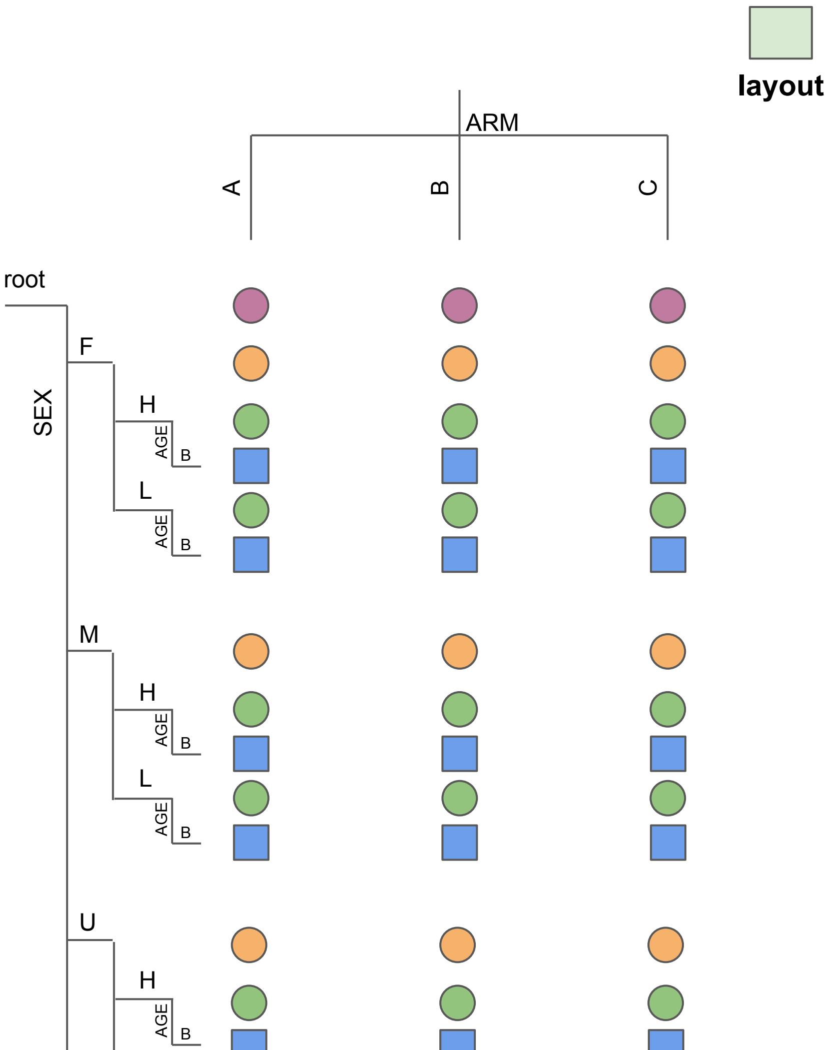
Group summaries

```
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

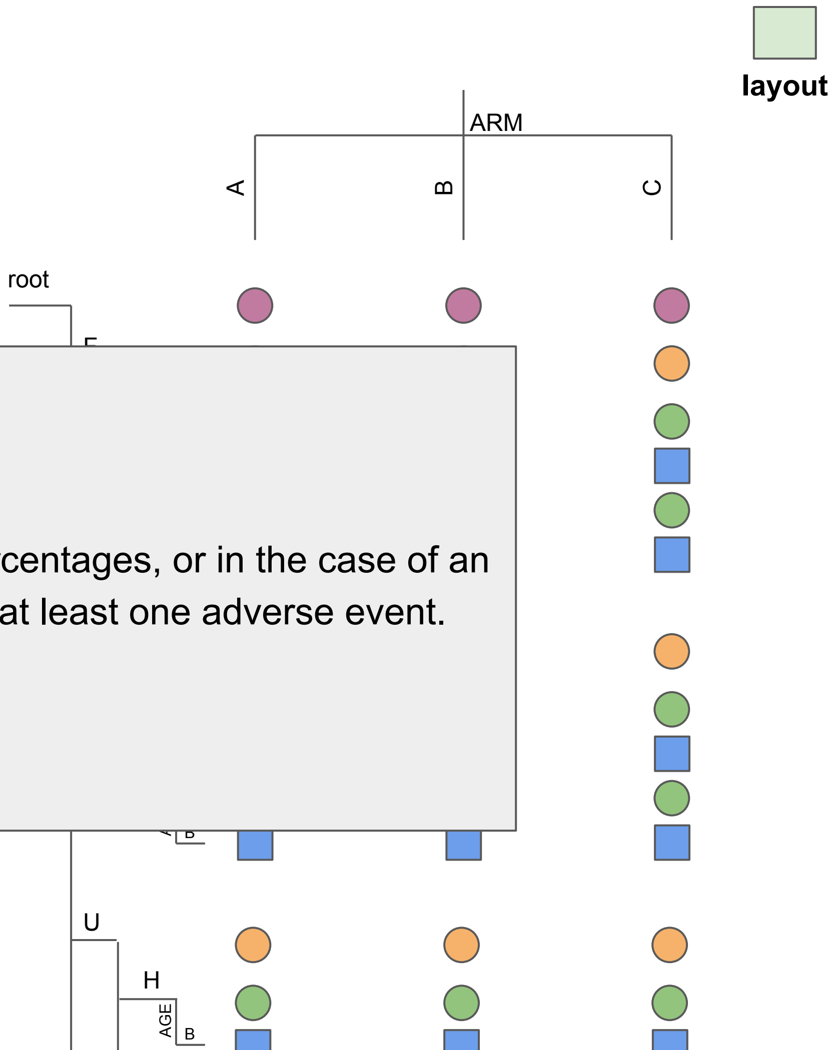
```r
lyt <- basic_table() |>
  split_cols_by("ARM") |>
  summarize_row_groups() |>
  split_rows_by("SEX") |>
  summarize_row_groups() |>
  split_rows_by("B1HL") |>
  summarize_row_groups() |>
  analyze("AGE", afun = \(x) list(B = "a"))

build_table(lyt, ex_adsl3)
```



Group summaries

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

build_table(
```

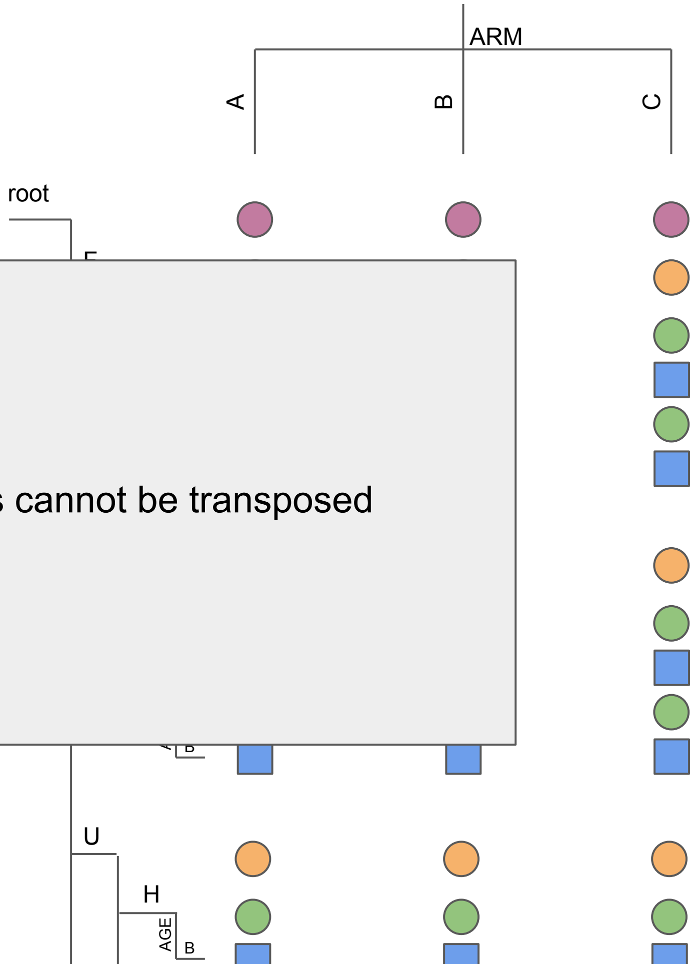
Usually group summaries hold counts, percentages, or in the case of an adverse events table unique patients with at least one adverse event.



Group summaries

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

build_table
```

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



layout

Group Summary -> Content Table/Rows

The table of rows resulting from a `summary_row_group` layout directive is called that facet's **content table**

- Artifact from *very* early in the design process
  - Unfortunate
  - Can't be easily changed at this stage

Table Structure

Table objects

- rtables table objects are implemented with a tree data structure

- Directly inspecting table objects' low-level structure is not going to be helpful

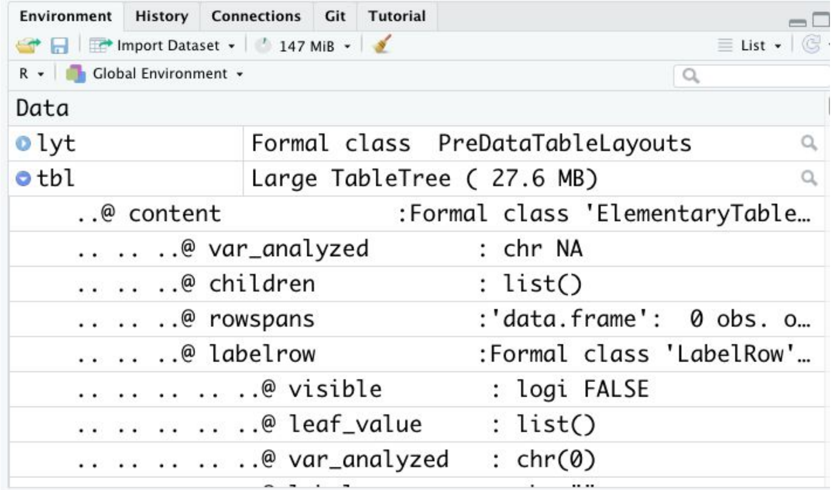


Table objects

- 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

Consider A Non-trivial Table

	ARM1		ARM2	
	Male (N=256)	Female (N=248)	Male (N=248)	Female (N=248)
Caucasian (n)	116 (45.3%)	144 (58.1%)	119 (48.0%)	119 (48.0%)
Level A	37 (14.5%)	48 (19.4%)	42 (16.9%)	35 (14.1%)
Age Analysis				
mean	56.42	55.57	56.19	54.53
median	55.91	55.42	58.40	51.73
Age Analysis redux				
range	40.1 - 69.9	40.8 - 69.7	42.2 - 69.7	41.7 - 69.3
Level B	44 (17.2%)	52 (21.0%)	37 (14.9%)	40 (16.1%)
Age Analysis				
mean	54.28	55.24	54.22	54.92
median	54.71	55.47	54.96	55.17
Age Analysis redux				
range	41.2 - 69.5	40.5 - 69.0	40.0 - 68.5	40.0 - 69.2
African American (n)	140 (54.7%)	104 (41.9%)	129 (52.0%)	129 (52.0%)
Level A	45 (17.6%)	40 (16.1%)	48 (19.4%)	44 (17.7%)
Age Analysis				
mean	55.77	55.33	56.26	54.30
median	55.06	54.39	57.20	53.94
Age Analysis redux				
range	42.7 - 69.6	40.4 - 69.3	41.3 - 69.3	40.7 - 68.9
Level B	45 (17.6%)	29 (11.7%)	44 (17.7%)	55 (22.2%)
Age Analysis				
mean	53.85	56.55	56.80	55.88
median	54.36	57.21	57.39	55.71
Age Analysis redux				
range	40.6 - 68.1	40.4 - 69.3	40.1 - 69.0	40.3 - 69.9
Var3 Counts				
level1	121	136	142	120
level2	135	112	106	128

We render this as a rectangular display, but *model* it with structure that is

1. Semantically meaningful
2. Reflects the layout used to create it

Table Structure

```
> table_structure(tbl3)
[TableTree] root
 [TableTree] RACE
  [TableTree] WHITE [cont: 1 x 4]
   [TableTree] FACTOR2
    [TableTree] A [cont: 1 x 4]
     [ElementaryTable] AGE (2 x 4)
     [ElementaryTable] AgeRedux (1 x 4)
    [TableTree] B [cont: 1 x 4]
     [ElementaryTable] AGE (2 x 4)
     [ElementaryTable] AgeRedux (1 x 4)
  [TableTree] BLACK [cont: 1 x 4]
   [TableTree] FACTOR2
    [TableTree] A [cont: 1 x 4]
     [ElementaryTable] AGE (2 x 4)
     [ElementaryTable] AgeRedux (1 x 4)
    [TableTree] B [cont: 1 x 4]
     [ElementaryTable] AGE (2 x 4)
     [ElementaryTable] AgeRedux (1 x 4)
 [ElementaryTable] VAR3 (2 x 4)
```

Table Structure and Layout

```
> table_structure(tbl3)
[TableTree] root
 [TableTree] RACE
  [TableTree] WHITE [cont: 1 x 4]
   [TableTree] FACTOR2
    [TableTree] A [cont: 1 x 4]
     [ElementaryTable] AGE (2 x 4)
     [ElementaryTable] AgeRedux (1 x 4)
    [TableTree] B [cont: 1 x 4]
     [ElementaryTable] AGE (2 x 4)
     [ElementaryTable] AgeRedux (1 x 4)
  [TableTree] BLACK [cont: 1 x 4]
   [TableTree] FACTOR2
    [TableTree] A [cont: 1 x 4]
     [ElementaryTable] AGE (2 x 4)
     [ElementaryTable] AgeRedux (1 x 4)
    [TableTree] B [cont: 1 x 4]
     [ElementaryTable] AGE (2 x 4)
     [ElementaryTable] AgeRedux (1 x 4)
 [ElementaryTable] VAR3 (2 x 4)
```

```
basic_table(show_colcounts = TRUE) %>%
   split_cols_by("ARM") %>%
   split_cols_by("SEX", "Gender", labels_var = "gend_label") %>%
   split_rows_by("RACE", "Ethnicity",
                 labels_var = "ethn_label") %>%
   summarize_row_groups("RACE", label_fstr = "%s (n)") %>%
   split_rows_by("FACTOR2", "Factor2",
                 split_fun = remove_split_levels("C"),
                 labels_var = "fac2_label",
                 label_pos = "hidden") %>%
   summarize_row_groups("FACTOR2") %>%
   analyze("AGE", "Age Analysis",
           afun = function(x) list(mean = mean(x),
                                   median = median(x)),
           format = "xx.xx") %>%
   analyze("AGE",
           "Age Analysis redux",
           afun = range,
           format = "xx.x - xx.x",
           table_names = "AgeRedux"
           ) %>%
   analyze("VAR3", "Var3 Counts", afun = list_wrap_x(table),
           nested = FALSE,
           show_labels = "visible")
```

Table Structure and Layout

```
> table_structure(tbl3)
[TableTree] root
 [TableTree] RACE
  [TableTree] WHITE [cont: 1 x 4]
   [TableTree] FACTOR2
    [TableTree] A [cont: 1 x 4]
     [ElementaryTable] AGE (2 x 4)
     [ElementaryTable] AgeRedux (1 x 4)
    [TableTree] B [cont: 1 x 4]
     [ElementaryTable] AGE (2 x 4)
     [ElementaryTable] AgeRedux (1 x 4)
  [TableTree] BLACK [cont: 1 x 4]
   [TableTree] FACTOR2
    [TableTree] A [cont: 1 x 4]
     [ElementaryTable] AGE (2 x 4)
     [ElementaryTable] AgeRedux (1 x 4)
    [TableTree] B [cont: 1 x 4]
     [ElementaryTable] AGE (2 x 4)
     [ElementaryTable] AgeRedux (1 x 4)
 [ElementaryTable] VAR3 (2 x 4)
```

```
basic_table(show_colcounts = TRUE) %>%
   split_cols_by("ARM") %>%
   split_cols_by("SEX", "Gender", labels_var = "gend_label") %>%
   split_rows_by("RACE", "Ethnicity",
                 labels_var = "ethn_label") %>%
   summarize_row_groups("RACE", label_fstr = "%s (n)") %>%
   split_rows_by("FACTOR2", "Factor2",
                 split_fun = remove_split_levels("C"),
                 labels_var = "fac2_label",
                 label_pos = "hidden") %>%
   summarize_row_groups("FACTOR2") %>%
   analyze("AGE", "Age Analysis",
           afun = function(x) list(mean = mean(x),
                                   median = median(x)),
           format = "xx.xx") %>%
   analyze("AGE",
           "Age Analysis redux",
           afun = range,
           format = "xx.x - xx.x",
           table_names = "AgeRedux"
           ) %>%
   analyze("VAR3", "Var3 Counts", afun = list_wrap_x(table),
           nested = FALSE,
           show_labels = "visible")
```

Table Structure and Layout

```
> table_structure(tbl3)
[TableTree] root
 [TableTree] RACE
  [TableTree] WHITE [cont: 1 x 4]
   [TableTree] FACTOR2
    [TableTree] A [cont: 1 x 4]
     [ElementaryTable] AGE (2 x 4)
     [ElementaryTable] AgeRedux (1 x 4)
    [TableTree] B [cont: 1 x 4]
     [ElementaryTable] AGE (2 x 4)
     [ElementaryTable] AgeRedux (1 x 4)
  [TableTree] BLACK [cont: 1 x 4]
   [TableTree] FACTOR2
    [TableTree] A [cont: 1 x 4]
     [ElementaryTable] AGE (2 x 4)
     [ElementaryTable] AgeRedux (1 x 4)
    [TableTree] B [cont: 1 x 4]
     [ElementaryTable] AGE (2 x 4)
     [ElementaryTable] AgeRedux (1 x 4)
 [ElementaryTable] VAR3 (2 x 4)
```

```
basic_table(show_colcounts = TRUE) %>%
   split_cols_by("ARM") %>%
   split_cols_by("SEX", "Gender", labels_var = "gend_label") %>%
   split_rows_by("RACE", "Ethnicity",
                 labels_var = "ethn_label") %>%
   summarize_row_groups("RACE", label_fstr = "%s (n)") %>%
   split_rows_by("FACTOR2", "Factor2",
                 split_fun = remove_split_levels("C"),
                 labels_var = "fac2_label",
                 label_pos = "hidden") %>%
   summarize_row_groups("FACTOR2") %>%
   analyze("AGE", "Age Analysis",
           afun = function(x) list(mean = mean(x),
                                   median = median(x)),
           format = "xx.xx") %>%
   analyze("AGE",
           "Age Analysis redux",
           afun = range,
           format = "xx.x - xx.x",
           table_names = "AgeRedux"
           ) %>%
   analyze("VAR3", "Var3 Counts", afun = list_wrap_x(table),
           nested = FALSE,
           show_labels = "visible")
```

Pathing

Pathing

- Semantically descriptive way of describing position in table based on facet structure
    - Individual cells
    - Rows
    - Columns and column groups
    - Subtables
- Enforces valid structure of result when used for subsetting
    - Unlike position-based subsetting

Paths Of Our Table

```
> row_paths_summary(tbl3)
rowname                    node_class    path
_________________________________________________________________________________________
—
Caucasian (n)              ContentRow    root, RACE, WHITE, @content, Caucasian (n)
  Level A                  ContentRow    root, RACE, WHITE, FACTOR2, A, @content, Level A
    Age Analysis           LabelRow      root, RACE, WHITE, FACTOR2, A, AGE
      mean                 DataRow       root, RACE, WHITE, FACTOR2, A, AGE, mean
      median               DataRow       root, RACE, WHITE, FACTOR2, A, AGE, median
    Age Analysis redux     LabelRow      root, RACE, WHITE, FACTOR2, A, AgeRedux
      range                DataRow       root, RACE, WHITE, FACTOR2, A, AgeRedux, range
  Level B                  ContentRow    root, RACE, WHITE, FACTOR2, B, @content, Level B
    Age Analysis           LabelRow      root, RACE, WHITE, FACTOR2, B, AGE
      mean                 DataRow       root, RACE, WHITE, FACTOR2, B, AGE, mean
      median               DataRow       root, RACE, WHITE, FACTOR2, B, AGE, median
    Age Analysis redux     LabelRow      root, RACE, WHITE, FACTOR2, B, AgeRedux
      range                DataRow       root, RACE, WHITE, FACTOR2, B, AgeRedux, range
African American (n)       ContentRow    root, RACE, BLACK, @content, African American
(n)
  Level A                  ContentRow    root, RACE, BLACK, FACTOR2, A, @content, Level A
    Age Analysis           LabelRow      root, RACE, BLACK, FACTOR2, A, AGE
      mean                 DataRow       root, RACE, BLACK, FACTOR2, A, AGE, mean
      median               DataRow       root, RACE, BLACK, FACTOR2, A, AGE, median
    Age Analysis redux     LabelRow      root, RACE, BLACK, FACTOR2, A, AgeRedux
      range                DataRow       root, RACE, BLACK, FACTOR2, A, AgeRedux, range
  Level B                  ContentRow    root, RACE, BLACK, FACTOR2, B, @content, Level B
    Age Analysis           LabelRow      root, RACE, BLACK, FACTOR2, B, AGE
      mean                 DataRow       root, RACE, BLACK, FACTOR2, B, AGE, mean
      median               DataRow       root, RACE, BLACK, FACTOR2, B, AGE, median
    Age Analysis redux     LabelRow      root, RACE, BLACK, FACTOR2, B, AgeRedux
      range                DataRow       root, RACE, BLACK, FACTOR2, B, AgeRedux, range
Var3 Counts                LabelRow      root, VAR3
  level1                   DataRow       root, VAR3, level1
```

```
> col_paths_summary(tbl3)
label        path
_____________________________
ARM1         ARM, ARM1
  Male       ARM, ARM1, SEX, M
  Female     ARM, ARM1, SEX, F
ARM2         ARM, ARM2
  Male       ARM, ARM2, SEX, M
  Female     ARM, ARM2, SEX, F
```

Path Introspection

- `row_paths_summary` - primarily for interactive use, returns `data.frame`
- `row_paths` - returns list of paths, useful programmatically
- `make_row_df` - returns larger amount of information in data.frame
  - paths in the `path` column

Example of Pathing

```
> table_structure(tbl3)
[TableTree] root
 [TableTree] RACE
  [TableTree] WHITE [cont: 1 x 4]
   [TableTree] FACTOR2
    [TableTree] A [cont: 1 x 4]
     [ElementaryTable] AGE (2 x 4)
     [ElementaryTable] AgeRedux (1 x 4)
    [TableTree] B [cont: 1 x 4]
     [ElementaryTable] AGE (2 x 4)
     [ElementaryTable] AgeRedux (1 x 4)
  [TableTree] BLACK [cont: 1 x 4]
   [TableTree] FACTOR2
    [TableTree] A [cont: 1 x 4]
     [ElementaryTable] AGE (2 x 4)
     [ElementaryTable] AgeRedux (1 x 4)
    [TableTree] B [cont: 1 x 4]
     [ElementaryTable] AGE (2 x 4)
     [ElementaryTable] AgeRedux (1 x 4)
 [ElementaryTable] VAR3 (2 x 4)
```

```
> cell_values(tbl3,  c("root", "RACE", "WHITE", "FACTOR2", "B", "AgeRedux"))
$ARM1.M
[1] 41.21257 69.45142

$ARM1.F
[1] 40.54095 68.99051

$ARM2.M
[1] 40.03586 68.45793

$ARM2.F
[1] 40.04115 69.21873
```

```
> tt_at_path(tbl3,c("root", "RACE", "WHITE", "FACTOR2", "B", "AgeRedux"))
                             ARM1                        ARM2
                     Male         Female         Male         Female
                     (N=256)      (N=248)        (N=248)      (N=248)
-----------------------------------------------------------------------
Age Analysis redux
  range            41.2 - 69.5  40.5 - 69.0  40.0 - 68.5  40.0 - 69.2
```

Pathing To Subtables vs Pathing To Content Tables

Selecting subtable

```
> tt_at_path(tbl, c("root", "RACE", "WHITE"))
                                 ARM1                        ARM2
                         Male          Female          Male          Female
                       (N=256)        (N=248)        (N=248)        (N=248)
—————————————————————————————————————————————————————————————————————————————
Caucasian (n)         116 (45.3%)   144 (58.1%)   119 (48.0%)   119 (48.0%)
  Level A             37 (14.5%)    48 (19.4%)    42 (16.9%)    35 (14.1%)
    Age Analysis
      mean               56.42         55.57         56.19         54.53
      median             55.91         55.42         58.40         51.73
    Age Analysis redux
      range           40.1 - 69.9   40.8 - 69.7   42.2 - 69.7   41.7 - 69.3
  Level B             44 (17.2%)    52 (21.0%)    37 (14.9%)    40 (16.1%)
    Age Analysis
      mean               54.28         55.24         54.22         54.92
      median             54.71         55.47         54.96         55.17
    Age Analysis redux
      range           41.2 - 69.5   40.5 - 69.0   40.0 - 68.5   40.0 - 69.2
```

Selecting content (group summary)

```
> tt_at_path(tbl, c("root", "RACE", "WHITE", "@content"))
                            ARM1                        ARM2
                    Male          Female          Male          Female
                  (N=256)        (N=248)        (N=248)        (N=248)
————————————————————————————————————————————————————————————————————————
Caucasian (n)    116 (45.3%)   144 (58.1%)   119 (48.0%)   119 (48.0%)
```

Layout <-> Structure <-> Pathing

- Layout maps directly (and predictably) to table structure
- Pathing directly describes table structure, therefor
- **Layout maps directly (and predictably) to paths, and vice versa**

Layout <-> Structure <-> Pathing

- Layout maps directly (and predictably) to table structure
- Pathing directly describes table structure, therefor
- **Layout maps directly (and predictably) to paths, and vice versa**

(Semi-) Advanced Topics

Tables With Subsections

Tables Can Have Multiple Top-level Sections

- With no row splitting, this is just multiple analyze calls
  - Or one `analyze` call with multiple vars
- With row splitting, this is done by adding new ***non-nested*** layouting instructions
  - `analyze(..., nested = FALSE)`
  - `split_rows_by(..., nested = FALSE)` **or**
  - `split_rows_by(...)` directly after `analyze()`

```
lyt_subtabs <- basic_table() %>%
    split_cols_by("ARM") %>%
    split_rows_by("SEX", split_fun = drop_split_levels) %>%
    analyze("AGE") %>%
    split_rows_by("RACE",
        split_fun = keep_split_levels(c("ASIAN", "WHITE"))) %>%
    split_rows_by("SEX", split_fun = drop_split_levels) %>%
    analyze("AGE")
```

```
> build_table(lyt_subtabs, DM)
           A: Drug X   B: Placebo   C: Combination
——————————————————————————————————————————————————
F
  Mean       33.71       33.84          34.89
M
  Mean       36.55       32.10          34.28
ASIAN
  F
    Mean     33.55       34.00          34.90
  M
    Mean     35.03       31.10          34.39
WHITE
  F
    Mean     35.88       38.57          36.50
  M
    Mean     44.00       35.29          34.00
```

```r
lyt_subtabs <- basic_table() %>%
    split_cols_by("ARM") %>%
    split_rows_by("SEX", split_fun = drop_split_levels) %>%
    analyze("AGE") %>%
    split_rows_by("RACE",
        split_fun = keep_split_levels(c("ASIAN", "WHITE"))) %>%
    split_rows_by("SEX", split_fun = drop_split_levels) %>%
    analyze("AGE")
```

```
> build_table(lyt_subtabs, DM)
          A: Drug X   B: Placebo   C: Combination
—————————————————————————————————————————————————
F
  Mean      33.71        33.84          34.89
M
  Mean      36.55        32.10          34.28
ASIAN
  F
    Mean    33.55        34.00          34.90
  M
    Mean    35.03        31.10          34.39
WHITE
  F
    Mean    35.88        38.57          36.50
  M
    Mean    44.00        35.29          34.00
```

```
> table_structure(tbl)
[TableTree] root
 [TableTree] SEX
  [TableTree] F
   [ElementaryTable] AGE (1 x 3)
  [TableTree] M
   [ElementaryTable] AGE (1 x 3)
 [TableTree] RACE
  [TableTree] ASIAN
   [TableTree] SEX
    [TableTree] F
     [ElementaryTable] AGE (1 x 3)
    [TableTree] M
     [ElementaryTable] AGE (1 x 3)
  [TableTree] WHITE
   [TableTree] SEX
    [TableTree] F
     [ElementaryTable] AGE (1 x 3)
    [TableTree] M
     [ElementaryTable] AGE (1 x 3)
```

Generally Don’t Need `rbind`

- Modeling the different subtables in the layout is cleaner
  - Nice paths
  - Same details during pagination
- Rare exceptions:
  - Subtables derived from different base data
  - Different column structure
  - *In both cases, this results in a table whose structure doesn’t match its presentation*

Overall, using rbind to construct tables should be avoided unless you *actually* need it, which you almost always don’t

Controlling Splitting Behavior Via Provided Functions

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 =)`**

Manipulating Factor Levels

We leave examples of these to the reader

- **`drop_split_levels`**
- **`remove_split_levels`**`(excl = <>)`
- **`drop_and_remove_levels`**`(excl = <>)`
- **`keep_split_levels`**`(only = <>, reorder = <>)`
- **`reorder_split_levels`**`(neworder = <>, newlabels = <>, drlevels = <>)`

Preventing extraneous 0-rows in nested splits

`trim_levels_in_group(innervar = <>)`

Drop unobserved levels of variable `innervar` ***independently within each facet generated by this split***

- Used to control levels of `innervar` going into a nested split or analyze
- In practice, this controls (`splitvar, innervar`) value pairs

trim_levels_in_group(innervar)

```
> lyt <- basic_table() %>%
+     split_rows_by("outer_fac") %>%
+     split_rows_by("inner_fac") %>%
+     analyze("value", mean, format = "xx.x")
> build_table(lyt, dat)
             all obs
——————————————————————
A
  A1
    mean      1.0
  A2
    mean      1.8
  B1
    mean      NA
  B2
    mean      NA
  global
    mean      5.4
B
  A1
    mean      NA
  A2
    mean      NA
  B1
    mean      2.9
  B2
    mean      4.0
  global
    mean      4.8
```

Unobserved Value Pairs

trim_levels_in_group(innervar)

```
> lyt <- basic_table() %>%
+     split_rows_by("outer_fac") %>%
+     split_rows_by("inner_fac") %>%
+     analyze("value", mean, format = "xx.x")
> build_table(lyt, dat)
               all obs
______________________
A
  A1
    mean       1.0
  A2
    mean       1.8
  B1
    mean       NA
  B2
    mean       NA
  global
    mean       5.4
B
  A1
    mean       NA
  A2
    mean       NA
  B1
    mean       2.9
  B2
    mean       4.0
  global
    mean       4.8
```

Unobserved Value Pairs

```
> lyt2 <- basic_table() %>%
+     split_rows_by("outer_fac", split_fun = trim_levels_in_group("inner_fac")) %>%
+     split_rows_by("inner_fac") %>%
+     analyze("value", mean, format = "xx.x")
> build_table(lyt2, dat)
               all obs
______________________
A
  A1
    mean       1.0
  A2
    mean       1.8
  global
    mean       5.4
B
  B1
    mean       2.9
  B2
    mean       4.0
  global
    mean       4.8
```

Full Control Of Nested Facet Value Combinations

`trim_levels_to_map(map)`

Fully restrict facet-value-combination space to a pre-specified set of combinations across any number of variables ***regardless of whether a combination is observed in the data***

- Used when some combinations are nonsensical but other combinations are rare but should be reported
    - E.g., adverse events tables with multiple levels of summary

trim_levels_to_map

```
>     lyt <- basic_table() %>%
+         split_rows_by("LBCAT") %>%
+         split_rows_by("PARAMCD") %>%
+         analyze("ANRIND")
> build_table(lyt, ex_adlb)
                all obs
————————————————————
CHEMISTRY
  ALT
    HIGH         279
    LOW          260
    NORMAL      2261
  CRP
    HIGH         293
    LOW          271
    NORMAL      2236
  IGA
    HIGH          0
    LOW           0
    NORMAL        0
IMMUNOLOGY
  ALT
    HIGH          0
    LOW           0
    NORMAL        0
  CRP
    HIGH          0
    LOW           0
    NORMAL        0
  IGA
    HIGH         278
    LOW          286
    NORMAL      2236
```

trim_levels_to_map

```
>     lyt <- basic_table() %>%
+         split_rows_by("LBCAT") %>%
+         split_rows_by("PARAMCD") %>%
+         analyze("ANRIND")
> build_table(lyt, ex_adlb)
                    all obs
———————————————————————————
CHEMISTRY
  ALT
    HIGH          279
    LOW           260
    NORMAL       2261
  CRP
    HIGH          293
    LOW           271
    NORMAL       2236
  IGA
    HIGH           0
    LOW            0
    NORMAL         0
IMMUNOLOGY
  ALT
    HIGH           0
    LOW            0
    NORMAL         0
  CRP
    HIGH           0
    LOW            0
    NORMAL         0
  IGA
    HIGH          278
    LOW           286
    NORMAL       2236
```

```
> map
       LBCAT PARAMCD ANRIND
1  CHEMISTRY     ALT    LOW
2  CHEMISTRY     CRP    LOW
3  CHEMISTRY     CRP   HIGH
4 IMMUNOLOGY     IGA   HIGH

> lyt2 <- basic_table() %>%
+         split_rows_by("LBCAT") %>%
+         split_rows_by("PARAMCD", split_fun = trim_levels_to_map(map = map)) %>%
+         analyze("ANRIND")
>  build_table(lyt2, ex_adlb)
              all obs
—————————————————————
CHEMISTRY
  ALT
    LOW         260
  CRP
    LOW         271
    HIGH        293
IMMUNOLOGY
  IGA
    HIGH        278
```

trim_levels_to_map

```
>     lyt <- basic_table() %>%
+         split_rows_by("LBCAT") %>%
+         split_rows_by("PARAMCD") %>%
+         analyze("ANRIND")
> build_table(lyt, ex_adlb)
                  all obs
————————————————————————————
CHEMISTRY
  ALT
    HIGH         279
    LOW          260
    NORMAL      2261
  CRP
    HIGH         293
    LOW          271
    NORMAL      2236
  IGA
    HIGH          0
    LOW           0
    NORMAL        0
IMMUNOLOGY
  ALT
    HIGH          0
    LOW           0
    NORMAL        0
  CRP
    HIGH          0
    LOW           0
    NORMAL        0
  IGA
    HIGH         278
    LOW          286
    NORMAL      2236
```

```
> map
       LBCAT PARAMCD ANRIND
1  CHEMISTRY     ALT    LOW
2  CHEMISTRY     CRP    LOW
3  CHEMISTRY     CRP   HIGH
4 IMMUNOLOGY     IGA   HIGH
```

Allowed Value Combinations

```
> lyt2 <- basic_table() %>%
+           split_rows_by("LBCAT") %>%
+           split_rows_by("PARAMCD", split_fun = trim_levels_to_map(map = map)) %>%
+           analyze("ANRIND")
>  build_table(lyt2, ex_adlb)
              all obs
————————————————————————
CHEMISTRY
  ALT
    LOW         260
  CRP
    LOW         271
    HIGH        293
IMMUNOLOGY
  IGA
    HIGH        278
```

Adding Combination Levels

`add_combo_levels(combodf)`,
`add_overall_level(valname)` (special case)

Add combination levels (overall level is a special case w/ convenience function) to existing levels of a split, optionally also excluding some of the original levels

add_combo_levels(combodf)

Combinations are declared in a data.frame/tribble with the following columns:

- `valname` - name of the combined value (and thus the generated facet)
- `label` - label for the generated facet
- `levelcombo` - character vector of values to combine (typically factor levels of underlying var)
- `exargs` - a list of extra arguments corresponding to the split level (usually empty list)

```
>  combodf <- tribble(
+      ~valname, ~label,     ~levelcombo,                          ~exargs,
+      "A_B",    "Arms A+B", c("A: Drug X", "B: Placebo"),       list(),
+      "A_C",    "Arms A+C", c("A: Drug X", "C: Combination"), list())
>
>  lyt <- basic_table(show_colcounts = TRUE) %>%
+      split_cols_by("ARM", split_fun = add_combo_levels(combodf)) %>%
+      analyze("AGE")
>
>  build_table(lyt, DM)
           A: Drug X   B: Placebo   C: Combination   Arms A+B   Arms A+C
            (N=121)     (N=106)        (N=129)        (N=227)    (N=250)
——————————————————————————————————————————————————————————————————————————
Mean         34.91       33.02          34.57          34.03      34.73
```

add_combo_levels(combodf)

Combinations are declared in a data.frame/tribble with the following columns:

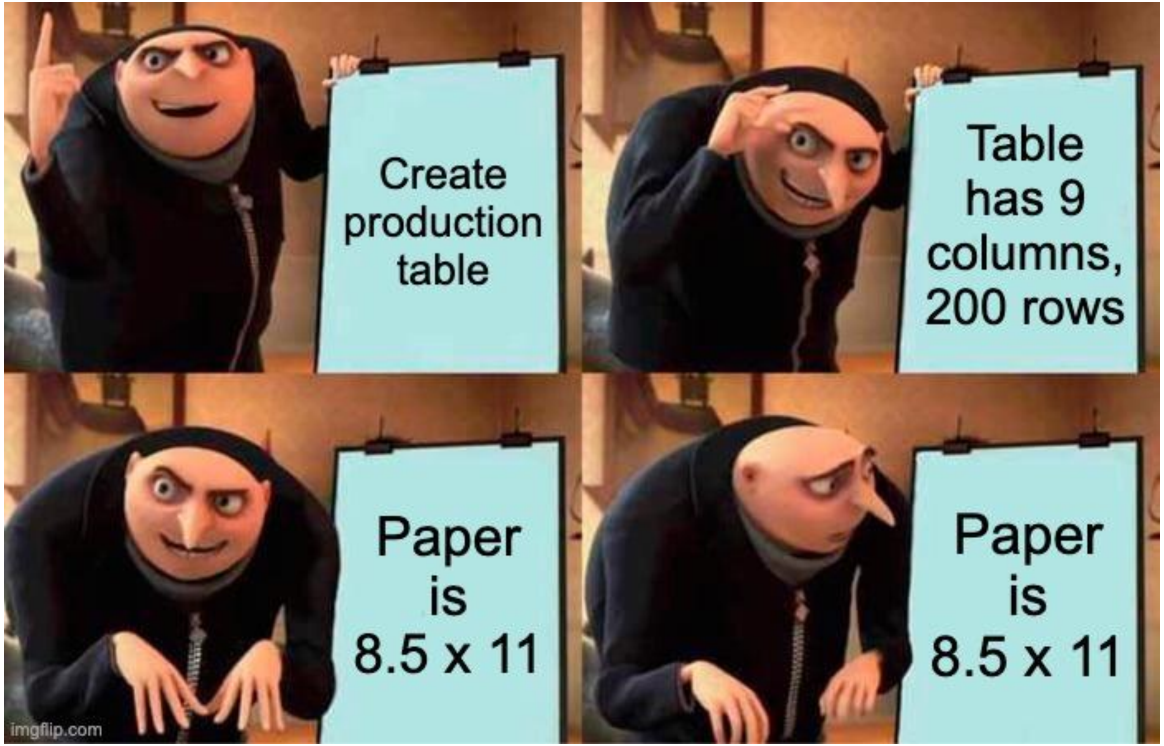
- `valname` - name of the combined value (and thus the generated facet)
- `label` - label for the generated facet
- `levelcombo` - character vector of values to combine (typically factor levels of underlying var)
- `exargs` - a list of extra arguments corresponding to the split level (usually empty list)

```
>  combodf <- tribble(
+      ~valname, ~label,     ~levelcombo,                        ~exargs,
+      "A_B",    "Arms A+B", c("A: Drug X", "B: Placebo"),      list(),
+      "A_C",    "Arms A+C", c("A: Drug X", "C: Combination"), list())
>
>  lyt <- basic_table(show_colcounts = TRUE) %>%
+      split_cols_by("ARM", split_fun = add_combo_levels(combodf)) %>%
+      analyze("AGE")
>
>  build_table(lyt, DM)
          A: Drug X   B: Placebo   C: Combination   Arms A+B   Arms A+C
           (N=121)     (N=106)        (N=129)        (N=227)    (N=250)
-----------------------------------------------------------------------
Mean       34.91       33.02          34.57          34.03      34.73
```

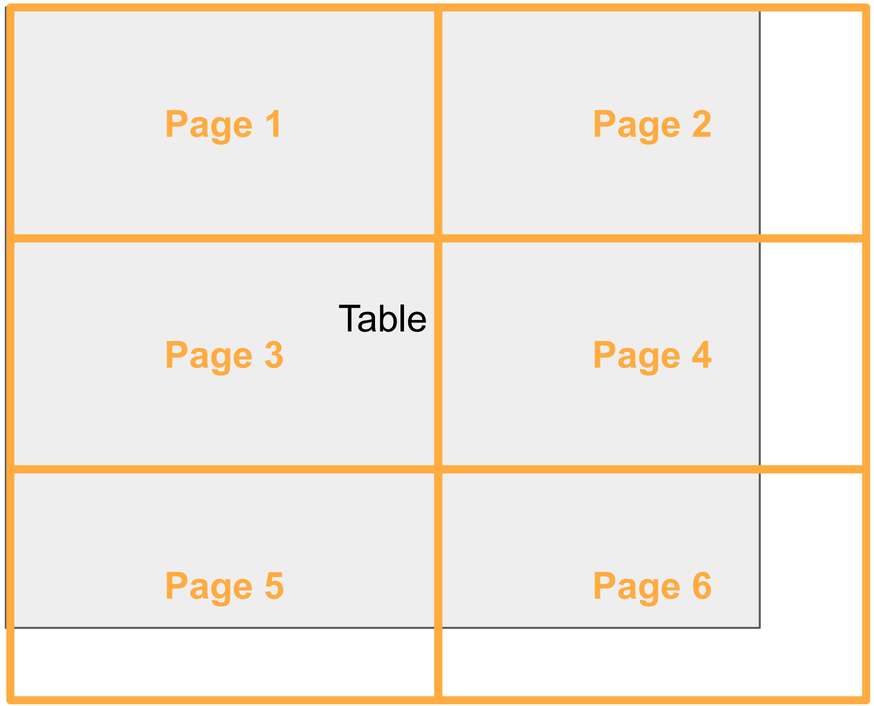
One new level per row.

N's handled automatically

Pagination

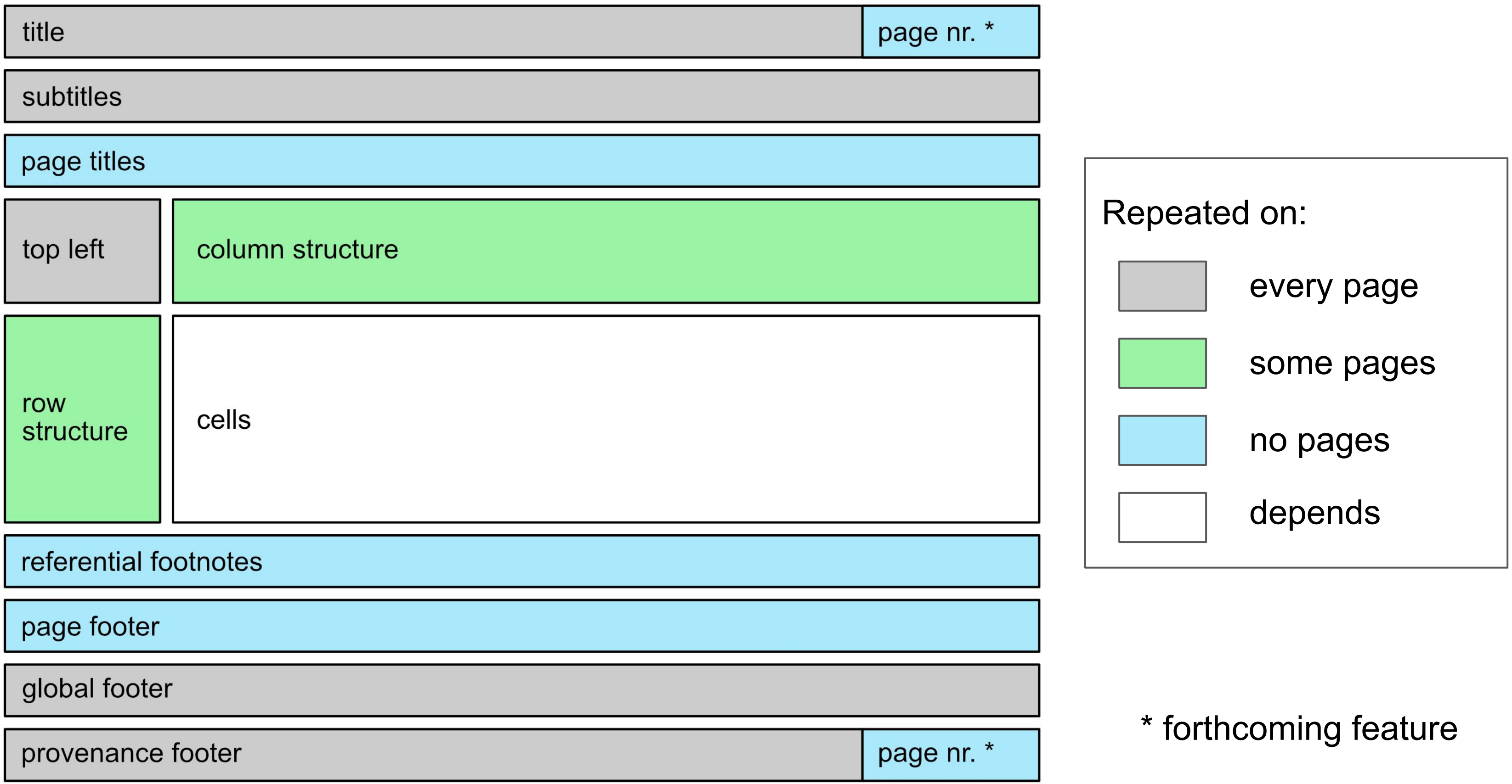

Create production table
Table has 9 columns, 200 rows
Paper is 8.5 x 11
Paper is 8.5 x 11
imgflip.com

Pagination Problem



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

Context-Preserving Pagination



Pagination Specifications

- Vertical Pagination repeats context information after page breaks
    - Content and label rows
- Pagination happens *after* wordwrapping
- Horiz pagination is identical across all sections of vert pagination
- Horizontal and Vertical pagination can be done separately, but
    - Both performed when page dimensions are defined
- Pagination assumes monospaced fonts
- ***Use `verbose = TRUE` for debugging when pagination fails***

Specifying page-size and font

- Page Dimensions
  - `page_type`
    - `letter, a4, legal`
    - Can be used with `landscape = TRUE`
  - `pg_width`, `pg_height` (in inches)
  - `lpp`, `cpp` (raw height in lines, width in chars)
- Font
  - `font_size`
  - `font_family` (must be monospaced)

Converted to lines and characters:

```
page_lcpp(page_type = "legal",
          font_size = 10,
          font_family = "Courier")
$cpp
[1] 84

$lpp
[1] 93
```

Invoking Pagination

- Directly: `paginate_table`
    - Returns list of tables representing the pages (width first ordering)
- Indirectly (`export_as_txt(page_type = “a4”)`)
    - Exports file containing paginated version of the table
    - txt, pdf, rtf exporters support pagination

Page-by Row Splits

- Force pagination between levels of a row-split
  - `split_rows_by("varname", page_by = TRUE)`
  - `page_prefix` - appended with split value label to create page titles
- Page-by row splits cannot be nested within non-page_by row splits
- Page-by pagination happens *before* size-based vertical pagination
  - Horizontal pagination is unaffected
  - Vertical pagination is performed separately within each ‘page’ defined by full set of page-by splits

Page-by Row Splitting

```
lyt <- basic_table(title = "Main Title",
                   subtitles = "Subtitle") |>
    split_cols_by("ARM") |>
    split_rows_by("SEX",
                  page_by = TRUE,
                  page_prefix = "Sex",
                  split_fun = drop_split_levels) |>
    analyze("AGE", mean)

tbl <- build_table(lyt, DM)
```

```
> paginate_table(tbl)
$F
Main Title
Subtitle
Sex: F

——————————————————————————————————————————————
       A: Drug X   B: Placebo   C: Combination
——————————————————————————————————————————————
mean     33.7         33.8           34.9

$M
Main Title
Subtitle
Sex: M

——————————————————————————————————————————————
       A: Drug X   B: Placebo   C: Combination
——————————————————————————————————————————————
mean     36.5         32.1           34.3
```

Note about Page-by row splits

Currently forced pagination happens *at pagination time*, meaning it won't show up when you print the table without pagination:

```
> tbl
Main Title
Subtitle

————————————————————————————————————————————————
         A: Drug X    B: Placebo   C: Combination
————————————————————————————————————————————————
F
  mean     33.7          33.8             34.9
M
  mean     36.5          32.1             34.3
```

This may change in future releases

Customizing Appearance And Rendering Behavior

Section Dividers

Horizontal dividers (incl. blank line) placed after row groups

```
lyt <- basic_table(title = "Main Title",
            subtitles = "Subtitle") %>%
    split_cols_by("ARM") %>%
    split_rows_by("STRATA1", section_div = "-") %>%
    split_rows_by("SEX", section_div = " ",
                  split_fun = drop_split_levels) %>%
    analyze("AGE", mean, format = "xx.x")

tbl <- build_table(lyt, DM)
```

```
> tbl
Main Title
Subtitle

———————————————————————————————————————————————————
             A: Drug X   B: Placebo   C: Combination
———————————————————————————————————————————————————
A
  F
    mean       30.9         32.9          36.0

  M
    mean       35.1         31.1          35.6
---------------------------------------------------
B
  F
    mean       34.9         32.9          34.4

  M
    mean       36.6         32.1          34.4
---------------------------------------------------
C
  F
    mean       35.2         36.0          34.3

  M
    mean       37.4         32.8          32.8
```

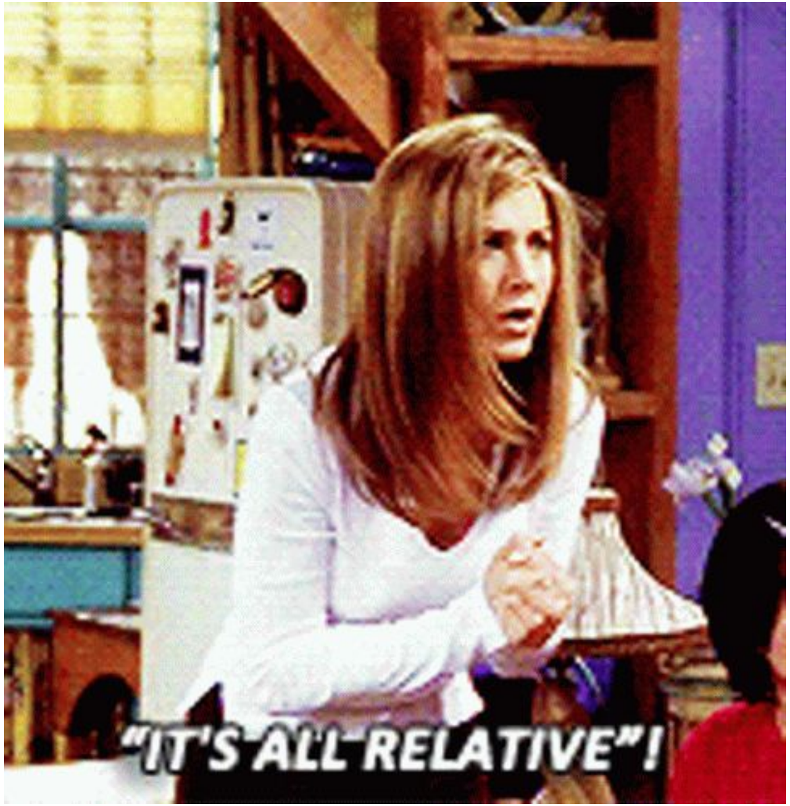
Section dividers

- Can be different for different levels of faceting
- Can be blank lines (use " ")
- When multiple section dividers would apply, only the one for the largest group is printed
- No section divider is ever printed at the end of the table body

Indent Modification

- Indenting happens automatically
  - Including visible label and hidden label cases
- Indent levels can be modified at the layout stage
  - `indent_mod` argument to, well, most things

Understanding `indent_mod`



Normal Table

```
lyt <- basic_table() %>%
    split_rows_by("STRATA1") %>%
    split_rows_by("SEX",
                  split_fun = drop_split_levels) %>%
    analyze("AGE", mean, format = "xx.x")

tbl1 <- build_table(lyt, DM)
```

```
> tbl1
              all obs
_____________________
A
  F
    mean       33.2
  M
    mean       34.5
B
  F
    mean       34.2
  M
    mean       34.0
C
  F
    mean       35.1
  M
    mean       34.5
```

Indent Mods

```
lyt1 <- basic_table() %>%
    split_rows_by("STRATA1") %>%
    split_rows_by("SEX",
                  split_fun = drop_split_levels) %>%
    analyze("AGE", mean, format = "xx.x")

tbl1 <- build_table(lyt1, DM)
```

Individual Mod

```
> tbl1
              all obs
---------------------
A
  F
    mean       33.2
  M
    mean       34.5
B
  F
    mean       34.2
  M
    mean       34.0
C
  F
    mean       35.1
  M
    mean       34.5
```

Indent Mods

```
lyt2 <- basic_table() %>%
    split_rows_by("STRATA1", indent_mod = 1) %>%
    split_rows_by("SEX",
                  split_fun = drop_split_levels) %>%
    analyze("AGE", mean, format = "xx.x")

tbl2 <- build_table(lyt2, DM)
```

Total effective change	Individual Mod
+1 = **+1**	
+1 = 0 + (+1)	
+1 = 0 + (+1)	
+1 = 0 + (+1)	
+1 = 0 + (+1)	
+1 = **+1**	
+1 = 0 + (+1)	
+1 = 0 + (+1)	
+1 = 0 + (+1)	
+1 = 0 + (+1)	
+1 = **+1**	
+1 = 0 + (+1)	
+1 = 0 + (+1)	
+1 = 0 + (+1)	
+1 = 0 + (+1)	

```
> tbl2
                    all obs
___________________________
  A
    F
      mean           33.2
    M
      mean           34.5
  B
    F
      mean           34.2
    M
      mean           34.0
  C
    F
      mean           35.1
    M
      mean           34.5
```

Indent Mods

```r
lyt3 <- basic_table() %>%
    split_rows_by("STRATA1", indent_mod = 1) %>%
    split_rows_by("SEX", indent_mod = -1,
                  split_fun = drop_split_levels) %>%
    analyze("AGE", mean, format = "xx.x")

tbl3 <- build_table(lyt3, DM)
```

Total effective change	Individual Mod
+1 = +1	
0 = -1 + (+1)	
0 = 0 + (0)	
0 = -1 + (+1)	
0 = 0 + (0)	
+1 = +1	
0 = -1 + (+1)	
0 = 0 + (0)	
0 = -1 + (+1)	
0 = 0 + (0)	
+1 = +1	
0 = -1 + (+1)	
0 = 0 + (0)	
0 = -1 + (+1)	
0 = 0 + (0)	

```
> tbl3
                all obs
-----------------------
 A
 F
   mean          33.2
 M
   mean          34.5
 B
 F
   mean          34.2
 M
   mean          34.0
 C
 F
   mean          35.1
 M
   mean          34.5
```

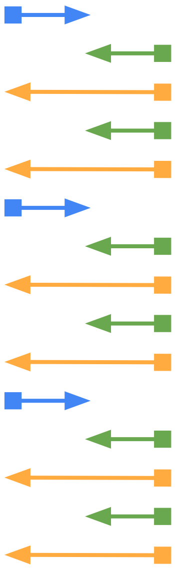
Indent Mods

```r
lyt4 <- basic_table() %>%
    split_rows_by("STRATA1", indent_mod = 1) %>%
    split_rows_by("SEX", indent_mod = -1,
                  split_fun = drop_split_levels) %>%
    analyze("AGE", mean, format = "xx.x",
            indent_mod = -2)

tbl4 <- build_table(lyt4, DM)
```

Total effective change

```
+1 = +1
 0 = -1 + (+1)
-2 = -2 + (0)
 0 = -1 + (+1)
-2 = -2 + (0)
+1 = +1
 0 = -1 + (+1)
-2 = -2 + (0)
 0 = -1 + (+1)
-2 = -2 + (0)
+1 = +1
 0 = -1 + (+1)
-2 = -2 + (0)
 0 = -1 + (+1)
-2 = -2 + (0)
```

Individual Mod



```
> tbl4
           all obs
  A
  F
mean        33.2
  M
mean        34.5
  B
  F
mean        34.2
  M
mean        34.0
  C
  F
mean        35.1
  M
mean        34.5
```

Indent Mods

- You can also apply indent mods to individual rows
    - `.indent_mods` argument in `in_rows()`
    - `indent_mod` argument in `rcell()`
        - First non-zero indent_mod promoted to apply to row,indent mods on other cells in the row are ignored
- I leave that as an exercise

Table Inset


TABLE CONTENT
LEFT EDGE OF
TITLE POSITION
NO TOUCHY
imgflip.com

Table Inset

```
lyt6 <- basic_table(title = "Main Title",
                    subtitles = "Subtitle",
                    main_footer = "Fooooooter",
                    prov_footer = "le provenance",
                    inset = 10) %>%
    split_cols_by("ARM") %>%
    split_rows_by("SEX",
                  split_fun = drop_split_levels) %>%
    analyze("AGE", mean, format = "xx.x")

tbl6 <- build_table(lyt6, DM)
```

Table Inset

```r
lyt6 <- basic_table(title = "Main Title",
                    subtitles = "Subtitle",
                    main_footer = "Fooooooter",
                    prov_footer = "le provenance",
                    inset = 10) %>%
    split_cols_by("ARM") %>%
    split_rows_by("SEX",
                  split_fun = drop_split_levels) %>%
    analyze("AGE", mean, format = "xx.x")

tbl6 <- build_table(lyt6, DM)
```

```
> tbl6
Main Title
Subtitle

          ————————————————————————————————————————————————————
                    A: Drug X    B: Placebo    C: Combination
          ————————————————————————————————————————————————————
          F
            mean      33.7          33.8            34.9
          M
            mean      36.5          32.1            34.3
          ————————————————————————————————————————————————————

          Fooooooter

le provenance
```

Table Inset

```r
lyt6 <- basic_table(title = "Main Title",
                    subtitles = "Subtitle",
                    main_footer = "Fooooooter",
                    prov_footer = "le provenance",
                    inset = 10) %>%
    split_cols_by("ARM") %>%
    split_rows_by("SEX",
                  split_fun = drop_split_levels) %>%
    analyze("AGE", mean, format = "xx.x")

tbl6 <- build_table(lyt6, DM)
```

Yes Touchy

```
> tbl6
Main Title
Subtitle

          ----------------------------------------------
                    A: Drug X   B: Placebo   C: Combination
          ----------------------------------------------
          F
            mean      33.7         33.8          34.9
          M
            mean      36.5         32.1          34.3
          ----------------------------------------------

          Fooooooter

le provenance
```

Word Wrapping and Width

Title/footer And Column Widths

- `max_width` - controls rendered width of title and footer information
  - Including referential footnotes
  - Currently need `tf_wrap = TRUE`
- `colwidths` (most funcs) - controls rendered width of columns
  - `widths (toString)` - same argument, difference in name an unfortunate artifact
  - Includes row labels + topleft as first "column"
  - Controls width of column labels
- Pagination machinery (Direct and via import) takes into account word-wrapping
  - ***Only*** if you specify the above to the paginator, table does not carry this info around

A Wide Table (From Our Tests)

```
> tt_for_wrap
Enough long title to be probably wider than expected

————————————————————————————————————————————————————————————————————————————————————————————————————————————————————————————————————————————————————————————
                              Incredibly long column name to be wrapped            This_should_be_somewhere_split                       C: Combination
————————————————————————————————————————————————————————————————————————————————————————————————————————————————————————————————————————————————————————————
ASIAN
  AGE
    Mean                                       32.50                                       36.68                                       36.99
  EOSDY
    Mean                   A very long content to_be_wrapped_and_splitted   A very long content to_be_wrapped_and_splitted   A very long content to_be_wrapped_and_splitted
BLACK OR AFRICAN AMERICAN
  AGE
    Mean                                       34.27                                       34.93                                       33.71
  EOSDY
    Mean                   A very long content to_be_wrapped_and_splitted   A very long content to_be_wrapped_and_splitted   A very long content to_be_wrapped_and_splitted
————————————————————————————————————————————————————————————————————————————————————————————————————————————————————————————————————————————————————————————

Also this seems quite wider than expected initially.
```

I Make It Fit And Then I Sits

```
> cat(export_as_txt(tt_for_wrap, tf_wrap = TRUE, max_width = 20,
                    colwidths = c(25, 20, 15, 20)))
Enough long title to
be probably wider
than expected

——————————————————————————————————————————————————————————————————————————————————————————
                              Incredibly long
                             column name to be    This_should_be_
                                  wrapped         somewhere_split      C: Combination
——————————————————————————————————————————————————————————————————————————————————————————
ASIAN
  AGE
    Mean                          32.50                36.68               36.99
  EOSDY
    Mean                     A very long content     A very long     A very long content
                            to_be_wrapped_and_sp  content to_be_w   to_be_wrapped_and_sp
                                   litted          rapped_and_spli          litted
                                                        tted
BLACK OR AFRICAN AMERICAN
  AGE
    Mean                          34.27                34.93               33.71
  EOSDY
    Mean                     A very long content     A very long     A very long content
                            to_be_wrapped_and_sp  content to_be_w   to_be_wrapped_and_sp
                                   litted          rapped_and_spli          litted
                                                        tted
——————————————————————————————————————————————————————————————————————————————————————————

Also this seems
quite wider than
expected initially.
```

Render the table

Multiple formats are supported

- ASCII Text - `export_as_txt`, `toString` (no pagination)
- PDF - `export_as_pdf`
- RTF - `export_as_rtf` (experimental) via `r2rtf`
- HTML - `as_html`
  - `No pagination`
- `flextable` object - `tt_to_flextable`
- PPT, DOCX - indirectly, via `officer` + `flextable`

And That's `rtables` in ~2.5 Hours