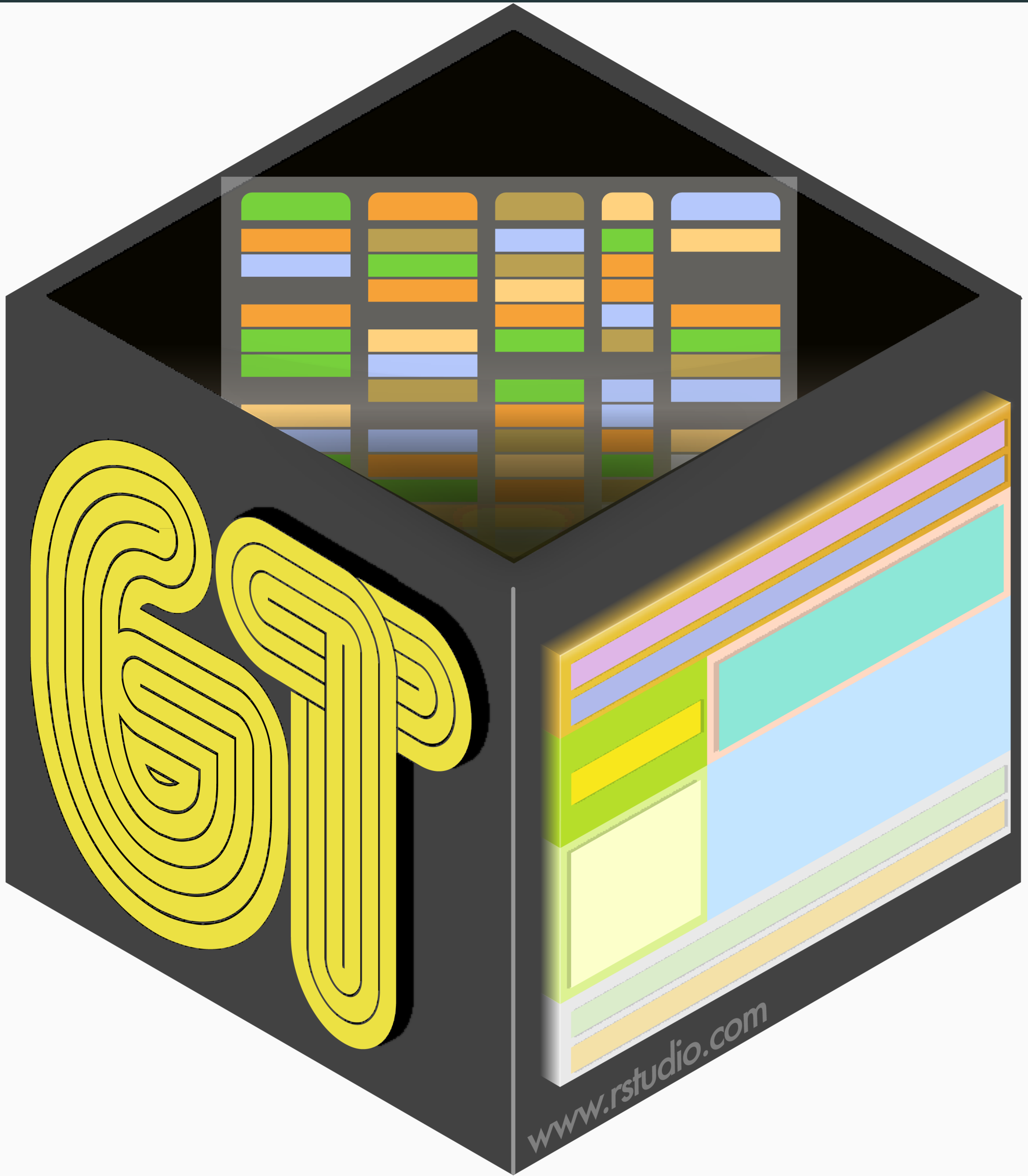
Introduction to {gt} + {gtsummary} Packages

Daniel D. Sjoberg

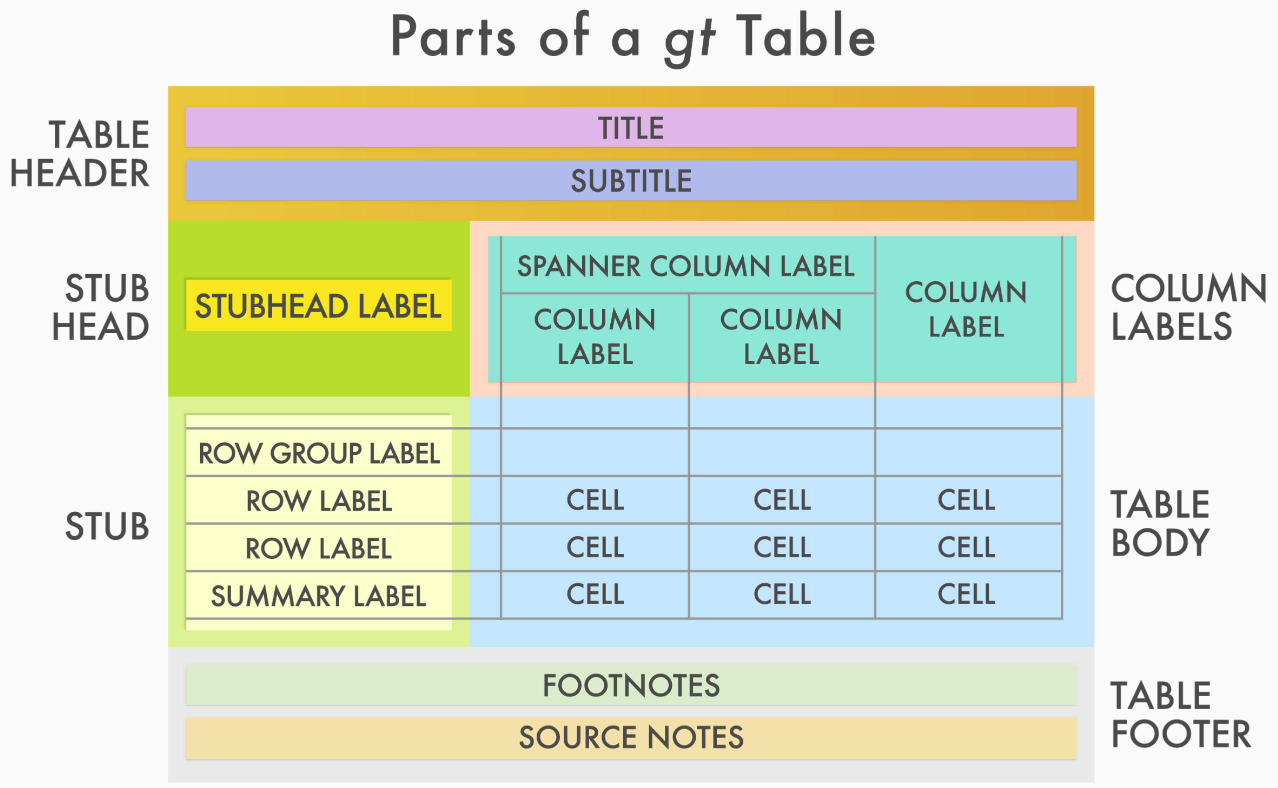
Memorial Sloan Kettering Cancer Center
June 27, 2019

{gt} package



{gt} philosophy

"We can construct a wide variety of useful tables with a cohesive set of table parts. These include the *table header*, the *stub*, the *stub head*, the *column labels*, the *table body*, and the *table footer*."



{gt} installation

- {gt} is not on CRAN.
- Use the code below to install from GitHub.

```r
remotes::install_github("rstudio/gt")
```

- While you're at it, install {gtsummary} as well.

```r
remotes::install_github("ddsjoberg/gtsummary")
```

- There is a version of {gtsummary} on CRAN, but with limited functionality.
- Use the version on GitHub (www.github.com/ddsjoberg/gtsummary).
- The full version of {gtsummary} be released on CRAN after {gt} is released.

{gt} examples: the data

When used alone, the `gt()` function prints a data frame. But so much more is possible!

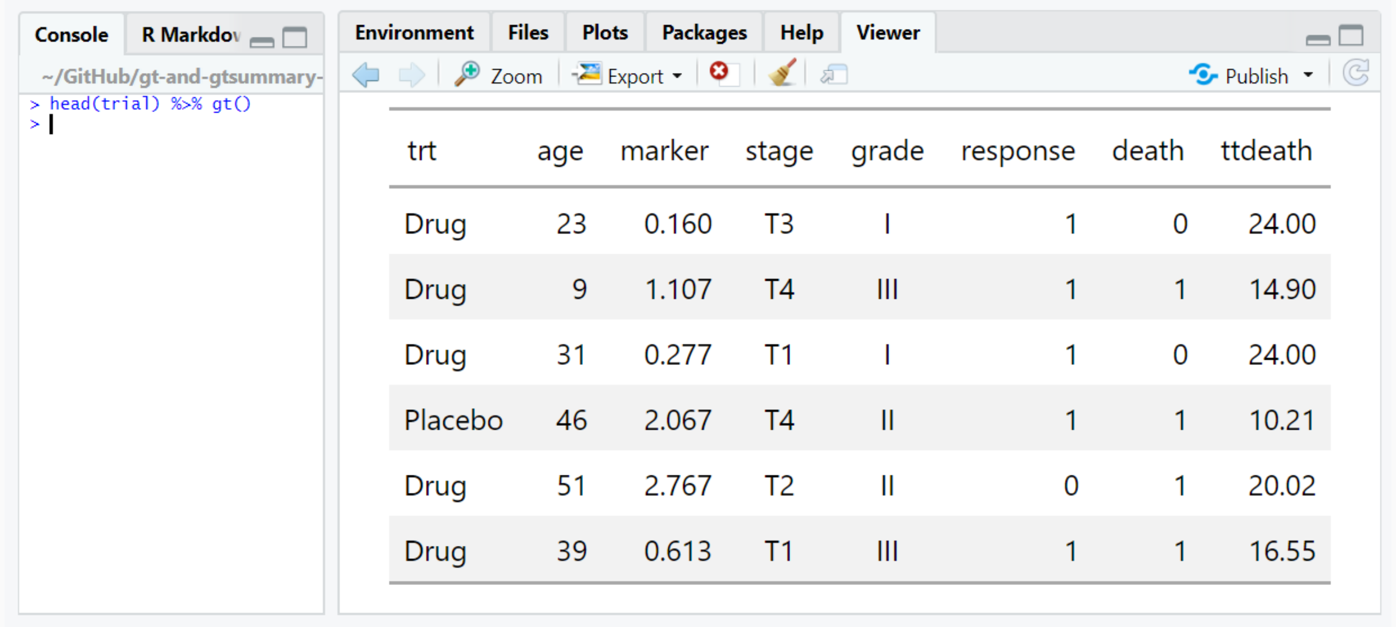
```r
library(gt)
# loading gtsummary for the data
library(gtsummary)
gt_trial_head <- head(trial) %>%
  gt()
```

trt	age	marker	stage	grade	response	death	ttdeath
Drug	23	0.160	T3	I	1	0	24.00
Drug	9	1.107	T4	III	1	1	14.90
Drug	31	0.277	T1	I	1	0	24.00
Placebo	46	2.067	T4	II	1	1	10.21
Drug	51	2.767	T2	II	0	1	20.02
Drug	39	0.613	T1	III	1	1	16.55

variable	class	label
trt	character	Treatment Randomization
age	numeric	Age, yrs
marker	numeric	Marker Level, ng/mL
stage	factor	T Stage
grade	factor	Grade
response	integer	Tumor Response
death	integer	Patient Died
ttdeath	numeric	Months to Death/Censor

{gt} examples: the viewer

- {gt} tables print to the RStudio viewer when in the global environment.



```
> head(trial) %>% gt()
>
```

trt	age	marker	stage	grade	response	death	ttdeath
Drug	23	0.160	T3	I	1	0	24.00
Drug	9	1.107	T4	III	1	1	14.90
Drug	31	0.277	T1	I	1	0	24.00
Placebo	46	2.067	T4	II	1	1	10.21
Drug	51	2.767	T2	II	0	1	20.02
Drug	39	0.613	T1	III	1	1	16.55

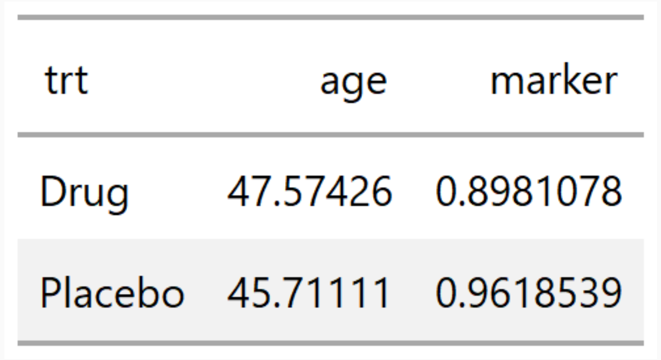
- {gt} tables also print in R markdown documents (HTML, PDF, RTF), Shiny apps, etc.

{gt} examples: formatting columns

```r
trial_summary ← trial %>% group_by(trt) %>% summarise_at(vars(age, marker), mean, na.rm = TRUE)
```

Raw Summary Statistics

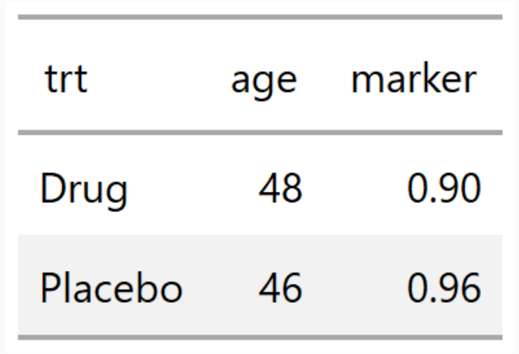
```r
gt_print ←
  gt(trial_summary)
```



trt	age	marker
Drug	47.57426	0.8981078
Placebo	45.71111	0.9618539

Formatted Summary Statistics

```r
gt_format ←
  gt(trial_summary) %>%
  fmt_number(columns = vars(age), decimals = 0) %>%
  fmt_number(columns = "marker", decimals = 2)
```

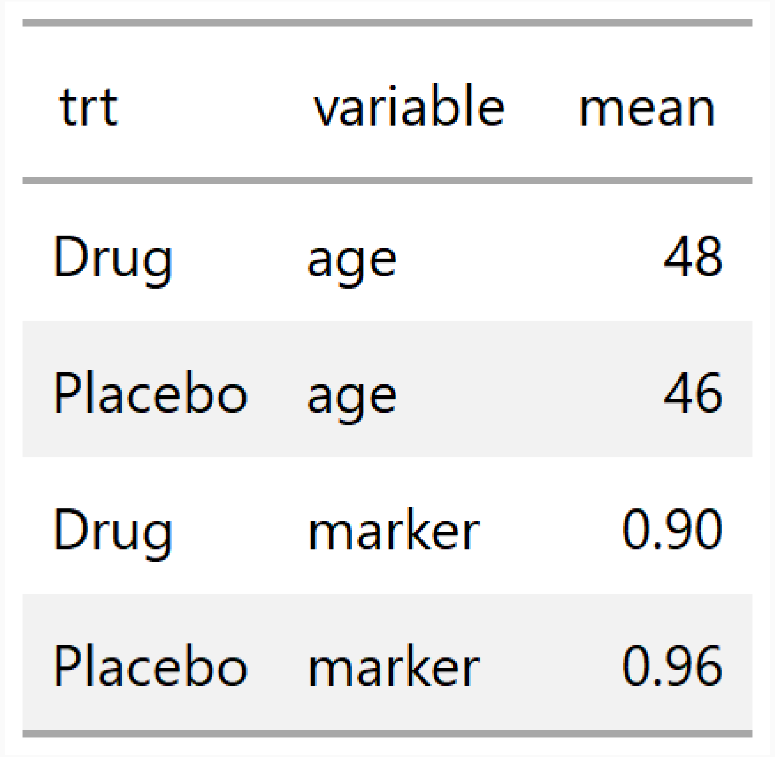


trt	age	marker
Drug	48	0.90
Placebo	46	0.96

Each column can be formatted without creating a character version of the column!

{gt} examples: formatting cells

```r
gt_fmt_cell <- trial_summary %>%
  gather("variable", "mean", -trt) %>%
  gt() %>%
  fmt_number(columns = vars(mean), rows = (variable == "age"), decimals = 0) %>%
  fmt_number(columns = vars(mean), rows = (variable == "marker"), decimals = 2)
```

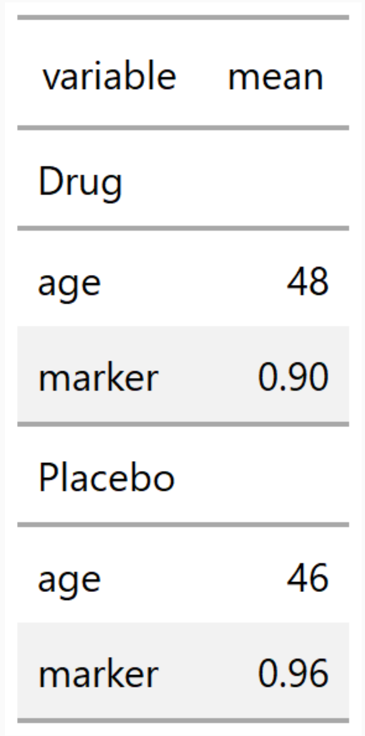


trt	variable	mean
Drug	age	48
Placebo	age	46
Drug	marker	0.90
Placebo	marker	0.96

- Use the `rows =` argument to pinpoint a cell to format.
- There are many formatting functions available: `fmt_percent()`, `fmt_currency()`, `fmt_date()`, `fmt_time()`, `fmt_missing()`, and more.
- You can write your own function and pass it to `fmt()` to format a table.

{gt} examples: grouping data

```r
gt_group ← trial_summary %>%
  gather("variable", "mean", -trt) %>%
  gt(groupname_col = "trt") %>%
  fmt_number(columns = vars(mean), rows = variable == "age", decimals = 0) %>%
  fmt_number(columns = vars(mean), rows = variable == "marker", decimals = 2)
```

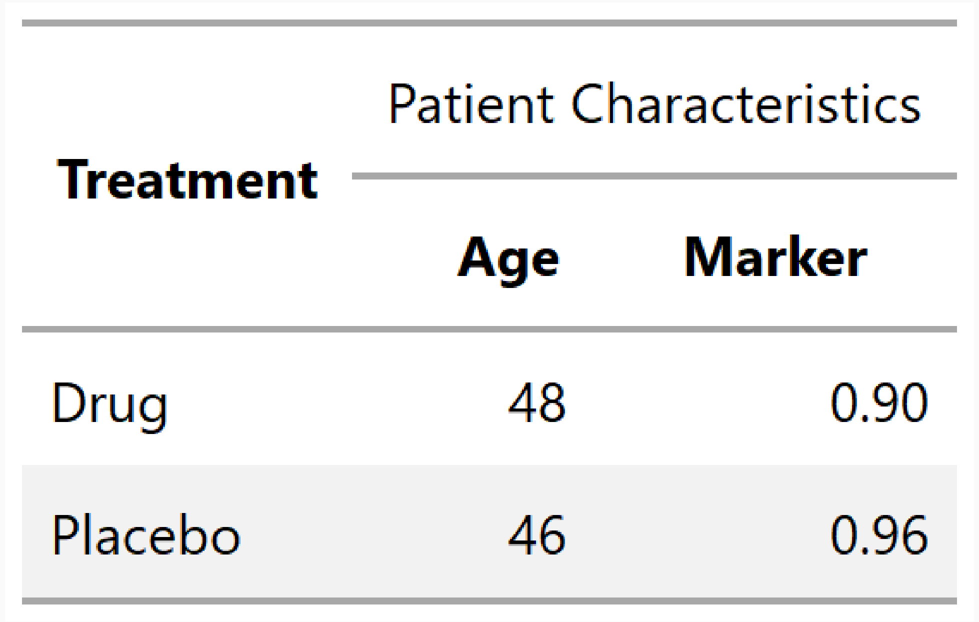


variable	mean
Drug	
age	48
marker	0.90
Placebo	
age	46
marker	0.96

- Use the `groupname_col =` argument to specify a column to group results.
- The grouping column is not printed and a stub row for each group is added.

{gt} examples: column formatting

```r
gt_cols ← trial_summary %>%
  gt() %>%
  fmt_number(columns = vars(age), decimals = 0) %>%
  fmt_number(columns = vars(marker), decimals = 2) %>%
  cols_label(trt = md("**Treatment**"), age = md("**Age**"), marker = md("**Marker**")) %>%
  tab_spanner(label = "Patient Characteristics", columns = vars(age, marker))
```



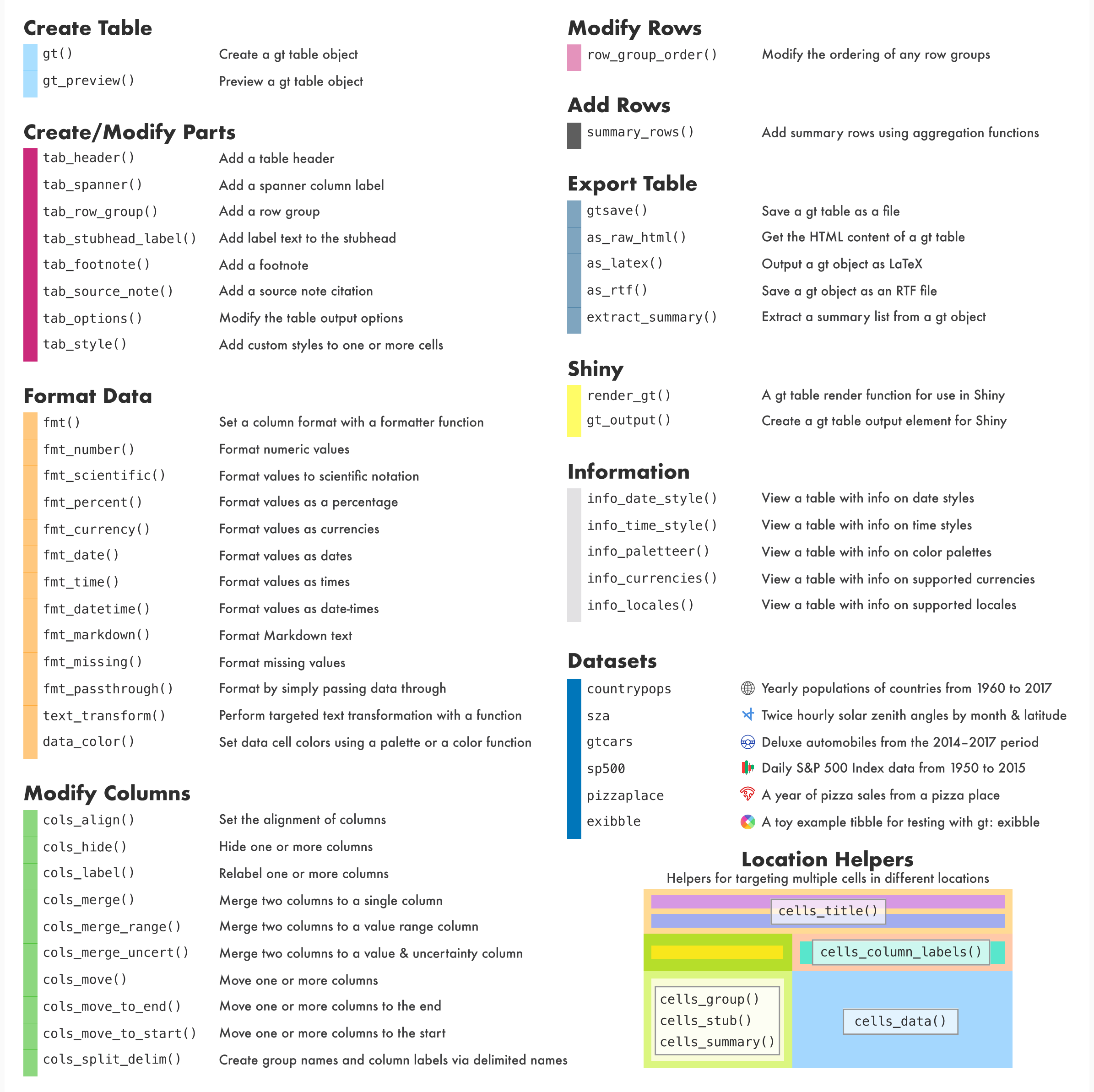
Treatment	Patient Characteristics	
	Age	**Marker**
Drug	48	0.90
Placebo	46	0.96

- The `cols_label()` function modifies the column headers.
- The `tab_spanner()` function includes a spanning header row.
- The `md()` function interprets input text as Markdown (see also `html()`).

{gt} examples: titles & footnotes

```r
gt_title_footnote <- trial_summary %>%
  gt() %>%
  fmt_number(columns = vars(age), decimals = 0) %>%
  fmt_number(columns = vars(marker), decimals = 2) %>%
  cols_label(trt = md("**Treatment**"), age = md("**Age**"), marker = md("**Marker**")) %>%
  tab_header(title = "Patient Characteristics", subtitle = "Presented by treatment") %>%
  tab_footnote(footnote = "Statistic presented is the mean.",
               locations = cells_column_labels(columns = vars(age, marker)))
```

much much more {gt} to learn



Create Table

gt()	Create a gt table object
gt_preview()	Preview a gt table object

Create/Modify Parts

tab_header()	Add a table header
tab_spanner()	Add a spanner column label
tab_row_group()	Add a row group
tab_stubhead_label()	Add label text to the stubhead
tab_footnote()	Add a footnote
tab_source_note()	Add a source note citation
tab_options()	Modify the table output options
tab_style()	Add custom styles to one or more cells

Format Data

fmt()	Set a column format with a formatter function
fmt_number()	Format numeric values
fmt_scientific()	Format values to scientific notation
fmt_percent()	Format values as a percentage
fmt_currency()	Format values as currencies
fmt_date()	Format values as dates
fmt_time()	Format values as times
fmt_datetime()	Format values as date-times
fmt_markdown()	Format Markdown text
fmt_missing()	Format missing values
fmt_passthrough()	Format by simply passing data through
text_transform()	Perform targeted text transformation with a function
data_color()	Set data cell colors using a palette or a color function

Modify Columns

cols_align()	Set the alignment of columns
cols_hide()	Hide one or more columns
cols_label()	Relabel one or more columns
cols_merge()	Merge two columns to a single column
cols_merge_range()	Merge two columns to a value range column
cols_merge_uncert()	Merge two columns to a value & uncertainty column
cols_move()	Move one or more columns
cols_move_to_end()	Move one or more columns to the end
cols_move_to_start()	Move one or more columns to the start
cols_split_delim()	Create group names and column labels via delimited names

Modify Rows

row_group_order()	Modify the ordering of any row groups

Add Rows

summary_rows()	Add summary rows using aggregation functions

Export Table

gtsave()	Save a gt table as a file
as_raw_html()	Get the HTML content of a gt table
as_latex()	Output a gt object as LaTeX
as_rtf()	Save a gt object as an RTF file
extract_summary()	Extract a summary list from a gt object

Shiny

render_gt()	A gt table render function for use in Shiny
gt_output()	Create a gt table output element for Shiny

Information

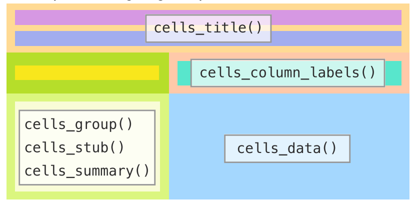
info_date_style()	View a table with info on date styles
info_time_style()	View a table with info on time styles
info_paletteer()	View a table with info on color palettes
info_currencies()	View a table with info on supported currencies
info_locales()	View a table with info on supported locales

Datasets

countrypops	Yearly populations of countries from 1960 to 2017
sza	Twice hourly solar zenith angles by month & latitude
gtcars	Deluxe automobiles from the 2014-2017 period
sp500	Daily S&P 500 Index data from 1950 to 2015
pizzaplace	A year of pizza sales from a pizza place
exibble	A toy example tibble for testing with gt: exibble

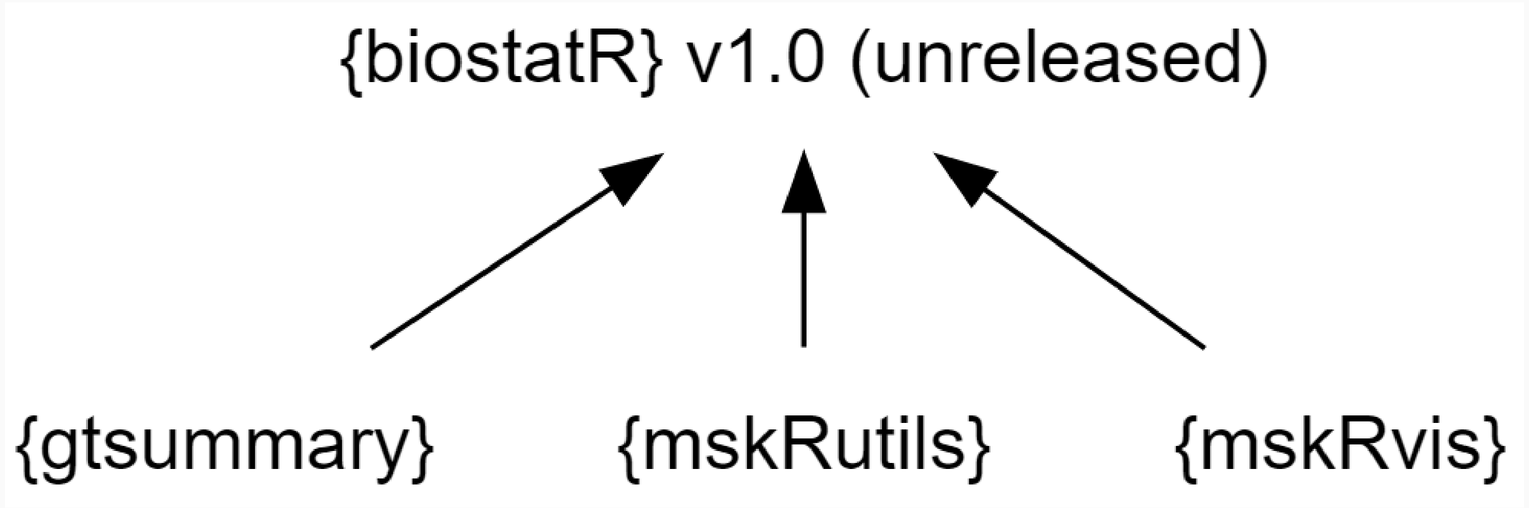
Location Helpers

Helpers for targeting multiple cells in different locations



{gtsummary}

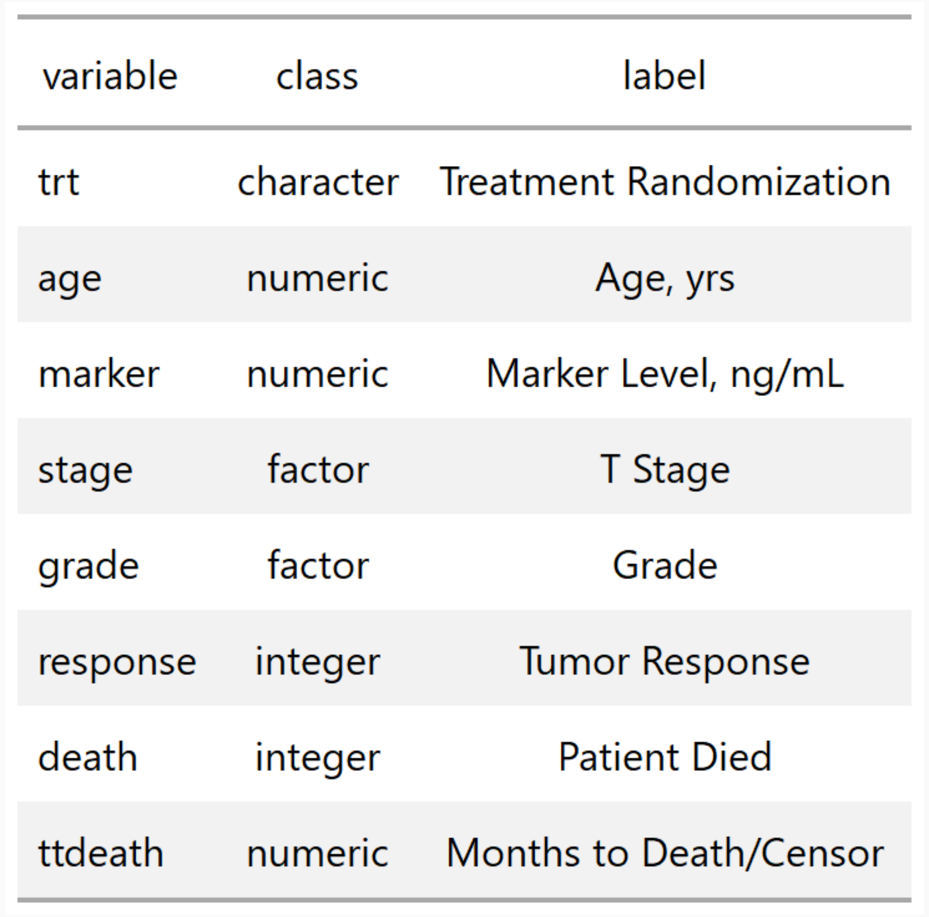
{gtsummary} introduction



- {gtsummary} will soon be a part of the biostatR-verse of packages.
- The package uses {gt} as its back end to create tables.
- Used to summarize data frames, regression models, and more.
- Has a tidy API, sensible defaults (meaning minimal code), and is highly customizable.

{gtsummary} summarize data with tbl_summary()

Let's review the data once more



variable	class	label
trt	character	Treatment Randomization
age	numeric	Age, yrs
marker	numeric	Marker Level, ng/mL
stage	factor	T Stage
grade	factor	Grade
response	integer	Tumor Response
death	integer	Patient Died
ttdeath	numeric	Months to Death/Censor

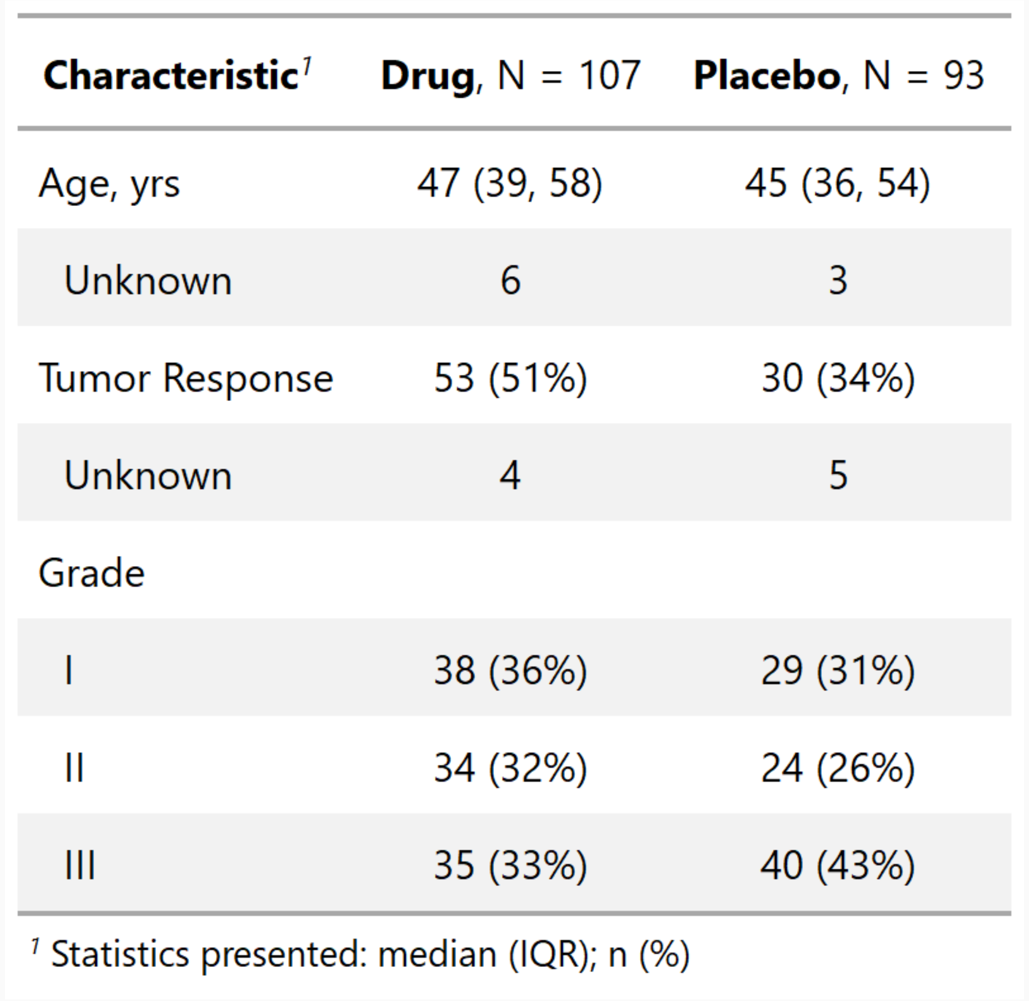
For brevity, we'll use an abbreviated version of the trial data set with fewer columns.

```
sm_trial ←
  trial %>%
  select(trt, age, response, grade)
```

{gtsummary} summarize data with tbl_summary()

```r
tbl_summary_1 ←
  tbl_summary(sm_trial, by = "trt")
```

- Default statistics are median (IQR) for continuous variables, and n (percent) for categorical data.
- By default, variables coded as 0/1, TRUE/FALSE, and Yes/No are presented dichotomously.



Characteristic[1]	**Drug**, N = 107	**Placebo**, N = 93
Age, yrs	47 (39, 58)	45 (36, 54)
Unknown	6	3
Tumor Response	53 (51%)	30 (34%)
Unknown	4	5
Grade		
I	38 (36%)	29 (31%)
II	34 (32%)	24 (26%)
III	35 (33%)	40 (43%)

[1] Statistics presented: median (IQR); n (%)

{gtsummary} summarize data with tbl_summary()

```r
tbl_summary_2 ←
  tbl_summary(sm_trial, by = "trt") %>%
  add_p()
```

- To compare values across two or more groups, use the `add_p()` function.
- The default tests are the Wilcoxon rank-sum test for continuous variables, chi-square test of independence for most categorical data, and Fisher's exact test for categorical data with low expected counts.

Characteristic[1]	**Drug**, N = 107	**Placebo**, N = 93	**p-value**[2]
Age, yrs	47 (39, 58)	45 (36, 54)	0.3
Unknown	6	3	
Tumor Response	53 (51%)	30 (34%)	0.023
Unknown	4	5	
Grade			0.3
I	38 (36%)	29 (31%)	
II	34 (32%)	24 (26%)	
III	35 (33%)	40 (43%)	

[1] Statistics presented: median (IQR); n (%)
[2] Statistical tests performed: Wilcoxon rank-sum test; chi-square test of independence

{gtsummary} and the {glue} package: an aside

- {glue} is similar to paste (but I like it so much more).
- Embed R expressions in curly braces.
- They are then evaluated and inserted into the argument string.

```
name = "Daniel"
x = 1
glue::glue("{name} is number {x}")
```

```
## Daniel is number 1
```

- Expression can be complex.

```
glue::glue("{name} is number {((x + 100) * 10) - 1009}")
```

```
## Daniel is number 1
```

{gtsummary} summarize data with tbl_summary()

```r
tbl_summary_3 ← sm_trial %>%
  tbl_summary(
    by = "trt",
    statistic = list(
      all_continuous() ~ "{mean} ({sd})",
      all_categorical() ~ "{n} / {N} ({p}%)"
    ),
    label = "age" ~ "Patient Age"
  ) %>%
  add_p(test = all_continuous() ~ "t.test")
```

- Report mean and standard deviation for continuous variables.
- Specify label for age variable.
- Report p-values from the t-test.

Characteristic[1]	**Drug**, N = 107	**Placebo**, N = 93	**p-value**[2]
Patient Age	48 (15)	46 (13)	0.4
Unknown	6	3	
Tumor Response	53 / 103 (51%)	30 / 88 (34%)	0.023
Unknown	4	5	
Grade			0.3
I	38 / 107 (36%)	29 / 93 (31%)	
II	34 / 107 (32%)	24 / 93 (26%)	
III	35 / 107 (33%)	40 / 93 (43%)	

[1] Statistics presented: mean (SD); n / N (%)
[2] Statistical tests performed: t-test; chi-square test of independence

{gtsummary} summarize data with tbl_summary()

```r
tbl_summary_4 <- sm_trial %>%
  tbl_summary(
    by = "trt",
    type = "response" ~ "categorical",
    statistic = all_continuous() ~ "{mean} ({sd})",
    digits = vars(age) ~ c(0, 1)
  ) %>%
  add_p(test = all_continuous() ~ "t.test") %>%
  add_stat_label()
```

- Report levels for the response variable.
- Modify the default rounding for age.
- Add column of statistics presented.
- Footnote about statistics is gone!

Characteristic	**Statistic**	**Drug**, N = 107	**Placebo**, N = 93	**p-value**[1]
Age, yrs	mean (SD)	48 (15.4)	46 (13.2)	0.4
Unknown	n	6	3	
Tumor Response				0.023
0	n (%)	50 (49%)	58 (66%)	
1	n (%)	53 (51%)	30 (34%)	
Unknown	n	4	5	
Grade				0.3
I	n (%)	38 (36%)	29 (31%)	
II	n (%)	34 (32%)	24 (26%)	
III	n (%)	35 (33%)	40 (43%)	

[1] Statistical tests performed: t-test; chi-square test of independence

{gtsummary} summarize data with tbl_summary()

Advanced Customization

- It's natural a {gtsummary} package user would want to customize the aesthetics of the table with one or more of the many {gt} functions available.

- Every function in {gt} is available to use with a {gtsummary} object.

1. Create a {gtsummary} table.

2. Convert the table to a {gt} object with the `as_gt()` function.

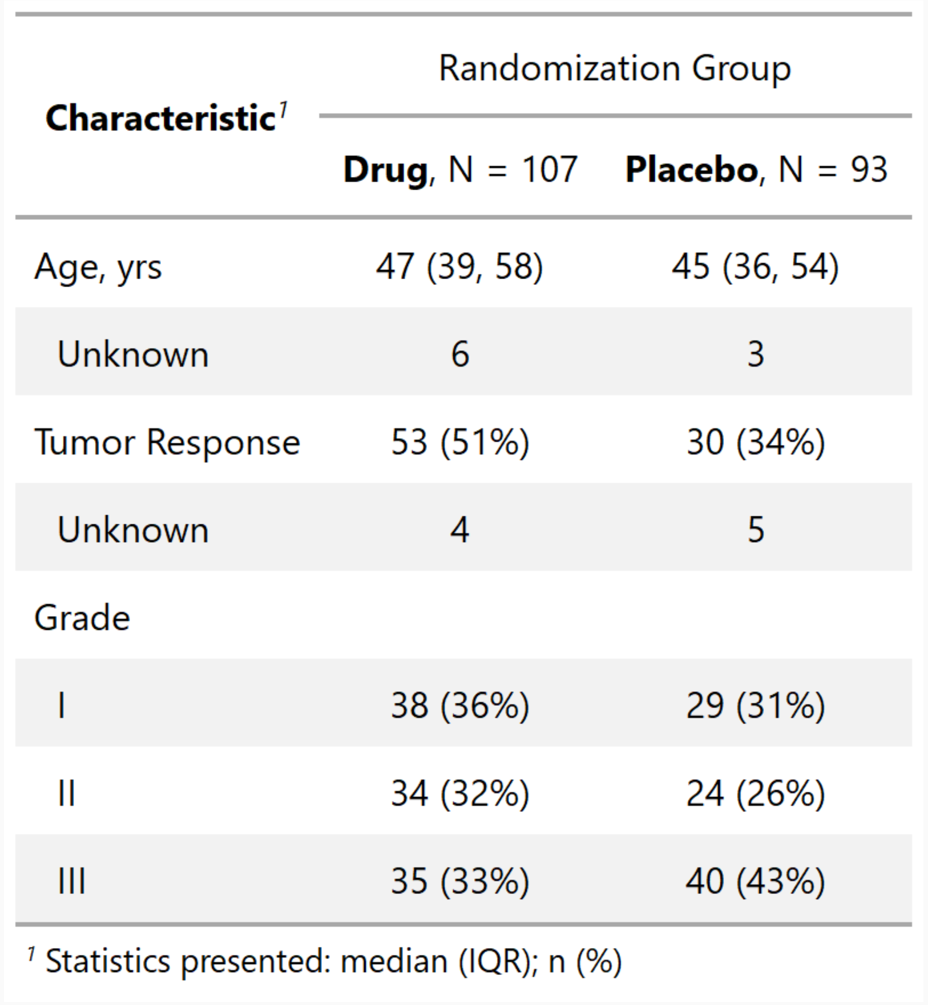
3. Continue formatting as a {gt} table with any {gt} function.

{gtsummary} summarize data with tbl_summary()

Advanced Customization

```r
tbl_summary_5 ← sm_trial %>%
  tbl_summary(by = "trt") %>%
  # convert from gtsummary object to gt object
  as_gt() %>%
  # modify with gt functions
  tab_spanner(
    label = "Randomization Group",
    columns = starts_with("stat_")
  )
```

More on this in the `tbl_summary()` vignette



Characteristic[1]	Randomization Group	
	Drug, N = 107	**Placebo**, N = 93
Age, yrs	47 (39, 58)	45 (36, 54)
Unknown	6	3
Tumor Response	53 (51%)	30 (34%)
Unknown	4	5
Grade		
I	38 (36%)	29 (31%)
II	34 (32%)	24 (26%)
III	35 (33%)	40 (43%)

[1] Statistics presented: median (IQR); n (%)

{gtsummary} summarize data with tbl_summary()

Review the tbl_summary vignette for more details
http://www.danieldsjoberg.com/gtsummary/articles/tbl_summary.html

- Reporting any statistic for continuous variables, including user-written functions.
- More on dichotomous variables and how to specify the level printed.
- Missing data options (e.g. report as a column rather than a row, always report N missing even when no missing data, modify missing text, etc.).
- Sort categorical variables by frequency.
- Report row percent, rather than column percent.
- Report q-values from various methods like false discovery rate.
- Sort data by ascending p-values when comparisons have been made.

{gtsummary} summarize models with tbl_regression()

Raw Output

```
m1 ← glm(response ~ trt + grade + age, data = trial, family = binomial)
m1
```

```
##
## Call:  glm(formula = response ~ trt + grade + age, family = binomial,
##     data = trial)
##
## Coefficients:
## (Intercept)   trtPlacebo      gradeII     gradeIII          age
##    0.449477    -0.660514    -0.504973    -0.167732    -0.004199
##
## Degrees of Freedom: 181 Total (i.e. Null);  177 Residual
##   (18 observations deleted due to missingness)
## Null Deviance:       249.1
## Residual Deviance: 242.8     AIC: 252.8
```

{gtsummary} summarize models with tbl_regression()

{broom} Output

```r
broom::tidy(m1, conf.int = TRUE, exponentiate = TRUE)
```

```
## # A tibble: 5 x 7
##   term        estimate std.error statistic p.value conf.low conf.high
##   <chr>          <dbl>     <dbl>     <dbl>   <dbl>    <dbl>     <dbl>
## 1 (Intercept)    1.57     0.565      0.796  0.426     0.520     4.81
## 2 trtPlacebo     0.517    0.309     -2.14   0.0324    0.280     0.941
## 3 gradeII        0.604    0.390     -1.30   0.195     0.279     1.29
## 4 gradeIII       0.846    0.369     -0.455  0.649     0.409     1.74
## 5 age            0.996    0.0106    -0.395  0.693     0.975     1.02
```

{gtsummary} summarize models with tbl_regression()

{gtsummary} Output

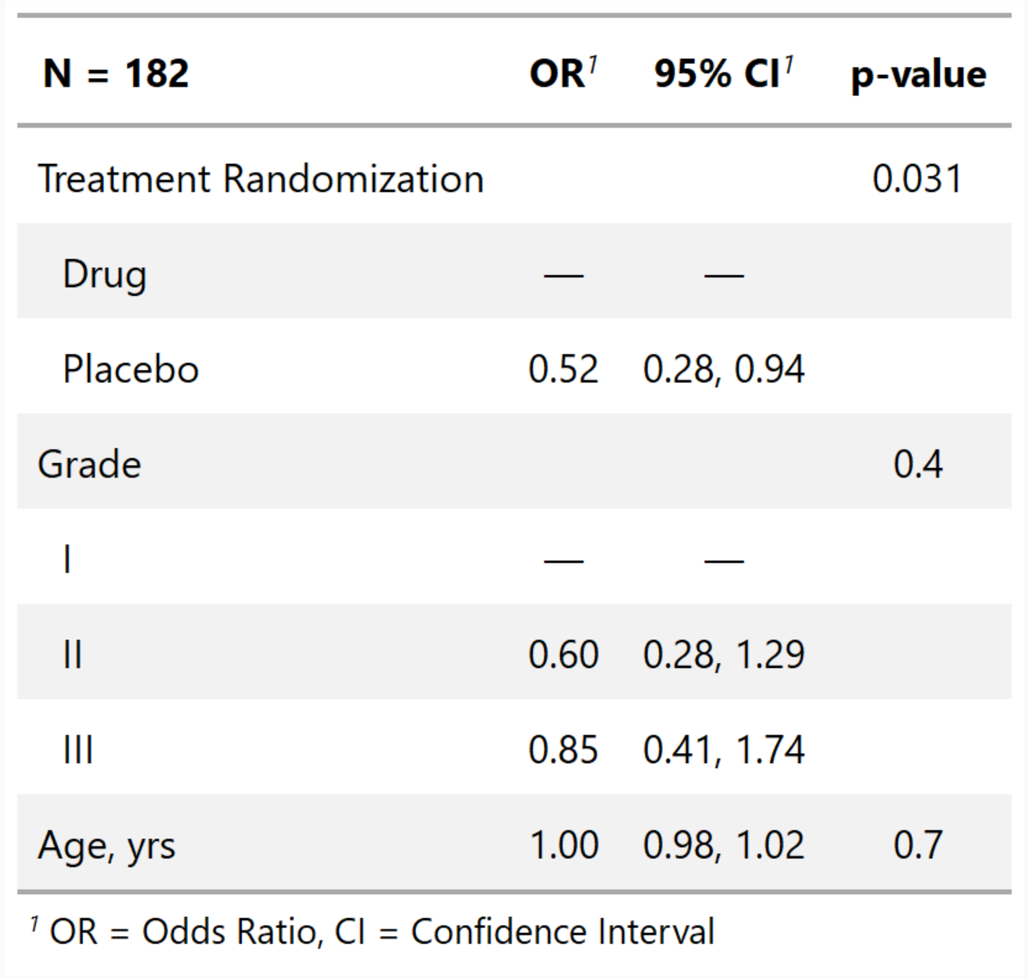
```
tbl_regression_1 ← tbl_regression(m1, exponentiate = TRUE)
```

- `tbl_regression()` accepts regression model object as inputs.
- Reference groups added to the table.
- Logistic regression model with odds ratio header and footnote.

{gtsummary} summarize models with tbl_regression()

```
tbl_regression_2 ← m1 %>%
  tbl_regression(exponentiate = TRUE) %>%
  add_global_p()
```

- Replace individual p-values for categorical variables with global p-value for the entire variable.



N = 182	OR[1]	95% CI[1]	p-value
Treatment Randomization			0.031
Drug	—	—	
Placebo	0.52	0.28, 0.94	
Grade			0.4
I	—	—	
II	0.60	0.28, 1.29	
III	0.85	0.41, 1.74	
Age, yrs	1.00	0.98, 1.02	0.7

[1] OR = Odds Ratio, CI = Confidence Interval

{gtsummary} summarize models with tbl_regression()

```r
library(survival)
tbl_regression_3 ←
  coxph(Surv(ttdeath, death) ~ trt + grade + age,
        data = trial) %>%
  tbl_regression(exponentiate = TRUE)
tbl_regression_4 ←
  tbl_merge(
    tbls = list(tbl_regression_1, tbl_regression_3),
    tab_spanner = c("Tumor Response", "Time to Death")
  )
```

- Build Cox regression model with same predictors as previous model.
- Merge the two regression models with the same predictors and present results side-by-side.

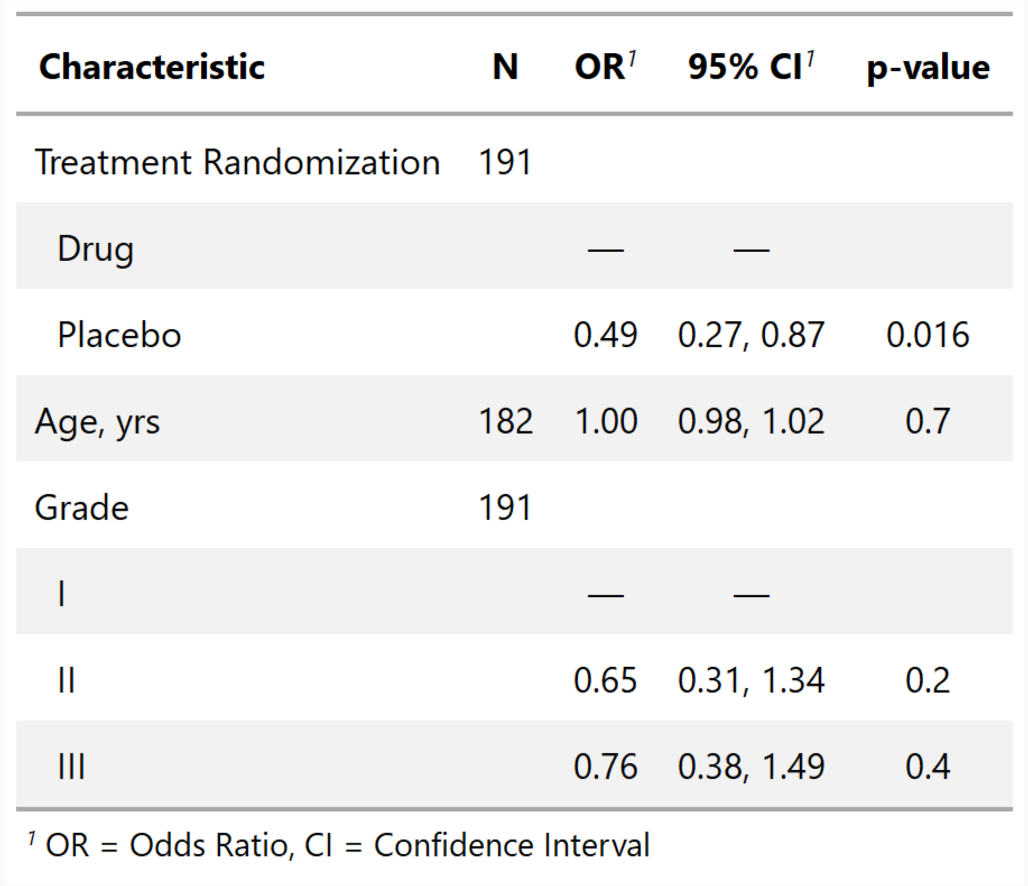
Characteristic	Tumor Response			Time to Death		
	OR[1]	**95% CI**[1]	**p-value**	**HR**[1]	**95% CI**[1]	**p-value**
Treatment Randomization						
Drug	—	—		—	—	
Placebo	0.52	0.28, 0.94	0.032	1.31	0.89, 1.94	0.2
Grade						
I	—	—		—	—	
II	0.60	0.28, 1.29	0.2	1.25	0.74, 2.12	0.4
III	0.85	0.41, 1.74	0.6	1.87	1.16, 3.01	0.011
Age, yrs	1.00	0.98, 1.02	0.7	1.01	0.99, 1.02	0.4

[1] OR = Odds Ratio, HR = Hazard Ratio, CI = Confidence Interval

{gtsummary} summarize data with tbl_uvregression()

```r
library(survival)
tbl_uvregression_1 <-
  tbl_uvregression(
    sm_trial,
    method = glm,
    y = response,
    method.args = list(family = binomial),
    exponentiate = TRUE
  )
```

- Table of univariate regression models.
- Specify the outcome, and the remaining variables in data frame serve as predictors.

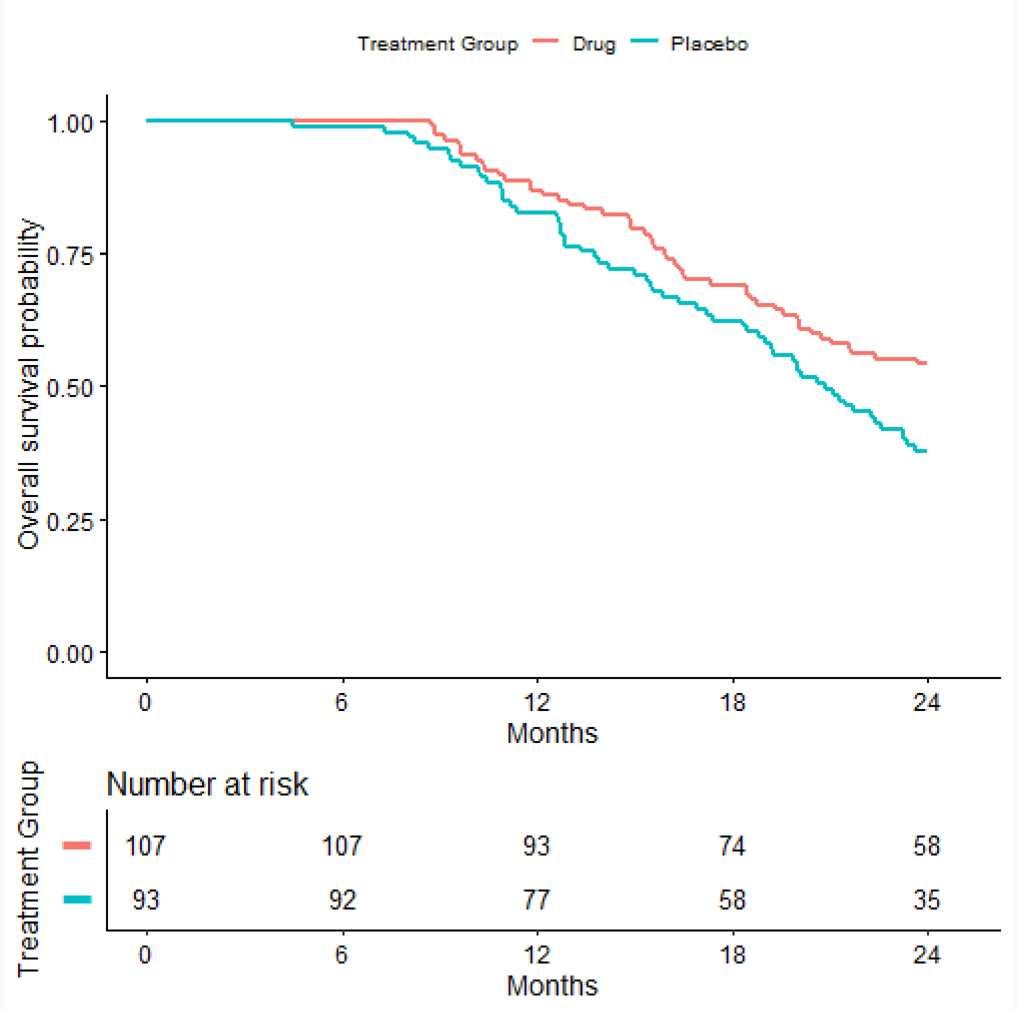


Characteristic	N	OR[1]	95% CI[1]	p-value
Treatment Randomization	191			
Drug		—	—	
Placebo		0.49	0.27, 0.87	0.016
Age, yrs	182	1.00	0.98, 1.02	0.7
Grade	191			
I		—	—	
II		0.65	0.31, 1.34	0.2
III		0.76	0.38, 1.49	0.4

[1] OR = Odds Ratio, CI = Confidence Interval

{gtsummary} summarize data with tbl_survival()

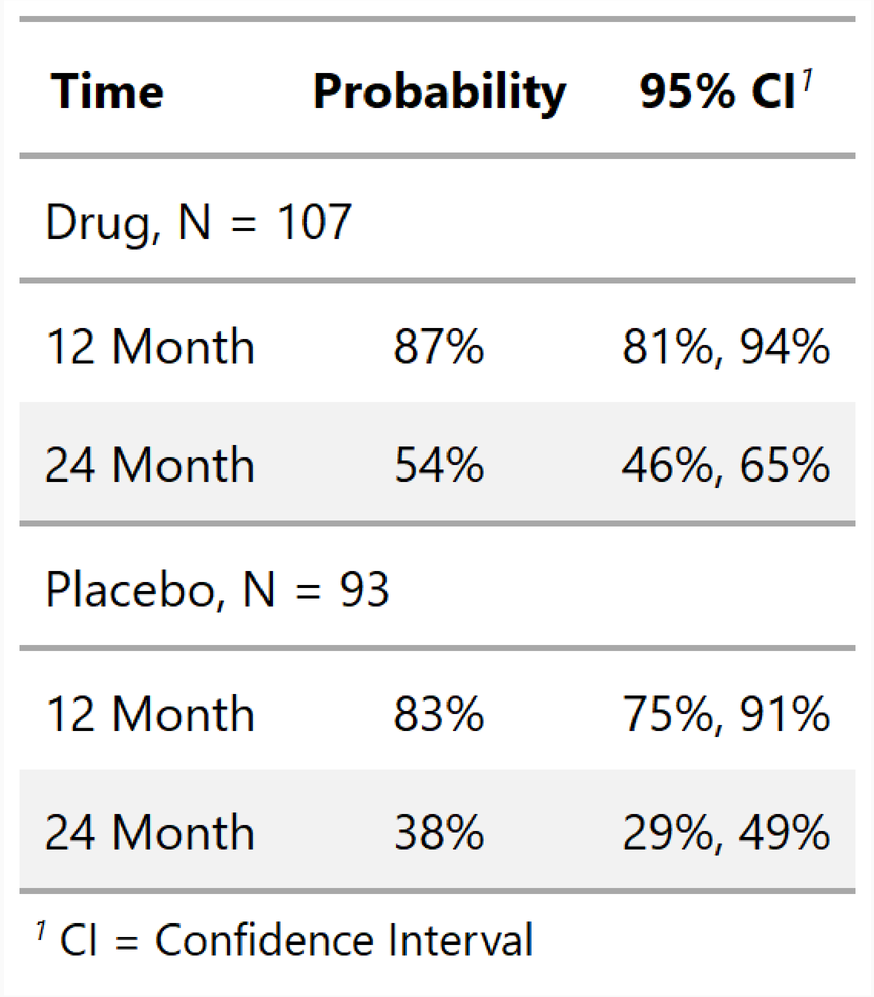
```r
fit1 ← survfit(Surv(ttdeath, death) ~ trt,
               data = trial)

survminer::ggsurvplot(
  fit = fit1,
  xlab = "Months",
  ylab = "Overall survival probability",
  legend.title = "Treatment Group",
  legend.labs = c("Drug", "Placebo"),
  break.x.by = 6,
  censor = FALSE,
  risk.table = TRUE,
  risk.table.y.text = FALSE
)
```



{gtsummary} summarize data with tbl_survival()

```r
tbl_survival_1 <- fit1 %>%
  tbl_survival(times = c(12, 24),
               label = "{time} Month")
```

- First, use `survfit()` to estimate survival times.
- Create table of estimates with `tbl_survival()`.
- Can use this function to print survival quantiles as well, e.g. median survival.



Time	Probability	95% CI[1]
Drug, N = 107		
12 Month	87%	81%, 94%
24 Month	54%	46%, 65%
Placebo, N = 93		
12 Month	83%	75%, 91%
24 Month	38%	29%, 49%

[1] CI = Confidence Interval

{gtsummary} reporting results with inline_text()

- Tables are important, but we often need to report results in-line in a report.
- Any statistic reported in a {gtsummary} table can be extracted and reported in-line in a R Markdown document with the `inline_text()` function.

```
inline_text(tbl_regression_1, variable = "trt", level = "Placebo")
```

```
0.52 (95% CI 0.28, 0.94; p=0.032)
```

- The pattern of what is reported can be modified with the `pattern =` argument.
- Default is `pattern = "{estimate} ({conf.level*100}% CI {conf.low}, {conf.high}; {p.value})"`.

{gtsummary}

· Every function is documented further in the help file ·

· Check out the package website for vignettes including detailed examples and explanations ·

{gtsummary} documentation danieldsjoberg.com/gtsummary/

{gtsummary} package github.com/ddsjoberg/gtsummary

slides at danieldsjoberg.com/gt-and-gtsummary-presentation

source code for slides at github.com/ddsjoberg/gt-and-gtsummary-presentation

{gt} package github.com/rstudio/gt

{gtsummary} Advanced

{gtsummary} output is a list that prints as a {gt} table.

```r
names(tbl_summary_1)
```

```
## [1] "gt_calls"   "table_body" "meta_data"  "inputs"     "call_list"
## [6] "by"         "df_by"
```

```r
pluck(tbl_summary_1, "table_body") %>% head()
```

```
## # A tibble: 6 x 5
##   variable row_type label          stat_1      stat_2
##   <chr>    <chr>    <chr>          <chr>       <chr>
## 1 age      label    Age, yrs       47 (39, 58) 45 (36, 54)
## 2 age      missing  Unknown        6           3
## 3 response label    Tumor Response 53 (51%)    30 (34%)
## 4 response missing  Unknown        4           5
## 5 grade    label    Grade          <NA>        <NA>
## 6 grade    level    I              38 (36%)    29 (31%)
```

```r
pluck(tbl_summary_1, "gt_calls") %>% head(n = 4)
```

```
## $gt
## gt(data = x$table_body)
##
## $cols_label_label
## cols_label(label = md('**Characteristic**'))
##
## $cols_align
## cols_align(align = 'center') %>% cols_align(align = 'lef
##
## $cols_hide
## cols_hide(columns = vars(variable, row_type))
```