gtsummary

Creating publication-ready analytical tables

Margie Hannum

Memorial Sloan Kettering Cancer Center

February 26, 2020

{gtsummary} package

{gtsummary} Overview

- Package will create your tabular summaries, with sensible defaults that are highly customizable
  - Summarize data frames/tibbles
  - Summarize regression models
  - Customize tables
  - Report statistics from {gtsummary} tables inline in R Markdown

{gtsummary} v1.2.5 - Package Website

- Package Website: http://www.danieldsjoberg.com/gtsummary/
  - 🕮 Installation instructions
  - 🕮 Thorough documentation on every function
  - 🕮 Detailed tutorials
- Install {gtsummary} with the following code:

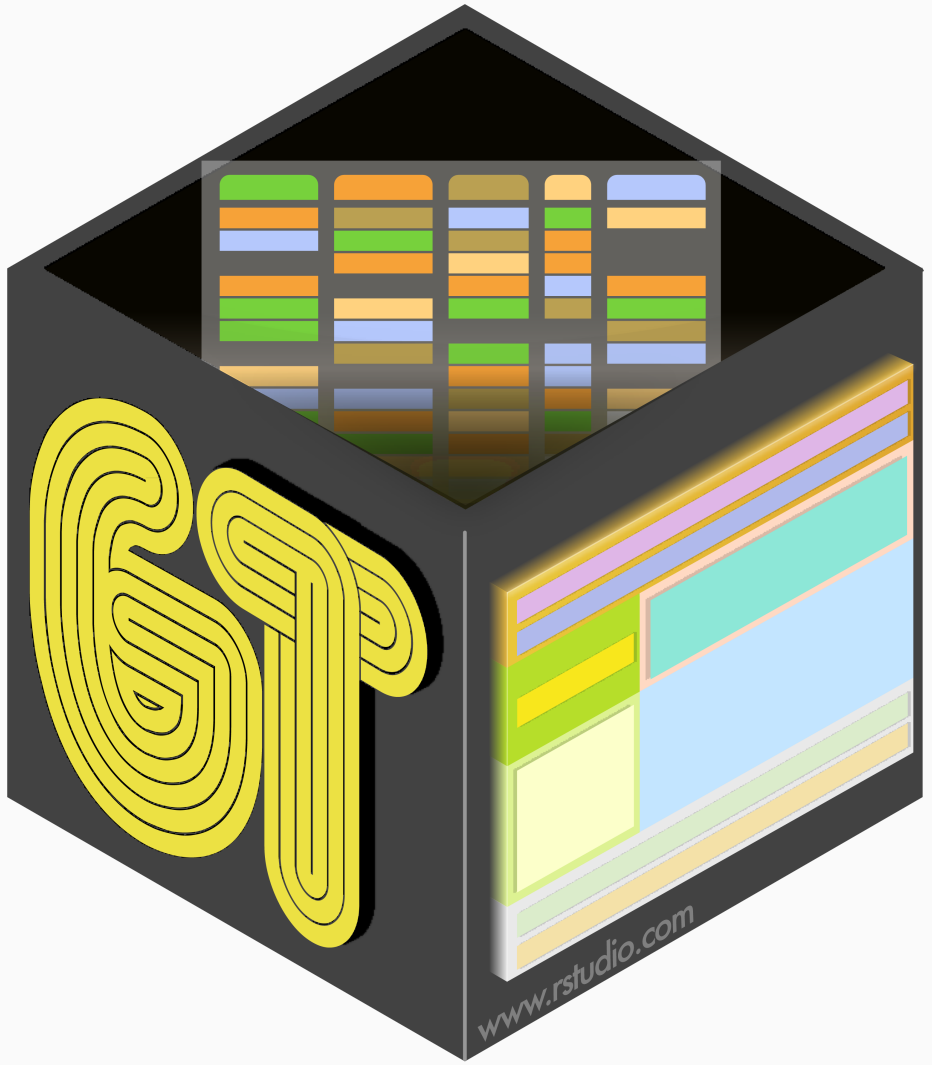
```
install.packages("gtsummary")
```

- Also recommended to install the development version of {gt} from GitHub.

```
install.packages("remotes")
remotes::install_github("rstudio/gt")
```

{gt} - an aside

- New package from RStudio
- Package for printing highly customized tables
- Goal is to unify code for creating tables in HTML, Word (via RTF), and PDF
- Check it out! https://gt.rstudio.com/



{gt} - an aside

- "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*."
- Workflow: input dataframe or tibble, create gt object (list with data and formatting elements), output gt table as HTML (previewed in the Viewer).

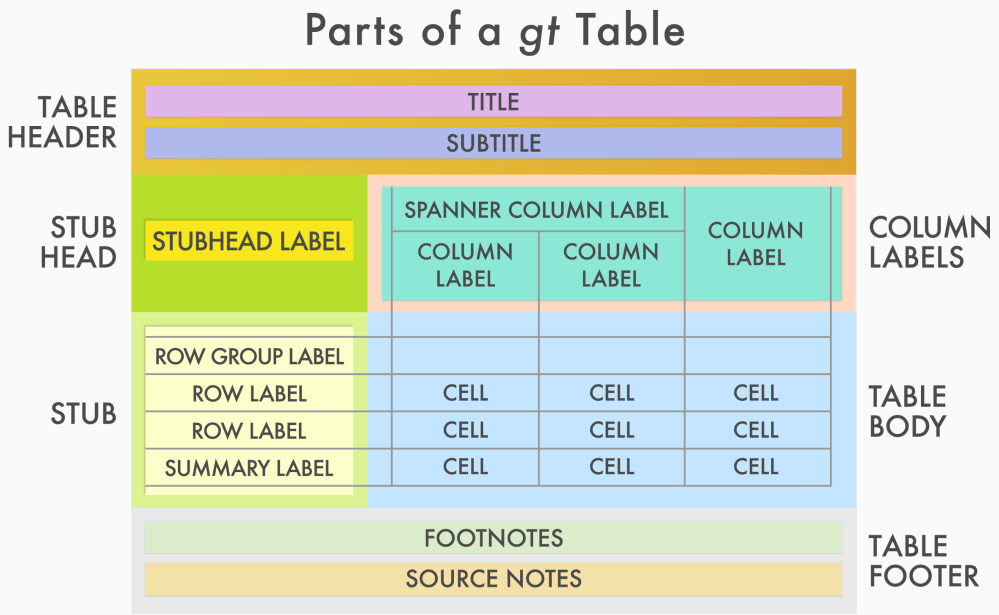


Image source: https://gt.rstudio.com/

{gtsummary} Print Engines - gt or kable?

gt

♥ Highly Customizable

♥ All output includes informative footnotes

♥ HTML Output

♥ PDF Output

⚠ MS Word Output via RTF still in dev,
requires re-sizing of tables by hand

kable

⚠ Less Customizable

⚠ Footnotes and spanning headers stripped from all output

♥ HTML Output

♥ PDF Output

♥ MS Word Output

We built {gtsummary} as a companion to {gt} and highly recommend it!

All examples shown use {gt} print engine

trial dataset overview

```
head(trial, 3)
```

```
## # A tibble: 3 x 8
##   trt      age marker stage grade response death ttdeath
##   <chr>  <dbl>  <dbl> <fct> <fct>    <int> <int>   <dbl>
## 1 Drug A    23  0.16  T1    II           0     0      24
## 2 Drug B     9  1.11  T2    I            1     0      24
## 3 Drug A    31  0.277 T1    II           0     0      24
```

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

- Throughout this presentation examples will use the `trial` dataset, included with {gtsummary}.
- Dataset contains baseline characterstics of 200 patients who received Drug A or Drug B. Includes outcome of tumor response to the treatment.
- Variables have label attributes assigned using the `labelled` package.
- For simplicity in this presentation, subset data to a few variables of interest:

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

tbl_summary()

{gtsummary} summarize data with tbl_summary()

Example: Summarizing clinical trial data

Produce a table of descriptive statistics using one line of code:

```
tbl_summary(sm_trial)
```

Notice some nice default behaviors:

- Detects variable types of input data and calculates descriptive statistics
  - 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.
- Recognizes `NA` values as "missing" and lists them as unknown
- Label attributes automatically printed
- Variable levels indented and footnotes added *({gt})*

Characteristic	**N = 200**[1]
Chemotherapy Treatment	
Drug A	98 (49%)
Drug B	102 (51%)
Age, yrs	47 (38, 57)
Unknown	11
Tumor Response	61 (32%)
Unknown	7
Grade	
I	68 (34%)
II	68 (34%)
III	64 (32%)

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

{gtsummary} summarize data with tbl_summary()

Start customizing using arguments and pipe operator `%>%` to string additional functions together

```
tbl_summary_1 ← sm_trial %>%
  tbl_summary(by = trt) %>%
  add_p()
```

- `by =` argument to split table by a categorical variable
- `add_p()` - default tests are the Wilcoxon rank-sum test for continuous variables, chi-square test of independence/ Fisher's exact test for categorical (Fisher's for low expected counts).
- `add_overall()` - to add back in an overall summary of the data (not split by the `by` argument)

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} summarize data with tbl_summary()

Customize further using formula syntax and tidy selectors

```
tbl_summary_3 ← sm_trial %>%
  tbl_summary(
    by = trt,
    statistic = list(
```

- `all_continuous() ~ "{mean} ({sd})",`
- `all_categorical() ~ "{n} / {N} ({p}%)"),`
- `label = vars(age) ~ "Patient Age") %>%`
- `add_p(test = all_continuous() ~ "t.test")`
- `statistic` - Report mean and standard deviation for continuous (default is median)

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} tbl_summary() Formulas

Formulas

- Most arguments to `tbl_summary()` require formula syntax, and provide many more options to easily select the table variables you want to modify. More on that later.

select variables ~ specify what you want to do

{gtsummary} summarize data with tbl_summary()

Additional `tbl_summary` Features

- Use **{tidyselect}** functions to select variables for customization
- Use **custom functions** for calculating p-values and reporting any statistic for continuous variables (inlcuding user-written functions)
- **Missing data** options
- **Sort variables** by significance (`sort_p()`); sort categorical variables by frequency
- Calculate **cell percents and row percents** (default is column-wide)
- Only report p-values for select variables (`add_p(include = ...)`); report q-values (like false discovery rate)
- **Rounding options** and ability to **set global options** for rounding p-values

Review `tbl_summary()` vignette for more details and examples!

tbl_regression()

{gtsummary} summarize models with tbl_regression()

Raw model output

⚠ Difficult to work with

⚠ Format varies from different types of models

⚠ Need to exponentiate betas to get odds ratios from a logistic regression etc.

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

```
##
## Call:  glm(formula = response ~ trt + grade + age, family
##     data = trial)
##
## Coefficients:
## (Intercept)    trtDrug B      gradeII     gradeIII
##   -1.694687     0.124330    -0.160518     0.007672     0.
##
## Degrees of Freedom: 182 Total (i.e. Null);  178 Residual
##   (17 observations deleted due to missingness)
## Null Deviance:        228.6
## Residual Deviance: 225.3     AIC: 235.3
```

{gtsummary} summarize models with tbl_regression()

`broom::tidy() output

♥ Using `broom::tidy()` a step in the right direction!

♥ All models returned with consistent table format (term, estimates, standard errors...)

⚠ Does not include reference groups

⚠ Needs additional modification before it can be presented

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

```
## # A tibble: 5 x 7
##   term        estimate std.error statistic p.value conf.l
##   <chr>          <dbl>     <dbl>     <dbl>   <dbl>    <db
## 1 (Intercept)    0.184     0.630    -2.69   0.00715   0.05
## 2 trtDrug B      1.13      0.321     0.387  0.699     0.60
## 3 gradeII        0.852     0.395    -0.406  0.685     0.38
## 4 gradeIII       1.01      0.385     0.0199 0.984     0.47
## 5 age            1.02      0.0114    1.67   0.0952    0.99
```

{gtsummary} summarize models with tbl_regression()

{gtsummary} Output

`tbl_regression()` accepts regression model object as input. Uses {broom} in the background, outputs table with nice defaults:

♥ Reference groups added to the table
♥ Sensible default number rounding and formatting
♥ Label attributes printed
♥ Certain model types detected ♥ Estimate header and footnote included

```
tbl_reg_1 ← tbl_regression(m1, exponentiate = TRUE)
```

Characteristic	OR[1]	95% CI[1]	p-value
Chemotherapy Treatment			
Drug A	—	—	
Drug B	1.13	0.60, 2.13	0.7
Grade			
I	—	—	
II	0.85	0.39, 1.85	0.7
III	1.01	0.47, 2.15	>0.9
Age, yrs	1.02	1.00, 1.04	0.10

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

{gtsummary} summarize models with tbl_regression()

Format variables with `tbl_regression` arguments

- `label` - Specify labels
- `show_single_row` - If variable is dichotomous (e.g. Yes/No), you can choose to print regression coefficient on a single row

```
tbl_reg_1a ← tbl_regression(
  m1,
  show_single_row = trt,
  label = trt ~ "Treatment B vs A",
  exponentiate = TRUE)
```

{gtsummary} summarize models with tbl_regression()

Format values with `tbl_regression` arguments

- `exponentiate` - default is `FALSE`
- `conf.level` - Specify between 0-1. Default 0.95.
- `estimate_fun`, `pvalue_fun` - Specify functions to round and format values
- `tidy_fun` - Specify a specific or custom tidier

```
tbl_reg_1b ← tbl_regression(
  m1,
  conf.level = 0.9,
  pvalue_fun = function(x) style_pvalue(x, digits = 2),
  exponentiate = TRUE)
```

{gtsummary} summarize models with tbl_regression()

```
library(survival)
tbl_reg_3 ←
  coxph(Surv(ttdeath, death) ~ trt + grade + age,
        data = trial) %>%
  tbl_regression(exponentiate = TRUE)
tbl_reg_4 ←
  tbl_merge(
    tbls = list(tbl_reg_1, tbl_reg_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()

```
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

inline_text()

{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_reg_1, variable = "trt", level = "Drug A")
```

```
NA (95% CI NA, NA; NA)
```

- 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})"`.

Customization

{gtsummary} tbl_summary() Formulas

Formulas

- Most arguments to `tbl_summary()` require formula syntax, and provide many more options to easily select the table variables you want to modify.

select variables ~ specify what you want to do

```
tbl_summary(
  trial,
  by = trt,
  statistic = age ~ "{mean} ({sd})"
)
```

{gtsummary} tbl_summary() Formulas

Formulas

select variables ~ specifiy what you want to do

- **select variables**
  - use quoted or unquoted variables, minus sign to negate (e.g. `age` or `"age"` to select, `-age` to deselect)
  - use any {tidyselect} functions, e.g. `contains("stage") ~ ...`
  - use attribute (e.g `all_character() ~ ...`) or type (e.g. , `all_continuous() ~ ...`)
- **specify what you want to do** (depends on the argument)
  - change the statistic you report using {glue} syntax. e.g `statistic = ... ~ "{mean} ({sd})"`
  - pass a string to change labels

{gtsummary} tbl_summary() Formulas

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

- Report levels for the response variable.
- Report mean instead of median (using glue)
- Modify the default rounding for age.
- Specify t-test for all continuous variables and Fisher's test for response variable.

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} tbl_summary() advanced customization

Advanced Customization Using {gt}

- 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} tbl_summary() advanced customization

Advanced Customization Using {gt}

- tab_header() - add a table title
- tab_spanner() - add headers that span columns
- tab_options() - change table padding and font size
- tab_footnote() - add additional footnotes to table

And many more! https://gt.rstudio.com/

{gtsummary} tbl_summary() advanced customization

Advanced Customization Using {gt}

```
tbl_summary_5 ← sm_trial %>%
  tbl_summary(by = trt) %>%
  # convert from gtsummary object to gt object
  as_gt() %>%
  # modify with gt functions
  tab_header("Table 1: Baseline Characteristics") %>%
  tab_spanner(
    label = "Randomization Group",
    columns = starts_with("stat_")
  ) %>%
  tab_options(
    table.font.size = "small",
    data_row.padding = gt::px(1))
```

Table 1: Baseline Characteristics

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 (%)

More on this in the `tbl_summary()` vignette

{gtsummary} Advanced

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

```
names(tbl_summary_1)
```

```
##  [1] "gt_calls"     "kable_calls"  "table_body"   "table_header" "meta_data"
##  [6] "inputs"       "N"            "call_list"    "by"           "df_by"
## [11] "fmt_fun"
```

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

```
## # A tibble: 6 x 6
##   variable row_type label          stat_1      stat_2
##   <chr>    <chr>    <chr>          <chr>       <chr>
## 1 age      label    Age, yrs       46 (37, 59) 48 (39, 56)
## 2 age      missing  Unknown        7           4
## 3 response label    Tumor Response 28 (29%)    33 (34%)
## 4 response missing  Unknown        3           4
## 5 grade    label    Grade          <NA>        <NA>
## 6 grade    level    I              35 (36%)    33 (32%)
```

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

```
## $gt
## gt::gt(data = x$table_body)
##
## $cols_align
## gt::cols_align(align = 'center') %>% gt::cols_align(align
##
## $fmt_missing
## gt::fmt_missing(columns = gt::everything(), missing_text
##
## $tab_style_text_indent
## gt::tab_style(style = gt::cell_text(indent = gt::px(10),
```

Conclusion

{gtsummary}

• Every function is documented further in the help file •

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

📖 {gtsummary} documentation/website danieldsjoberg.com/gtsummary/

{gtsummary} package github.com/ddsjoberg/gtsummary

slides at github.com/margarethannum/gtsummary-presentation-rladies

source code for slides at github.com/margarethannum/gtsummary-presentation-rladies

{gt} package github.com/rstudio/gt

Download {gtsummary} today!

Thank you

Author/Maintainer: Daniel Sjoberg

Authors: Margie Hannum, Karissa Whiting, Emily Zabor, Michael Curry

Contributors: Esther Drill, Jessica Flynn