

Tangram Examples

Shawn Garbett, MS

2018-12-04

A Grammar of Tables

This package is meant to implement the concept of a grammar of tables. It allows for a simple formula expression and a data frame to create a rich summary table in a variety of formats. It is designed for extensibility at each step of the process, so that one is not limited by the authors choice of table statistics, output format. The grammar however is an integral part of the package, and as such is not modifiable.

Here's an example similar to summaryM from Hmisc to get us started:

```
tbl1 <- tangram("drug~bili+albumin+stage::Categorical+protime+sex+age+spiders",
  pbc, id="joe", caption="Context Aware Compilation")
summary(tbl1)
```

Context Aware Compilation

	N	D-penicillamine (N=154)		placebo (N=158)		not randomized (N=106)	
Serum Bilirubin (mg/dl)	418	0.70	*1.30*	3.60	0.80	*1.40*	3.12
Albumin (gm/dl)	418	3.34	*3.54*	3.78	3.21	*3.56*	3.83
Histologic Stage, Ludwig Criteria	412						
1		0.026	4/154	0.076	12/158	0.050	5/100
2		0.208	32/154	0.222	35/158	0.250	25/100
3		0.416	64/154	0.354	56/158	0.350	35/100
4		0.351	54/154	0.348	55/158	0.350	35/100
Prothrombin Time (sec.)	416	10.0	*10.6*	11.4	10.0	*10.6*	11.0
sex : female	418	0.903	139/154	0.867	137/158	0.925	98/106
Age	418	41.4	*48.1*	55.8	42.9	*51.9*	59.0
spiders : present	312	0.292	45/154	0.285	45/158		

N is the number of non-missing value. ^1 Kruskal-Wallis. ^2 Pearson. ^3 Wilcoxon.

Notice that stage in the formula wasn't stored as a factor, i.e. Categorical variable, so by adding a type specifier in the formula given, it is treated as a Categorical. There is no preconversion applied to the data frame, nor is there a guess based on the number of unique values. Full direct control of typing is provided in the formula specification.

It also supports HTML5, with styling fragments.

Rmd Direct Example

```
x <- tangram("drug~bili[2]+albumin+stage::Categorical+protime+sex+age+spiders",
  pbc,
  msd=TRUE,
  quant=seq(0, 1, 0.25),
  style="hmisc",
  caption = "Table Rmd Style",
  pct_width=0.6)
rmd(x)
```

##

Table Rmd Style

N	D-penicillamine	placebo	not randomized
	(N=154)	(N=158)	(N=106)

	N	D-penicillamine				placebo				not randomized			
Serum	418	0.30	0.70	1.30	3.60	0.30	0.80	0.40	0.70	1.40	3.12		
Bilirubin (mg/dl)		28.00	3.65±5.28			1.40	3.22	18.00	3.12±4.04				
Albumin (gm/dl)	418	1.96	3.34	3.54	3.78	2.10	3.21	2.31	3.12	3.47	3.73		
		4.38	3.52±0.40			3.56	3.83	4.52	3.43±0.43				
						4.64	3.52±0.44						
Histologic Stage, Ludwig Criteria	412												
1		0.026	4/154			0.076	12/158	0.050	5/100				
2		0.208	32/154			0.222	35/158	0.250	25/100				
3		0.416	64/154			0.354	56/158	0.350	35/100				
4		0.351	54/154			0.348	55/158	0.350	35/100				
Prothrombin Time (sec.)	416	9.2	10.0	10.6	11.4	9.0	10.0	10.6	9.0	10.1	10.6	11.0	
		17.1	10.8±1.1			11.0		18.0	10.8±1.1				
						14.1	10.7±0.9						
sex : female	418	0.903	139/154			0.867	137/158	0.925	98/106				
Age	418	30.6	41.4	48.1	55.8	26.3	42.9	33.0	46.0	53.0	61.1		
		74.5	48.6±10.0			51.9	59.0	75.0	52.9±9.8				
						78.4	51.4±11.0						
spiders :	312	0.292	45/154			0.285	45/158						
present													

N is the number of non-missing value. ¹Kruskal-Wallis. ²Pearson. ³Wilcoxon.

Hmisc Style Example

```
tangram("drug~bili[2]+albumin+stage::Categorical+protime+sex+age+spiders",
  pbc,
  msd=TRUE,
  quant=seq(0, 1, 0.25),
  style="hmisc",
  caption = "Table Hmisc Style",
  pct_width=0.6,
  capture_units=TRUE
)
```

	N	D-penicillamine				placebo				not randomized			
		(N=154)				(N=158)				(N=106)			
Serum Bilirubin <i>mg/dl</i>	418	0.30	0.70	1.30	3.60 28.00	3.65±5.28	0.30	0.80	1.40	3.22 20.00	2.87±3.63	0.40	0.70 1.40 3.12 18.00 3.12±4.04
Albumin <i>gm/dl</i>	418	1.96	3.34	3.54	3.78 4.38	3.52±0.40	2.10	3.21	3.56	3.83 4.64	3.52±0.44	2.31	3.12 3.47 3.73 4.52 3.43±0.43
Histologic Stage, Ludwig Criteria	412												
1		0.026	⁴ / ₁₅₄				0.076	¹² / ₁₅₈		0.050	⁵ / ₁₀₀		
2		0.208	³² / ₁₅₄				0.222	³⁵ / ₁₅₈		0.250	²⁵ / ₁₀₀		
3		0.416	⁶⁴ / ₁₅₄				0.354	⁵⁶ / ₁₅₈		0.350	³⁵ / ₁₀₀		
4		0.351	⁵⁴ / ₁₅₄				0.348	⁵⁵ / ₁₅₈		0.350	³⁵ / ₁₀₀		
Prothrombin Time <i>sec.</i>	416	9.2	10.0	10.6	11.4 17.1	10.8±1.1	9.0	10.0	10.6	11.0 14.1	10.7±0.9	9.0	10.1 10.6 11.0 18.0 10.8±1.1
sex : female	418	0.903	¹³⁹ / ₁₅₄				0.867	¹³⁷ / ₁₅₈		0.925	⁹⁸ / ₁₀₆		
Age	418	30.6	41.4	48.1	55.8 74.5	48.6±10.0	26.3	42.9	51.9	59.0 78.4	51.4±11.0	33.0	46.0 53.0 61.1 75.0 52.9±9.8
spiders : present	312	0.292	⁴⁵ / ₁₅₄				0.285	⁴⁵ / ₁₅₈					

Table 2: Table Hmisc Style

N is the number of non-missing value. ¹Kruskal-Wallis. ²Pearson. ³Wilcoxon.

NEJM Style Example

Fragments can have localized style sheets specified by given id. Notice the specification of 2 digits for bilirubin in the formula.

```
tangram("drug~bili[2]+albumin+stage::Categorical+prottime+sex+age+spiders", pbc,
        style="nejm", caption = "Table NEJM Style",
        pct_width=0.8,
        capture_units=TRUE)
```

Table NEJM Style				
	N	D-penicillamine	placebo	not randomized
		(N=154)	(N=158)	(N=106)
Serum Bilirubin (mg/dl)	418			
Median (interquartile range)		1.30 (0.70—3.60)	1.40 (0.80—3.22)	1.40 (0.70—3.12)
Range		0.30—28.00	0.30—20.00	0.40—18.00
Albumin (gm/dl)	418			
Median (interquartile range)		3.54 (3.34—3.78)	3.56 (3.21—3.83)	3.47 (3.12—3.73)
Range		1.96—4.38	2.10—4.64	2.31—4.52
Histologic Stage, Ludwig Criteria	412			
1		4/154 (2.6)	12/158 (7.6)	5/100 (5.0)
2		32/154 (20.8)	35/158 (22.2)	25/100 (25.0)
3		64/154 (41.6)	56/158 (35.4)	35/100 (35.0)
4		54/154 (35.1)	55/158 (34.8)	35/100 (35.0)
Prothrombin Time (sec.)	416			
Median (interquartile range)		10.6 (10.0—11.4)	10.6 (10.0—11.0)	10.6 (10.1—11.0)
Range		9.2—17.1	9.0—14.1	9.0—18.0
sex : female	418	139/154 (90.3)	137/158 (86.7)	98/106 (92.5)
Age	418			
Median (interquartile range)		48.1 (41.4—55.8)	51.9 (42.9—59.0)	53.0 (46.0—61.1)
Range		30.6—74.5	26.3—78.4	33—75
spiders : present	312	45/154 (29.2)	45/158 (28.5)	

Table 3: Table NEJM Style

N is the number of non-missing value. ¹Kruskal-Wallis. ²Pearson. ³Wilcoxon.

Lancet Style Example

Fragments can have localized style sheets specified by given id. Specifications of digit output can also be c specifiers if enclosed in quotes.

```
tangram('drug~bili["%4.03f"]+albumin+stage::Categorical[1]+prottime+sex[1]+age+spiders[1]',
        data=pbcc,
        pformat = 5,
        style="lancet",
        caption = "Table Lancet Style",
        pct_width = 0.8,
        capture_units=TRUE
    )
```

	N	D-penicillamine (N=154)	placebo (N=158)	not randomized (N=106)
Serum Bilirubin <i>mg/dl</i>	418	3.649 (5.282)	2.873 (3.629)	3.117 (4.043)
Albumin <i>gm/dl</i>	418	3.52 (0.40)	3.52 (0.44)	3.43 (0.43)
Histologic Stage, Ludwig Criteria	412			
1		4 (2.6%)	12 (7.6%)	5 (5.0%)
2		32 (20.8%)	35 (22.2%)	25 (25.0%)
3		64 (41.6%)	56 (35.4%)	35 (35.0%)
4		54 (35.1%)	55 (34.8%)	35 (35.0%)
Prothrombin Time <i>sec.</i>	416	10.8 (1.1)	10.7 (0.9)	10.8 (1.1)
sex : female	418	139 (90.3%)	137 (86.7%)	98 (92.5%)
Age	418	48.6 (10.0)	51.4 (11.0)	52.9 (9.8)
spiders : present	312	45 (29.2%)	45 (28.5%)	

Table 4: Table Lancet Style

N is the number of non-missing value. ¹Kruskal-Wallis. ²Pearson. ³Wilcoxon.

Intercept Model Example

```
x <- round(rnorm(375, 79, 10))
y <- round(rnorm(375, 80, 9))
y[rbinom(375, 1, prob=0.05)] <- NA
attr(x, "label") <- "Global score, 3m"
attr(y, "label") <- "Global score, 12m"
tangram(1 ~ x+y,
  data.frame(x=x, y=y),
  style="hmisc", caption="Intercept Example", id="tbl5") %>%
del_row(2) %>% del_col(4)
```

	N
Global score, 3m	375
Global score, 12m	374

Table 5: Intercept Example

N is the number of non-missing value. ¹Kruskal-Wallis. ²Pearson. ³Wilcoxon.

Tables with Multicolumns/Multirow

What about the `table` object in R? Can I have multicol/multirow output for cells?

The answer is yes to both but multi row/col is only for supported rendering formats, i.e. HTML and LaTeX.

```
tangram(with(warpbreaks, table(wool, tension)), id="warpbreaks") %>%
insert_row(0, cell(""), cell("Tension", colspan=3), NA, NA) %>%
insert_column(0, "", cell("Wool", rowspan=3), NA, NA)
```

		Tension		
		L	M	H
Wool	A	9	9	9
	B	9	9	9

Table 6:

Types

The Hmisc default style recognizes 3 types: Categorical, Bionimial, and Numerical. Then for each product of these two, a function is provided to generate the corresponding rows and columns. As mentioned before, the user can declare any type in a formula, and one is not limited to the Hmisc defaults. This is completely customizable, which will be covered later.

Let's cover the phases of table generations.

1. Syntax. The formula is parsed into an abstract syntax tree (AST), and factors are right distributed, and the data frame is split into appropriate pieces attached to each node in the AST. The syntax and parser are the only portions of this library that are fixed, and not customizable. The grammar may expand with time, but cautiously as to not create an overly verbose set of possibilities to interpret. The goal is to create a clean grammar that describes the bold areas of a table to fill in.
2. Semantics. The elements of the AST are examined, and passed to compilation functions. The compilation function is chosen by determining the type of the row variable, and the type of column variable. For example, `drug ~ stage::Categorical`, is a `Categorical×Categorical` which references the `summarize_chisq` for compiling. One can easily specify different compilers for a formula and get very different results inside a formula. Note: the application of multiplication `*` cannot be done in the previous phase, because this involves semantic meaning of what multiplication means. In one context it might be an interaction, in another simple multiplication. Handling multiplicative terms can be tricky. Once compiling is finished a table object composed of cells (list of lists) which are one of a variety of S3 types is the result.
3. Rendering. With a compiled table object in memory, the final stage is conversion to an output format which could be plain text, HTML5, LaTeX or anything. These are overrideable via S3 classes representing the different possible types of cells that are present inside a table. User specified rendering is possible as well.

Summary columns

A simple example of using an intercept in a formula, with some post processing to remove undesired columns.

```
d1 <- iris
d1$A <- d1$Sepal.Length > 5.1
attr(d1$A,"label") <- "Sepal Length > 5.1"
tangram(
  Species + 1 ~ A + Sepal.Width,
  data = d1,
  style="nejm",
  caption="Example All Summary"
) %>%
drop_statistics() %>%
del_col(6)
```

Example All Summary					
	N	setosa	versicolor	virginica	All
		(N=50)	(N=50)	(N=50)	(N=150)
Sepal Length > 5.1 : TRUE	150	14/50 (28.0)	46/50 (92.0)	49/50 (98.0)	109/150 (72.7)
Sepal.Width	150				
Median (interquartile range)		3.40 (3.19—3.70)	2.80 (2.50—3.00)	3.00 (2.80—3.20)	3.00 (2.80—3.31)
Range		2.30—4.40	2.00—3.40	2.20—3.80	2.00—4.40

Table 7: Example All Summary

N is the number of non-missing value. ¹Kruskal-Wallis. ²Pearson. ³Wilcoxon.

Extensibility

The library is designed to be extensible, in the hopes that more useful summary functions can generate results into a wide variety of formats. This is done by the translator functions, which given a row and column from a formula will process the data into a table.

This example shows how to create a function that given a row and column, to construct summary entries for a table.

```
### Make up some data, which has events nested within an id
n <- 1000
df <- data.frame(id = sample(1:250, n*3, replace=TRUE), event = as.factor(rep(c("A", "B", "C"), n)))
attr(df$id, "label") <- "ID"

### Now create custom function for counting events with a category
summarize_count <- function(table, row, column, ...)
{
  ### Getting Data for row column ast nodes, assuming no factors
  datar <- row$data
  datac <- column$data

  ### Grabbing categories
  col_categories <- levels(datac)

  n_labels <- lapply(col_categories, FUN=function(cat_name){
    x <- datar[datac == cat_name]
    cell_n(length(unique(x)), subcol=cat_name)
  })

  # Test a poisson model
  test <- summary(aov(glm(x ~ treatment,
    aggregate(datar, by=list(id=datar, treatment=datac), FUN=length),
    family=poisson)))[[1]]
  test <- hmisc_fstat(f = render_f(test$'F value'[1], "%.2f"),
    df1 = test$Df[1], df2 = test$Df[2],
    p = hmisc_p(test$'Pr(>F)')[1]))

  # Build the table
  table %>%
  # Create Headers
  row_header(derive_label(row)) %>%
  col_header("N", col_categories, "Test Statistic") %>%
```

```

col_header("", n_labels, "" ) %>%
# Add the First column of summary data as an N value
add_col(cell_n(length(unique(datar)))) %>%
# Now add quantiles for the counts
table_apply(col_categories, FUN=
  function(tbl, cat_name) {
    # Compute each data set
    x <- datar[datar == cat_name]
    xx <- aggregate(x, by=list(x), FUN=length)$x

    # Add a column that is a quantile
    add_col(tbl, hmisc_iqr(xx, row$format, na.rm=TRUE))
  }) %>%
# Now add a statistical test for the final column
add_col(test)
}

tangram(event ~ id["%1.0f"], df, id="tbl7", transforms=summarize_count)

```

	N	A	B	C	Test Statistic
		247	247	247	
ID	250	3 4 5	3 4 5	3 4 5	$F_{2,738}=0.00$, $P=1.000^1$

Table 8:

And that concludes the basic demonstration of formula capabilities and writing one's own simple transform bundle.