MasteR of Tables

Tomasz Żółtak

Educational Research Institute (Warsaw, Poland) & University of Warsaw

WhyR? Conference Warsaw, 29.09.2019

Section 1

Introduction

Introductory remarks

- Initially I planned this to be a workshop (for beginners)...
- This speech will be only about package *tables*
- I haven't authored this package but I find it very useful and want to promote it's use

Who cares about tables?!

- Everyone sometimes needs to make a table (of percentages)

Who cares about tables?!

- Everyone sometimes needs to make a table (of percentages)
- If we ask "WhyR?"...

Who cares about tables?!

- Everyone sometimes needs to make a table (of percentages)
- If we ask "WhyR?"...
- ...many people need to make a lot of tables in their everyday work

Who cares about tables?!

- Everyone sometimes needs to make a table (of percentages)
- If we ask "WhyR?"...
- ...many people need to make a lot of tables in their everyday work
- Improving popularity of R in some domains (like social science) needs not only enabling people to use new, sophisticated, exciting methods of data analysis...

Who cares about tables?!

- Everyone sometimes needs to make a table (of percentages)
- If we ask "WhyR?"...
- ...many people need to make a lot of tables in their everyday work
- Improving popularity of R in some domains (like social science) needs not only enabling people to use new, sophisticated, exciting methods of data analysis...
- ...but also ability to support them in performing easy and rather burdensome everyday tasks.

But it's already so easy! (1)

Let's try to make a rather simple table summarizing *Titanic* data:

	Count			Percent		
	Survived			Survived		
Class	No	Yes	Total	No	Yes	Total
1st	122	203	325	37.54	62.46	100
2nd	167	118	285	58.60	41.40	100
3rd	528	178	706	74.79	25.21	100
Crew	673	212	885	76.05	23.95	100
Total	1490	711	2201	67.70	32.30	100

assuming we have this data provided as a data frame:

```
str(titanic)
```

```
## 'data.frame':    2201 obs. of  4 variables:
##  $ Class   : Factor w/ 4 levels "1st","2nd","3rd",..: 3 3 3 3 3 3 3
##  $ Sex     : Factor w/ 2 levels "Male","Female": 1 1 1 1 1 1 1 1 1 1
##  $ Age     : Factor w/ 2 levels "Child","Adult": 1 1 1 1 1 1 1 1 1 1
##  $ Survived: Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
```

But it's already so easy! (2)

Neither so easy nor so pretty:

```
library(knitr)
counts = table(Class = titanic$Class, Survived = titanic$Survived)
counts = addmargins(counts, 1, list(Total = sum))
percents = 100*prop.table(counts, 1)
percents = addmargins(percents, 2, list(Total = sum))
counts = addmargins(counts, 2, list(Total = sum))
colnames(counts) = paste("#", colnames(counts))
colnames(percents) = paste("%", colnames(percents))
kable(cbind(counts, percents), digits = 2)
```

	# No	# Yes	# Total	% No	% Yes	% Total
1st	122	203	325	37.54	62.46	100
2nd	167	118	285	58.60	41.40	100
3rd	528	178	706	74.79	25.21	100
Crew	673	212	885	76.05	23.95	100
Total	1490	711	2201	67.70	32.30	100

But it's already so easy! (2)

Neither so easy nor so pretty:

```
library(knitr)
counts = table(Class = titanic$Class, Survived = titanic$Survived)
counts = addmargins(counts, 1, list(Total = sum))
percents = 100*prop.table(counts, 1)
percents = addmargins(percents, 2, list(Total = sum))
counts = addmargins(counts, 2, list(Total = sum))
colnames(counts) = paste("#", colnames(counts))
colnames(percents) = paste("%", colnames(percents))
kable(cbind(counts, percents), digits = 2)
```

	# No	# Yes	# Total	% No	% Yes	% Total
1st	122	203	325	37.54	62.46	100
2nd	167	118	285	58.60	41.40	100
3rd	528	178	706	74.79	25.21	100
Crew	673	212	885	76.05	23.95	100
Total	1490	711	2201	67.70	32.30	100

And it won't be any easier with *tidyverse*.

Package *tables*

- Authored by Duncan Murdoch
  - *Inspired by my* [Duncan's] *20 year old memories of SAS PROC TABULATE*
- *Computes and displays complex tables of summary statistics. Output may be in LaTeX, HTML, plain text, or an R matrix for further processing.*
  - *tables* uses *knitr* package to prepare HTML or LaTeX output of its objects (*kableExtra* can be used as well to make this output even better)
- Allows describing structure of a table with convenient formula interface.

```
tabular(Class + (Total=1) ~
          ((Count=length) + Percent("row"))*(Survived + (Total=1)),
        data = titanic) %>%
  toKable("latex")
```

	Count			Percent		
	Survived			Survived		
Class	No	Yes	Total	No	Yes	Total
1st	122	203	325	37.54	62.46	100
2nd	167	118	285	58.60	41.40	100
3rd	528	178	706	74.79	25.21	100
Crew	673	212	885	76.05	23.95	100
Total	1490	711	2201	67.70	32.30	100

Section 2

Features of the *tables* package

Formula syntax - numeric variables (1)

- `what in rows ~ what in columns`
- *numeric* variables:
  - `variable * function` to get statistic returned by a function
  - `(label=variable)` or `(label=function)` to assign labels
  - `(variable > condition)` to define subsets
  - `Factor(variable)` to treat variable as categorical (i.e. grouping factor)
  - `foo + foo` to define consecutive rows/columns
  - `+ 1` indicates adding *total* row/column

```
tabular((cyl == 4) + (cyl > 4) + 1 ~
          (Horsepower=hp)*(mean + sd),
        mtcars) %>% toKable("latex")
```

	Horsepower	
	mean	sd
cyl == 4	82.64	20.93
cyl > 4	180.24	60.24
All	146.69	68.56

Formula syntax - numeric variables (2)

- `what in rows ~ what in columns`
- *numeric* variables:
  - `variable * function` to get statistic returned by a function
  - `(label=variable)` or `(label=function)` to assign labels
  - `(logical expression)` to define subsets
  - `Factor(variable)` to treat variable as categorical (i.e. grouping factor)
  - `+` to define consecutive rows/columns
  - `+ 1` indicates adding *total* row/column

```
tabular((cyl == 4) + (cyl > 4) + 1 ~ (Horsepower=hp)*(
  (q0.2 = quantile*Arguments(probs = 0.2)) +
    (q0.8 = quantile*Arguments(probs = 0.8)) +
    (`Number of cases`=length)),
  mtcars) %>% toKable("latex")
```

	Horsepower		
	q0.2	q0.8	Number of cases
cyl == 4	65.0	97	11
cyl > 4	123.0	230	21
All	93.4	200	32

Formula syntax - additional features (1)

- You can describe complex table structure using parenthesis
- *Pseudofunction* `Format()` allows defining how to print numbers
- *Pseudofunction* `Heading()` allows not to put variable/function name into rows/columns names

```
tabular((cyl == 4) + (cyl > 4) + 1 ~
          ((Horsepower=hp) + (`Miles per galon`=mpg))*
          (mean + sd)*
          Format(digits=3, nsmall=3) +
          (`Number of cases`=length)*Heading()*hp,
        mtcars) %>% toKable("latex")
```

	Horsepower		Miles per galon		
	mean	sd	mean	sd	Number of cases
cyl == 4	82.636	20.935	26.664	4.510	11
cyl > 4	180.238	60.240	16.648	3.150	21
All	146.688	68.563	20.091	6.027	32

Formula syntax - categorical variables (1)

- `what in rows ~ what in columns`
- *categorical* variables - always use factors:
  - `Factor(variable)` to covert character variables to factors on the fly
  - `variable` to get counts; `variable1 * variable2` to get subgroups
  - `(label=variable)` to assign label
  - `variable*Percent("row")` or `variable*Percent("col")` to compute percentages
  - `+ 1` indicates adding *total* row/column

```
tabular(Survived ~ (Age * Class),
        titanic) %>% toKable("latex")
```

	Age							
	Child				Adult			
	Class				Class			
Survived	1st	2nd	3rd	Crew	1st	2nd	3rd	Crew
No	0	0	52	0	122	167	476	673
Yes	6	24	27	0	197	94	151	212

Formula syntax - categorical variables (2)

- `what in rows ~ what in columns`
- *categorical* variables (always use factors):
  - `Factor(variable)` to covert character variables to factors on the fly
  - `variable` to get counts; `variable1 * variable2` to get subgroups
  - `(label=variable)` to assign label
  - `variable*Percent("row")` or `variable*Percent("col")` to compute percentages
  - `+ 1` indicates adding *total* row/column

```
tabular(Survived + 1 ~ ((Age + 1) * (Sex + 1))*
          (pct.=Percent("col")),
        titanic) %>% toKable("latex")
```

	Age								
	Child			Adult			All		
	Sex			Sex			Sex		
Survived	Male pct.	Female pct.	All pct.	Male pct.	Female pct.	All pct.	Male pct.	Female pct.	All pct.
No	54.69	37.78	47.71	79.72	25.65	68.74	78.8	26.81	67.7
Yes	45.31	62.22	52.29	20.28	74.35	31.26	21.2	73.19	32.3
All	100.00	100.00	100.00	100.00	100.00	100.00	100.0	100.00	100.0

Formula syntax - additional features (2)

- *Pseudofunction* `All()` allows to use all variables in the data
  - You can indicate what kind of variables should be used (by default function uses only numeric ones but you can change this using function arguments)

```
tabular(All(iris)*(mean + sd)*Format(digits=2, nsmall=2) ~
          Species,
        iris) %>% toKable("latex")
```

		Species		
		setosa	versicolor	virginica
Sepal.Length	mean	5.01	5.94	6.59
	sd	0.35	0.52	0.64
Sepal.Width	mean	3.43	2.77	2.97
	sd	0.38	0.31	0.32
Petal.Length	mean	1.46	4.26	5.55
	sd	0.17	0.47	0.55
Petal.Width	mean	0.25	1.33	2.03
	sd	0.11	0.20	0.27

Formula syntax - additional features (3)

- *Pseudofunction* `DropEmpty()` allows dropping empty (i.e. regrading to non existing observations) rows/columns
  - Below: no column for Child-Crew (column containing only 0 cells); other zero cells content replaced with "—"
  - Useful not to have `NaNs` while computing percentages

```
tabular(Survived + 1 ~ (Age * Class)*(`\\%`=Percent("col"))*
          DropEmpty(empty="---"),
        titanic) %>% toKable("latex")
```

	Age						
	Child			Adult			
	Class			Class			
Survived	1st %	2nd %	3rd %	1st %	2nd %	3rd %	Crew %
No	—	—	65.82	38.24	63.98	75.92	76.05
Yes	100	100	34.18	61.76	36.02	24.08	23.95
All	100	100	100.00	100.00	100.00	100.00	100.00

Formula syntax - additional features (4)

- *Pseudofunction* `Paste()` allows constructing complex cell content, i.e. showing confidence intervals

```
ci_lb <- function(x) {
  mean(x) - qt(0.975, df=length(x)-1) * sd(x) / sqrt(length(x))}
ci_ub <- function(x) {
  mean(x) + qt(0.975, df=length(x)-1) * sd(x) / sqrt(length(x))}
tabular(Species ~
          (`Sepal length`=Sepal.Length)*
          (mean + Paste(ci_lb, ci_ub,
                        digits = 2, head = "95\\% CI",
                        sep = ",", prefix = "[", postfix = "]")),
        iris) %>% toKable("latex")
```

	Sepal length	
Species	mean	95% CI
setosa	5.006	[4.9, 5.1]
versicolor	5.936	[5.8, 6.1]
virginica	6.588	[6.4, 6.8]

Another way of using *tables*

- Sometimes one needs to compute (complicated) statistics with some other tools
- Still, one can use *tables* to easy arrange computed statistics into a table in a way you want
- I.e. survey data with weights or even complex sampling schemes
  1. Compute statistics with `svyby()` or `svytable()` from the *survey* package
  2. Use `as.dataframe()` method on these results
  3. Provide these as a data to `tabular()`; use function `indentity` or `sum` respectively as a summary function in a formula
  4. If you want to compute percentages, define function: `percent_sum = function(x, y) 100 * sum(x) / sum(y)` and provide it while calling `Percent()` *pseudofunction* within a formula as `fn` argument

The end

Thank you for your attention!

Tomasz Żółtak
tomek@zozlak.org