# The mosaic Package

*by Randall Pruim, Nicholas J. Horton and Daniel T. Kaplan*

**Abstract** An abstract of less than 150 words.

Introductory section which may include references in parentheses (**?**), or cite a reference such as **?** in the text.

## Project MOSAIC

Project MOSAIC is an NSF-funded initiative to provide a broader approach to quantitative studies that provides better support for work in science and technology. The focus of the project has been to tie together better diverse aspects of quantitative work that students in science, technology, and engineering will need in their professional lives, but which are usually taught in isolation, if at all.

The name MOSAIC reflects a vision of quantitative education where the basic components come together to form a complete and compelling picture. The name also loosely reflects the first letters – M, S, C, C – of four important components of a quantitative education:

Modeling. The ability to create, manipulate and investigate useful and informative mathematical representations of a real-world situations.

Statistics. The analysis of variability that draws on our ability to quantify uncertainty and to draw logical inferences from observations and experiment.

Computation. The capacity to think algorithmically, to manage data on large scales, to visualize and interact with models, and to automate tasks for efficiency, accuracy, and reproducibility.

Calculus. The traditional mathematical entry point for college and university students and a subject that still has the potential to provide important insights to today's students.

The **mosaic** package originated in early attempts by each of the authors to ease new users (student, primarily) into using R, primarily in the context of other scientific disciplines.

## Expanding the formula interface

### Less volume, more creativity

One of the guiding principles behind the development of the **mosaic** package has been "Less volume, more creativity". Beginners are easily overwhelmed by the scope of R and its many packages: Often there are multiple ways to accomplish the same task, and authors of the many packages are not required to follow any particular style guidelines. Early on in the development of **mosaic**, we decided to reduce the number of code templates that users would need to know to as few as possible, while still providing them with substantial power to be creative within the templates provided.

Our most important template makes use what we call the "formula-interface" and is modeled after `lm()` and the plotting functions in **lattice**. We present the template as

```
goal(y ~ x | z, data = mydata)
```

or even more simply as

```
goal(formula, data = mydata)
```

This emphasizes that functions in R are typically named after the object they produce (the goal) and that a wide variety of goals can be achieved using a single template. Having learned the formula interface to make **lattice** plots, new users are well prepared for modeling with `lm()` and `glm()` when the time comes.

### Adding the formula interface to existing functions

Although this formula interface works well for graphical summaries of data, the same is not true of the standard numerical summary functions (e.g., `mean()`, `median()`, `sd()`, etc.). One of our initial difficult decisions was whether to add such a formula interface to these standard functions (by masking them)

or create new functions. We decided on the former option since the standard names are natural and familiar and having two sets of functions for essentially the same task would be unnecessarily complicated for new users. The **mosaic** versions of these functions can still be used as in **stats** and **base** so that existing code should not be affected.

```
mean(HELPrct$age)
[1] 35.65
mean(~age, data = HELPrct)  # same result as above
[1] 35.65
mean(age ~ sex, data = HELPrct)  # additional features are possible
female   male
 36.25  35.47
```

There are, of course, other ways to achieve this last result, including the use of `aggregate()`, or `summary()` from **Hmisc**, or the functions in **plyr**. But each of these solutions would require learning additional coding templates. In our approach, the following (and many more) are all identical except for the specification of the desired result via the function name.

```
   bwplot( age ~ sex, data=HELPrct )
     mean( age ~ sex, data=HELPrct )
       sd( age ~ sex, data=HELPrct )
       lm( age ~ sex, data=HELPrct )
   t.test( age ~ sex, data=HELPrct )   # formula interface exists in stats
```

Similarly, for one-variable situations we have

```
      mean( ~ age, data=HELPrct )
        sd( ~ age, data=HELPrct )
 histogram( ~ age, data=HELPrct )
    t.test( ~ age, data=HELPrct )       # formula interface added in mosaic
binom.test( ~ sex, data=HELPrct )       # formula interface added in mosaic
 prop.test( ~ sex, data=HELPrct )       # formula interface added in mosaic
```

And for plots and numerical summaries, we can add covariates into the mix as well:

```
     mean( ~ age | sex, data=HELPrct )
       sd( ~ age | sex, data=HELPrct )
histogram( ~ age | sex, data=HELPrct )
   t.test( ~ age | sex, data=HELPrct ) # formula interface added in mosaic (TO DO!!)
```

## Some things that we didn't change

The formula interface is powerful and flexible. Nevertheless, there are some sticky issues that we have chosen not to address.

### The geometry of lattice formlas

In `lm()`, `glm()` and the numerical summary functions, `y ~ x` can typically be read as "`y` depends on `x`" or "`y` is modeled by `x`". From this perspective,

```
histogram(age ~ sex, data = HELPrct)
```

would be perfectly reasonable. But **lattice**, for the most part, uses the `y` and `x` slots to indicate which axis of the plot is used for which variable. So the correct **lattice** command for the desired pair of histograms is

```
histogram(~age | sex, data = HELPrct)
```

and for overlaid density plots it is

```
histogram(~age, groups = sex, data = HELPrct)
```

While this can produce some initial challenges for users, clearly explaining the roles of the components of the variables for plotting, for numerical summaries, and for model fitting helps demystify the situation. Furthermore, in several functions we promote formulas of the form `x | z` to `x z` to allow either format. Similarly, the `groups` argument can often be used, for example

```
mean(~age, groups = sex, data = HELPrct)

Error:  object 'sex' not found
```

### Handling missing data

### Other stuff as we think of it

## Extracting information

Modeled on functions like `resid()`, a number of additional functions have been added to **mosaic** to facilitate extracting information from more complicated objects. Some examples include

```
confint(t.test(~age, data = HELPrct))  # works for htest objects
mean of x     lower     upper     level
    35.65     34.94     36.37      0.95
pval(t.test(age ~ sex, data = HELPrct))  # works for htest objects
p.value
 0.3537
stat(t.test(age ~ sex, data = HELPrct))  # works for htest objects
     t
0.9298
r.squared(lm(age ~ sex, data = HELPrct))
[1] 0.00187
```

## Simplifying randomization

### Using do() for repetition

### sample(), resample(), and shuffle()

### do() and confint()

### statTally()

## Calculus in R

## Some additional bells and whistles

## Summary

This file is only a basic article template. For full details of *The R Journal* style and information on how to prepare your article for submission, see the Instructions for Authors.

## Bibliography

*Randall Prim*
*Calvin College*

*3201 Burton St SE*
*Grand Rapids, MI 49546*
*USA* rpruim@calvin.edu

*Author Two*
*Affiliation*
*Address*
*Country* author2@work

*Author Three*
*Affiliation*
*Address*
*Country* author3@work