Principles and Practice of Data Analysis

for Reproducible Research in R

Publication Quality Reports with **knitr**

Heather Turner

Department of Statistics, University of Warwick

2016–09–29

Markdown

Markdown is a *lightweight* markup language.

This means it is relatively easy to learn and use, but inevitably has limitations

- lack of control over page layout: page breaks, columns, figure placement
- markup may be insufficient for complex tables, nested elements, etc
- it can be hard to meet journal style requirements

The **rmarkdown** Package

RStudio uses the **rmarkdown** package to convert `.Rmd` files to HTML, PDF or Word docx.

There are two steps to this process

- use the `knitr` package to create a `.md` file with the R chunks replaced by their output
- use the `pandoc` software to convert the `.md` file to HTML, PDF (via LaTeX) or docx.

Beyond Markdown

The limitations of markdown can be overcome in a number of ways

- using markup language of the intermediate/final output, e.g. HTML for HTML output, LaTeX for PDF output
- adding metadata to specify parameters used in the intermediate/final output, e.g. add header HTML, use an add-on LaTeX package
- using a custom template for the intermediate/final output

These all require knowledge of how to produce the intermediate/final output directly.

HTML, LaTeX or Word?

Word document templates can control the general layout e.g. page margins, fonts. But some aspects cannot be controlled from markdown (e.g. centering figures) and converting to word does not offer richer markup.

HTML offers more control over layout, but is not commonly used by journals.

LaTeX offers full control over layout and style files are often provided by journals. More options available to annotate PDFs vs HTML for collaborators that don't want to edit source code.

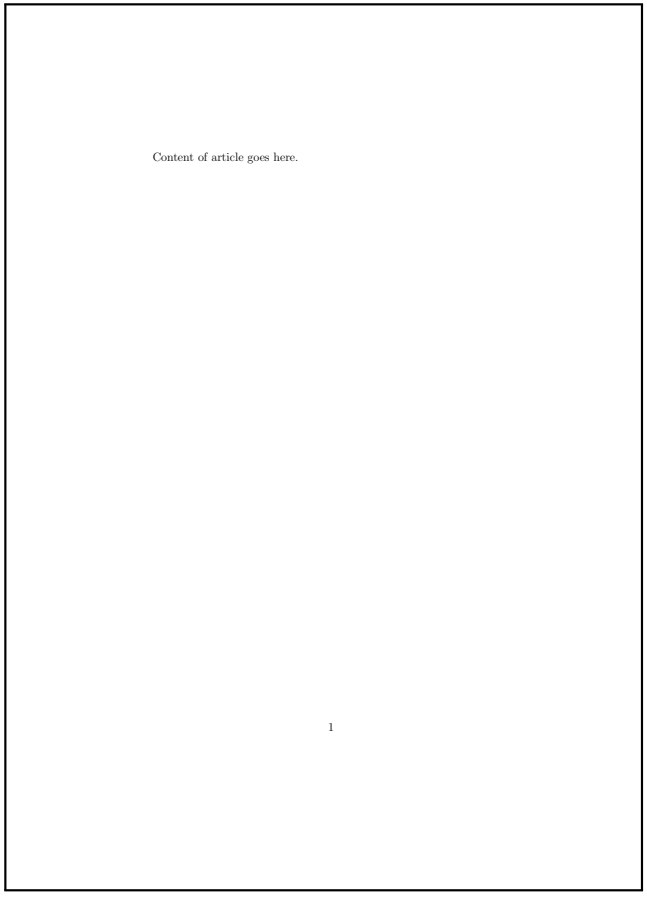
Structure of a LaTeX Document

```
% preamble
\documentclass[12pt]{article}

% document
\begin{document}

Content of article goes here.

\end{document}
```



Document Classes

The document classes available by default are `article`, `report`, `book` and `letter`.

A `report` can have chapters and the title page and abstract take a whole page each. Therefore `article` is most suited to journal articles/simple reports. This has options:

Option	article
Papersize	**a4paper**/letterpaper
Font size	**10pt**/11pt/12pt
Number of columns	**onecolumn**/two column
Margins	**oneside**/twoside
Title page	**notitlepage**
Orientation	**portrait**/landscape
Formula options	**center; right label**
Draft or final	**final**/draft

Custom Class

Journals sometime provide a custom class, defined in a `.cls` file, to set aspects of the layout.

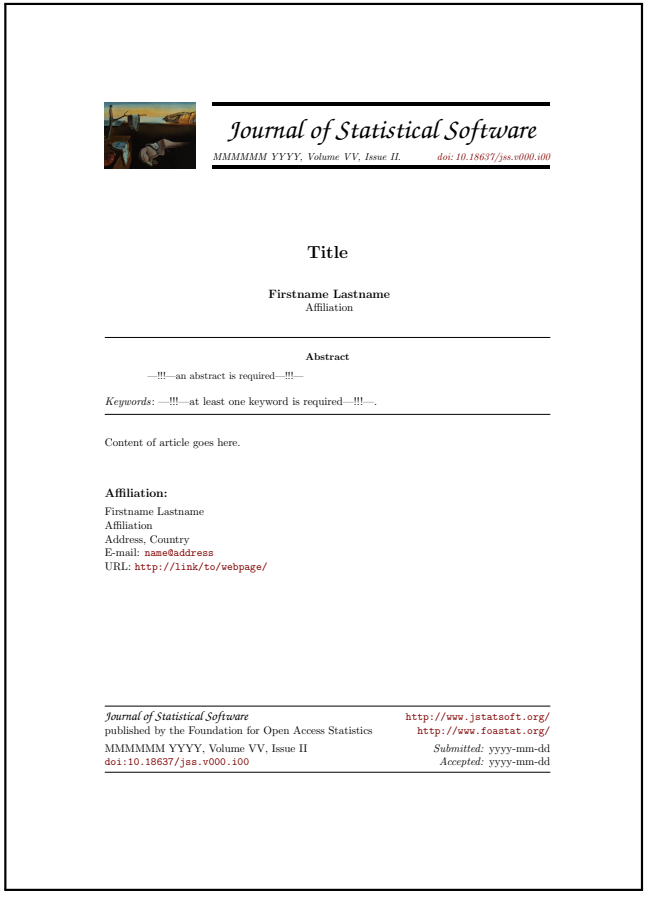
Put the `.cls` file in the same directory as the `.Rnw`/`.tex` file to use it. (Or look up how to add to your LaTeX installation.)

```
% preamble
\documentclass[a4paper]{jss}

% document
\begin{document}

Content of article goes here.

\end{document}
```



Journal of Statistical Software

MMMMMM YYYY, Volume VV, Issue II. doi: 10.18637/jss.v000.i00

Title

Firstname Lastname
Affiliation

Abstract

—!!!—an abstract is required—!!!—

Keywords: —!!!—at least one keyword is required—!!!—.

Content of article goes here.

Affiliation:
Firstname Lastname
Affiliation
Address, Country
E-mail: name@address
URL: http://link/to/webpage/

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
MMMMMM YYYY, Volume VV, Issue II *Submitted:* yyyy-mm-dd
doi:10.18637/jss.v000.i00 *Accepted:* yyyy-mm-dd

Preamble

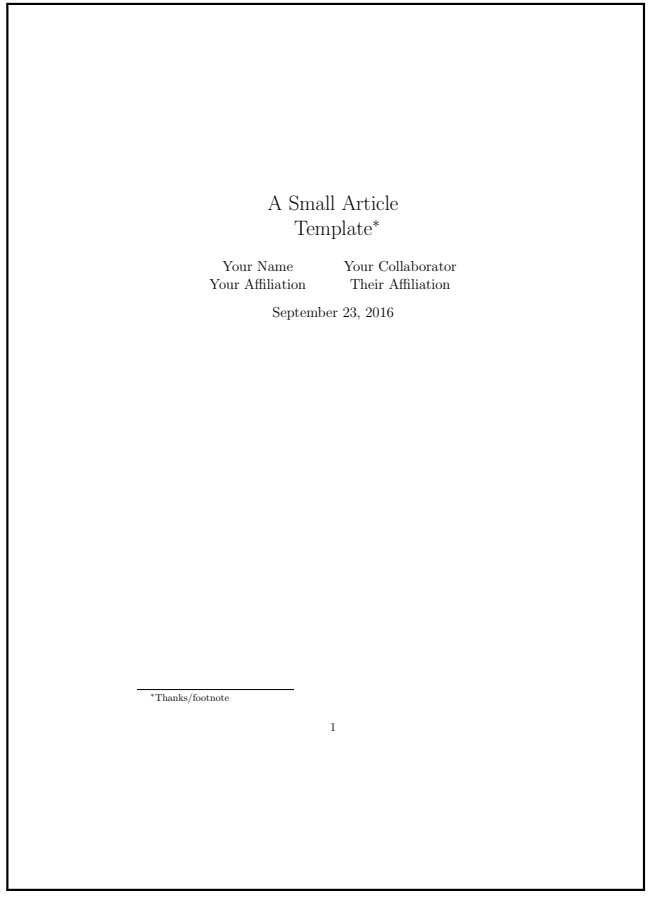
The preamble (before `\begin{document}`) is for all the set up code before we can start writing the document. Apart from setting the document class this includes

- defining parameters used by the document class, e.g. title, author
- loading addition packages and setting package options
- defining custom commands

Document Parameters

In the standard classes, the document parameters relate to the title section which is created in the body using `\maketitle`.

```
\documentclass[12pt]{article}
\title{A Small Article \\
Template\thanks{Thanks/footnote}}
\author{Your Name \\
Your Affiliation \\
\and
Your Collaborator \\
Their Affiliation \\
}
\date{\today}
\begin{document}
\maketitle
\end{document}
```



Add-on Packages

Like R, LaTeX has thousands of add-on packages. Important ones include

amsmath	for typsetting mathematical formulae
graphicx	include PNG or PDF graphics (needed for R graphics!)
hyperref	for defining hyperlinks within documents/for urls
booktabs	for publication quality tables
natbib	for including citations from a bibliography

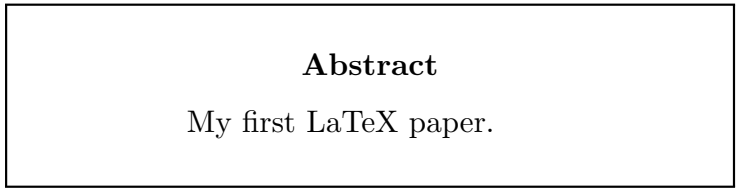
Package Example

```
\documentclass[12pt]{article}
\usepackage[margin = 0cm]{geometry}
\title{A Small Article \\
Template\thanks{Thanks/footnote}}
\author{Your Name \\
Your Affiliation \\
\and
Your Collaborator \\
Their Affiliation \\
}
\date{\today}
\begin{document}
\maketitle
\end{document}
```

Abstract

After beginning the document with `\maketitle`, the next part of the body will typically be the abstract. This can be added using the the `\abstract{}` command or the `abstract` *environment*:

```
\begin{abstract}
My first LaTeX paper.
\end{abstract}
```



Sectioning

The remainder of the paper will be organised by sections. The starred versions will be unnumbered.

```
\section{First Section}
The first section.
\subsection{A Subsection}
Text.
\subsection{Another Subsection}
Text.
\subsubsection{A Subsubsection}
Text.
\paragraph{A Labelled Paragraph}
Text.
\section{Second Section}

\subsection*{Unnumbered Subsection}
```

1 First Section

The first section.

1.1 A Subsection

Text.

1.2 Another Subsection

Text.

1.2.1 A Subsubsection

Text.

A Labelled Paragraph Text.

2 Second Section

Unnumbered Subsection

Text Markup

```
\textbf{bold} \\
\textit{italic} \\
\texttt{monospace}\\
Non-breaking space: A.~Author \\
"quote" ``double quotes'' \\
inter-word \\
page range, 1--10 \\
punctuation dash---like this \\
\href{http://r-project.org/}{
  R Project} \\
\url{http://r-project.org/}
```

bold
italic
`monospace`
Non-breaking space: A. Author
"quote" “double quotes”
inter-word
page range, 1–10
punctuation dash—like this
R Project
http://r-project.org/

Special Characters

Some characters are reserved for LaTeX commands:

$$\# \$ \% \_ \& \{ \}$$

They must be escaped to use show the character itself, e.g. `\#`.

The **underscore** package enables _ to be typed without escaping.

For extended character support, including

$$| < >$$

use `\usepackage[T1]{fontenc}`

For Unicode character support, e.g. for characters such as

á ä © €

use `\usepackage[utf8]{inputenc}`

Mark up for Mathematics

```
Inline math $(x + y)^2$.

Displayed math:
\[
\left( \frac{3 + x}{5} \right)
\]
$$ x_1, x_2, \ldots, x_n $$

Numbered equation:
\begin{equation}
\label{model}
y_i = \beta_0 + \beta_1 x_1 +
    \epsilon_i
\end{equation}
The model is given in Equation
\ref{model}.
```

Inline math $(x + y)^2$.
Displayed math:

$$\left(\frac{3 + x}{5} \right)$$

$$x_1, x_2, \ldots, x_n$$

Numbered equation:

$$y_i = \beta_0 + \beta_1 x_1 + \epsilon_i \qquad (1)$$

The model is given in Equation 1.

Lists

```
\begin{itemize}
\item A bullet point
\end{itemize}
\begin{enumerate}
\item The first item
\end{enumerate}
\begin{description}
\item[cats] a example of a mammal.
\end{description}
```

- A bullet point

1. The first item

cats a example of a mammal.

Including Graphics

```
\begin{figure}[tb]
\centering
\caption{\label{fig:boxplot} Example of a boxplot}
\includegraphics[height=3cm]{boxplot}
\end{figure}
Figure \ref{fig:boxplot} shows an example of a boxplot.
```

Figure 1: Example of a boxplot

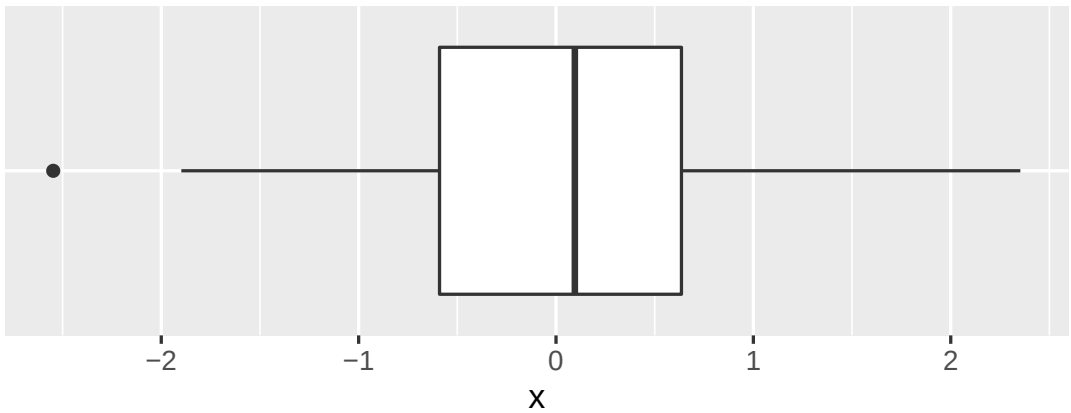


Figure 1 shows an example of a boxplot.

Floats

The figure environment is a floating environment. The figure will be floated to a suitable position in the document. The options give some control over placement

Option	Allowed placement
h	will be placed *approximately* where code appears
t	top of page
b	bottom of page
p	separate page for floats only
H	will be placed *exactly* where code appears, requires **float** package

Graphic Format

`includegraphics` can include PNG or PDF graphics - the PNG will be used if there are two files with the same extension.

PNG is a *raster* format, i.e. composed of coloured pixels. This generally a poor choice for publications as the graphics can look "fuzzy" and look worse when zoomed in.

PDF is a *vector* format, i.e. composed of coloured shapes. This is generally the best choice for statistical graphics. Exceptions include

- The graph has 1000s of points
- The graph has large blocks of colour, e.g. a heatmap

In these cases, the PDF file can be very large and the graph can take a long time to "load".

Saving Base Graphics as PDF in R

A base graphic can be saved as PDF in an R script as follows

```
pdf("plot.pdf", width = 5, height = 5)
plot(y ~ x)
dev.off()
```

The width and height are specified in inches.

Setting the width and height controls the aspect ratio. Increasing the width and/or height makes the text appear smaller as it has a fixed size.

Saving ggplot Graphics as PDF in R

A ggplot can be saved as PDF in an R script as follows

```
library(ggplot2)
p <- ggplot(data, aes(x = x, y = y)) + geom_point()
ggsave("plot.pdf", p, width = 10, height = 10,
       units = "cm")
```

The width and height are specified in inches by default, different units with the `units` argument.

If the width and height are unspecified, the dimensions of the plot window are used.

If `p` is unspecified, the last plot displayed in the plot window is saved.

Saving Base Graphics as PNG in R

A base graphic can be saved as PNG in an R script as follows

```
png("plot.png", width = 5, height = 5, units = "in",
    res = 600)
plot(y ~ x)
dev.off()
```

The width and height are specified in pixels by default, so `units` must be specfied to use in or cm.

Here the resolution is set to 600 ppi (pixels per inch) which is a high resolution for publication. 300 ppi is generally fine for self-printing. The default is 72 ppi, which is suitable for on-screen viewing.

Saving ggplot2 Graphics as PNG in R

A ggplot2 graphic can be saved as PNG in an R script as follows

```
ggsave("plot.png", p, width = 10, height = 10,
       units = "cm", dpi = 600)
```

The `dpi` argument is used to set the resolution.

The **knitr** Package

The **knitr** package is a general purpose package for dynamic reporting with R. Supported formats include the following:

- `.Rmd` to `.md`
- `.Rhtml` to `.html`
- `.Rtex` or `.Rnw` to `.tex`

It is conventional to use `.Rnw` "R no web" for R + LaTeX, as this was used by an earlier package, **Sweave**.

.Rnw Code Chunks

As in R markdown documents, R code can be included inline or as a code chunk.

Inline R expressions are written as `\Sexpr{1+1}`.

R chunks are written as

```
<<"label", echo = TRUE, eval=TRUE>>=
sample(1:9)
@
```

```
sample(1:9)

## [1] 7 9 2 5 3 1 4 6 8
```

Code Chunks producing Graphics

knitr provides several chunk options to control graphical output, including

Option	Setting
dev	e.g. `"pdf"`, `"png"`; `"pdf"` by default
dpi	e.g. `300`; 72 by default
fig.width, fig.height	e.g. `5`, the dimensions of the image file *always* in inches; 7 by default
out.width, out.height	e.g. `"5in"`, `"40%"`, the dimensions of the image in the document, line width by default
fig.align	e.g `"center"`, alignment; default none (left)
fig.cap	e.g. `"A caption."` a caption for the figure
fig.pos	e.g. `"H"` the position of the figure float
fig.show	`"asis"`, `"hold"`, `"hide"`; show as soon as produced, hold till end of chunk or hide.
fig.lp	e.g. `"fig:"`, prefix for the figure label, `"fig:"` by default

Boxplot Example

The following chunk saves the plot produced by the R code as a PDF file (by default in a `figures/` sub-directory), then creates a LaTeX chunk to include the graphic as in our earlier example.

```
<<"boxplot", fig.height = 2, fig.width = 5,
 ¬ out.height = "3cm", fig.cap = "Example of a boxplot",
 ¬ fig.align = "center", fig.pos = "tb">>=
ggHorizBoxplot(data.frame(x = rnorm(100)), ~x)
@
```

Adding `fig.cap` makes `knit` put the `\includegraphics` command in a figure environment, labelled as `fig:label`.

Side-by-side Plots

Multiple plots can be created in a chunk. If the width is set so there is space, plots will be placed side by side with `fig.show = "hold"`

```
<<"side-by-side", fig.height = 2, fig.width = 5,
 ¬ out.width = "50%", fig.cap = "Side-by-side boxplots",
 ¬ fig.show = "hold", echo = FALSE>>=
ggHorizBoxplot(data.frame(x = rnorm(100)), ~x)
ggHorizBoxplot(data.frame(x = rnorm(100)), ~x)
@
```

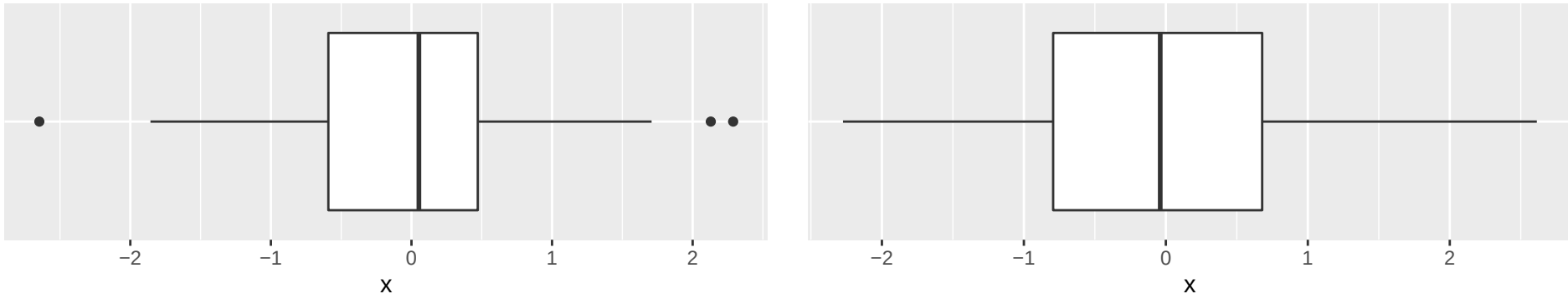


Figure 1: Side-by-side boxplots

Multiple Figures

Without `fig.show = "hold"`, i.e. with `fig.show = "asis"`, separate figures are produced

```
<<"multiple", fig.height = 2, fig.width = 5,
 ¬ out.width = "50%", fig.align = "center", fig.cap = c("Boxplot 1",
 ¬ "Boxplot 2"), echo = FALSE>>=
ggHorizBoxplot(data.frame(x = rnorm(100)), ~x)
ggHorizBoxplot(data.frame(x = rnorm(100)), ~x)
@
```

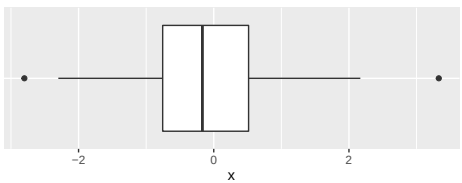


Figure 1: Boxplot 1

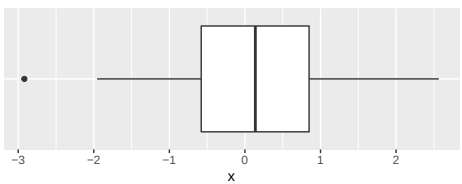
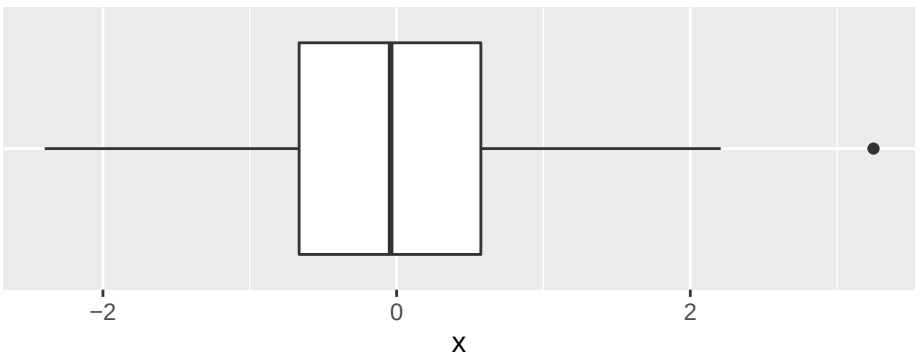


Figure 2: Boxplot 2

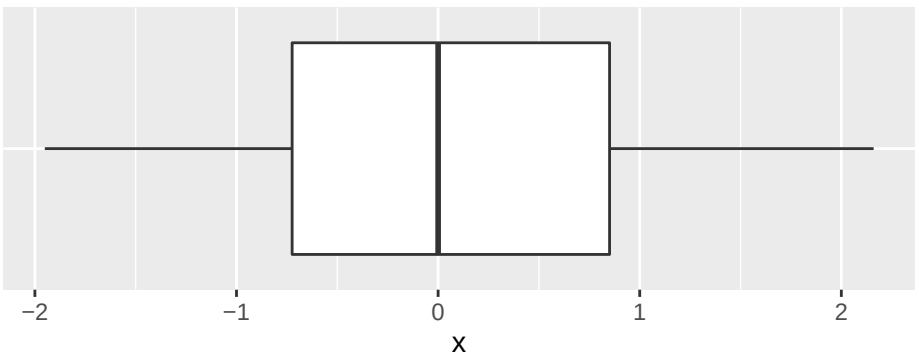
Subcaptions

Subcaptions can be added, this requires the **subfig** package

```
<<"subcaption", fig.height = 2, fig.width = 5,
 ¬ out.width = "50%", fig.align = "center", fig.cap = "Boxplots",
 ¬ fig.subcap = c("Boxplot 1", "Boxplot 2"), echo = FALSE>>=
ggHorizBoxplot(data.frame(x = rnorm(100)), ~x)
ggHorizBoxplot(data.frame(x = rnorm(100)), ~x)
@
```



(a) Boxplot 1



(b) Boxplot 2

Figure 1: Boxplots

Setting Default Chunk Options

Default chunk options can be set by including a chunk at the start of your document like the following

```
<<"setup", echo = FALSE>>=
library(knitr)
opts_chunk$set(fig.align = 'center', fig.show = 'hold',
               out.width = "5cm")
@
```

Colours in Statistical Graphics

Some useful references/packages

- Practical rules for using color in charts
- Color Brewer 2.0; palettes for sequential, diverging qualitative scales provided in **RColorBrewer** package. Palettes can be used in `ggplot`, e.g. `scale_colour_brewer(palette = "Set1")`.
- Escaping RGBland: Selecting Colors for Statistical Graphics. Perceptually-based colour scales implemented in **colorspace** - not limited to fixed number of levels in scale.
- **viridis** perceptually uniform, colour-blind and print-friendly colour scales for continuous data.
- **dichromat** colour-blind friendly scales.

Including Tables in LaTeX

```
\begin{table}[tb]
\centering
\caption{\label{tab:table}
        Example of a table}
\begin{tabular}{lc}
\toprule
Col 1 & Col 2 \\
\midrule
Text 1 & 1 \\
Text 2 & 2 \\
\bottomrule
\end{tabular}
\end{table}
Table \ref{tab:table} shows
an example of a table.
```

Table 1: Example of a table

Col 1	Col 2
Text 1	1
Text 2	2

Table 1 shows an example of a table.

Including Tables in Knitr

We can use `kable` in the `.Rnw` file, to generate LaTeX code for the table:

```
<<"kable">>=
library(knitr)
library(tibble) # use tibble to keep spaces in column names
dat <- data_frame(`Col 1` = paste("Text", 1:2), `Col 2` = 1:2)
kable(dat, "latex", align = "lc", booktabs = TRUE,
      caption = "\\label{tab:table} Example of a table")
@
```

Table : Example of a table

Col 1	Col 2
Text 1	1
Text 2	2

When the `.Rnw` file is converted to `.tex`, the chunk will be replaced by the LaTeX code, so the final PDF contains the desired table.

LaTex Code Generated by *kable*

```
\begin{table}

\caption{\label{tab:table} Example of a table}
\centering
\begin{tabular}[t]{lclc}
\toprule
Col 1 & Col 2\\
\midrule
Text 1 & 1\\
Text 2 & 2\\
\bottomrule
\end{tabular}
\end{table}
```

Table 1: Example of a table

Col 1	Col 2
Text 1	1
Text 2	2

Almost the same as before, but table won't float (no float options).

Including Floating Tables in Knitr

For a floating table, we can create an `xtable` object:

```
<<"xtable-code">>=
library(xtable)
tab <- xtable(dat, caption = "Example of a table",
              label = "tab:table", align = "llc")
@
```

`xtable` adds the rownames as a column by default so we align 3 columns.

Then we print the xtable in a chunk with the argument `results = "asis"`, so the generated LaTeX code gets printed "as is" in the `.tex` file.

```
<<"xtable-print", results = "asis">>=
print(tab, include.rownames = FALSE, table.placement = "tb",
      booktabs = TRUE, comment = FALSE)
@
```

The print method has > 30 (!) arguments allowing us to control exactly what gets printed in the LaTeX document.

xtable output

The LaTeX code generated by `xtable` is as follows

```
\begin{table}[tb]
\centering
\begin{tabular}{lc}
  \toprule
Col 1 & Col 2 \\
  \midrule
Text 1 &  1 \\
  Text 2 &  2 \\
   \bottomrule
\end{tabular}
\caption{Example of a table}
\label{tab:table}
\end{table}
```

Table 1: Example of a table

Col 1	Col 2
Text 1	1
Text 2	2

Now this is the same as our original LaTeX code (in a slightly different order).

More Complex Table

Here we add a custom header to the printed table

```
<<"xtable-more", results = "asis", eval = TRUE>>=
add2row <- list(pos = list(0, 0, 0),
                command = c("\\multicolumn{2}{c}{Columns} \\\\
                            "\\cmidrule{1-2}",
                            "Col 1 & Col 2 \\\\"))
print(tab, add.to.row = add2row, include.colnames = FALSE,
      include.rownames = FALSE,  table.placement = "tb",
      booktabs = TRUE, comment = FALSE)
@
```

More Complex Table Output

```
\begin{table}[tb]
\centering
\begin{tabular}{lc}
  \toprule
  \multicolumn{2}{c}{Columns} \\ \cmidrule{1-2} Col 1 & Col 2 \\ \m
Text 1 &  1 \\
  Text 2 &  2 \\
   \bottomrule
\end{tabular}
\caption{Example of a table}
\label{tab:table}
\end{table}
```

Columns	
Col 1	Col 2
Text 1	1
Text 2	2

Table 1: Example of a table

Tables

`kable` and `xtable` assume you have a data frame ready to output as a table. There are other R package that help you create the contents of the table, including

tables for computing and tabulating (multiple) summary statistics, cross-classified by different variables, with optional total rows/columns.

texreg for summarizing statistical models, e.g. tables of coefficients with confidence intervals and optional model summaries, e.g. R^2

Both of these packages can be used to produce HTML or LaTeX tables, look at the package vignettes for help and examples.

References

LaTeX can be used in combination with BibTeX to generating the bibilography or references section of your document. BibTeX will be included as part of your LaTeX installation.

You need to create a separate `.bib` file which is a plain text database with an entry for each reference. Such files can be created from many reference managers e.g. JabRef, Zotero, Mendeley, ... Alternatively BibTeX entries for individual papers can often be downloaded from journal websites.

Example BibTeX Entry

```
@Article{Zeil09,
title = {Escaping {RGB}land: Selecting Colors for Statistical Graphics},
author = {Achim Zeileis and Kurt Hornik and Paul Murrell},
journal = {Computational Statistics & Data Analysis},
year = {2009},
volume = {53},
pages = {3259--3270},
doi = {10.1016/j.csda.2008.11.033},
}
```

Using the Bibliography

Put the `.bib` file in the same directory as your `.tex` (or `.Rnw`) file.

In your preamble, add `\usepackage{natbib}`.

Then references are cited in the body of the document using their key, e.g.

- `\citet{Zeil09}`, produces Zeileis et al. (2009).
- `\citep{Zeil09}`, produces (Zeileis et al., 2009).
- `\citep[See][p.~22]{Zeil09}`, produces (See Zeileis et al., 2009, p. 22).

Creating the References Section

Finally at the end of your document, add the following lines

```
\bibliographystyle{chicago}
\bibliography{paper}
```

References

Zeileis, A., K. Hornik, and P. Murrell (2009). Escaping RGBland: Selecting colors for statistical graphics. *Computational Statistics & Data Analysis 53*, 3259–3270.

where `paper.bib` is the name of your `.bib` file.

Sometimes journals will require a custom bibliography style defined in a `.bst` file. Put that in the same directory as the `.tex` file and use the file name instead of `chicago`.

Alternatively, look up where to install `.bst` files for use across different documents.

Back to R Markdown

When compiling `.md` to PDF, we can use some of the extra features of LaTeX by cutomising the YAML header to modify the preamble of the intermediate `.tex` document, e.g.

```
---
title: "My Title"
output:
  pdf_document:
    citation_package: "natbib"
documentclass: "article"
fontsize: 11pt
geometry: margin=1in
bibliography: paper.bib
---

Testing natbib \citep{Zeil09}.

# References
```

My Title

Testing natbib [Zeileis et al., 2009].

References

Achim Zeileis, Kurt Hornik, and Paul Murrell. Escaping RGBland: Selecting colors for statistical graphics. *Computational Statistics & Data Analysis*, 53:3259–3270, 2009. doi: 10.1016/j.csda.2008.11.033.

Getting Styles Right

Although we can change the document class in the YAML header, a custom `.cls` file may not work, e.g. `jss.cls`. In this case a template `.tex` file, or a customised output format is required.

The **rticles** package provides special R markdown templates for

- JSS articles
- R Journal articles
- CTeX documents
- ACM articles
- ACS articles
- Elsevier journal submissions
- AEA journal submissions

These templates can be accessed in the dialog found via File > New File > R Markdown...From Template in RStudio.

See R Markdown help for custom bib styles.