Notes on Reproducible Research with knitr

General knitr Chunk Options

How each code chunk is handled during `knit`-ing depends on *chunk options* that control the behavior and appearance of each chunk. The main chunk options are:

- `eval`: Whether the chunk should be evaluated
- `echo`: Whether the code contained in the chunk should be displayed in the PDF
- `results`:
  - "markup" (default): Results are displayed like in the R console
  - "asis": Used for any chunk containing LaTeX output (e.g., tables)
  - `hold`: All results are held until the end of the chunk (versus displayed line-by-line)
  - "hide": Results are not displayed
- `warning` and `message`: Whether to suppress warnings and messages
- `tidy` and `highlight`: Whether R code should be tidied and syntax highlighted, respectively

Chunk options can be set in each chunk:

```
<<a, eval = TRUE, echo = FALSE>>=
2 + 2
@
```

Or, they can be set globally inside a "setup" chunk that applies options to all other chunks:

```
<<a, eval = TRUE, echo = FALSE>>=
opts_chunk$set(echo = FALSE, cache = TRUE, message = FALSE)
@
```

General knitr Package Options

knitr also allows you to specify some *package options* to control the behavior of knitr in general. These are probably not helpful to beginners, but may be useful for advanced users. You can find a list of these options at http://yihui.name/knitr/options#package_options.

knitr Plotting

There are two ways to include plots in a knitr workflow. One involves using manual includes of plots that are generated in a chunk, saved locally, and then included using `\includegraphics{}`, for example:

```
<<>>=
pdf('figures/barplot.pdf')
barplot(mtcars$cyl)
dev.off()
@

\begin{figure}
\caption{A barplot}
\label{fig:barplot}
\includegraphics{figures/barplot}
\end{figure}
```

The alternative is to specify figure-related options and let knitr handle the details:

```
<<fig.cap='A barplot', fig.lp='fig:barplot'>>=
barplot(mtcars$cyl)
@
```

The figure-relevant options useful for controlling the behavior of a knitr plot chunk are:

- `fig.path`: The path to store and retrieve figures from (default is `./figure`)
- `fig.show`
  - `asis`: All plots are rendered line-by-line
  - `hold`: All plots are held until the end of the chunk
  - `animate`: Plot(s) converted into an animation (doesn't work in all PDF viewers)
  - `hide`: Plot(s) not shown
- `fig.height` and `fig.width`
- LaTeX-related options
  - `fig.env`: LaTeXfigure environment to use
  - `fig.cap`: LaTeXcaption
  - `fig.lp`: LaTeXlabel (for cross-referencing)
- `dev`
  - `dev.args`: A list containing arguments passed to the graphics device function

LaTeX Tables

Tables are created in LaTeX using the `tabular` environment:

```
\begin{tabular}{l r r} \toprule
Header 1 & Header 2 & Header 3\\
\midrule
Row 1 \\
Row 2 \\ \bottomrule
\end{tabular}
```

Header 1	Header 2	Header 3
Row 1		
Row 2		

By default, a `tabular` environment will appear in the document exactly where in occurs in the LaTeX source. In most academic writing, we'll additionally want to include this `tabular` environment inside a `table` environment, which is a *float* environment that will cause the table to be displayed at the top or bottom of an appropriate page. This matters because we can only cross-reference tables if they are in a float environment and contain a `label`. For example:

```
\begin{table}
\caption{This is the table's title}
\label{tab:simple}
\begin{tabular}{l r r} \toprule
Header 1 & Header 2 & Header 3\\
\midrule
Row 1 \\
Row 2 \\ \bottomrule
\end{tabular}
\end{table}
```

Using that, we can then refer to the table anywhere in our document using `Table \ref{tab:simple}` and it will be replaced by "Table 1" (and the number will be automatically updated depending on the table's location in the document).

Creating Tables

You can mock-up your own LaTeX tables by-hand, but it is much easier to use R packages to produce that LaTeX code automatically. This applies even if you don't plan to use knitr in your workflow. You can use R package to write LaTeX output to a local file (e.g., `table.tex`) and then include that table in your article using: `\input{table.tex}`, wherever you want the table to appear in your document.

The following sections detail how to create tables using three popular packages. The Reproducible Research TaskView on CRAN lists functionality found in other packages for creating LaTeX-formatted output.

`kable`: Simple tables

`kable` from the knitr package is an easy way to create simple tables without a float environment. For example, we can use `kable` to print a simple correlation matrix with something like:
`kable(cor(mtcars), "latex")`
`kable` includes a few options to control appearance:

- `format`: This will always be "latex" for our purposes
- `digits`: How many digits to print in a numeric table
- `row.names`, `col.names`: Whether to include row or column names for a table
- `align`: A vector of alignments for the table columns

`xtable`: Generic table creation

`xtable` from the `xtable` package is a much more powerful (and thus complicated) function for create simple tables (with or without float environments). For example we can use it to create one- or more-way tabulations using, e.g., `xtable(table(mtcars$cyl))` or `xtable(with(mtcars, table(cyl, gear)))`. There are a number of options that can be specified for `xtable`:

- `caption`: A LaTeX caption
- `label`: A LaTeX label for cross-referencing
- `align`: A vector of column alignments
- `digits`: How many digits to print in a numeric table

`xtable` is also a little confusing because it includes options for its `print` method that further control the appearance of tables, such as the float environment for the table. These options are called within `print`, but outside of `xtable`, e.g.: `print(xtable(table(mtcars$cyl)), floating = FALSE)` produces a table with no float environment.

- `file`: A file path to write the table to
- `floating`: Whether to include the table in a float environment
- `floating.environment`: What float environment to use (`table`, by default)
- `hline.after`: Where to place horizontal lines in the table
- `NA.string`: How missing values (`NA`'s) should be displayed
- `include.rownames`, `include.colnames`: Whether to include row and column names
- `sanitize.text.function`: A function to handle LaTeX-incompatible text strings
- `booktabs`: Whether to use the LaTeX `booktabs` package for more attractive tables

`stargazer`: Tables for model objects

`stargazer` from the stargazer package is useful for creating tables for model objects (e.g., regression results). While `xtable` can also do this, `stargazer` does it more elegantly by aligning variables from multiple models and better controlling printing of supplemental information (e.g., goodness-of-fit statistics). This produces a simple two-model output:

```
stargazer(lm(mpg ~ cyl, data = mtcars), lm(mpg ~ cyl + hp, data = mtcars))
```

Output can be controlled using many options, some of which are shown below:

- `title`: A LaTeX caption
- `label`: A LaTeX label for cross-referencing
- `out`: A file path to print the table to
- `column.labels`: Labels for the model columns
- `covariate.labels`: Labels to replace covariate variable names. (Note: This requires some care.)
- `digits`: How many digits to print in the table
- `float`: Whether to include the table in a float environment
- `float.env`: What float environment to use (`table`, by default)
- `model.names`: Whether to print type of model for each column
- `model.numbers`: Whether to number the models
- `dep.var.caption`: The caption for the dependent variables
- `dep.var.labels.include`: Whether to include dependent variable labels
- `multi.column`: Whether to span column headers across like columns
- `keep.stat`: A vector of named statistics to include in the table:
  "all", all statistics; "adj.rsq", adjusted R-squared; "aic", Akaike Information Criterion; "bic", Bayesian Information Criterion; "chi2", chi-squared; "f", F statistic; "ll", log-likelihood; "logrank", score (logrank) test; "lr", likelihood ratio (LR) test; "max.rsq", maximum R-squared; "n", number of observations; "null.dev", null deviance; "Mills", Inverse Mills Ratio; "res.dev", residual deviance; "rho", rho; "rsq", R-squared; "scale", scale; "theta", theta; "ser", standard error of the regression (i.e., residual standard error); "sigma2", sigma squared; "ubre", Un-Biased Risk Estimator; "wald", Wald test.
- `style`: A character string naming a journal style. Includes some common social science journal formats.

Working with knitr

The best way to learn knitr is to use it. I would recommend you try one of the following activities:

1. Migrate one of your existing projects to knitr, either:
   - Move R code into knitr code chunks
   - Move all code into one chunk and configure automated LaTeX includes for the results.
2. Setup the knitr template for a new project you haven't started
3. Try creating a knitr document for a toy analysis project

Ideas for Toy Analysis Project

Use one of R's built-in datasets (just type `data()` in the RStudio console) to conduct a simple analysis in a knitr workflow.

- Possible datasets might include: `iris`, `mtcars`, `ChickWeight`, and `infert`
- Run descriptive analyses that can be summarized in tables and included using `kable` or `xtable`
- Run basic plots (`hist`, `barplot`, `plot`) and include those using knitr plot options
- Run some regression models (e.g., `lm`, `glm`, etc.) and include the results using `stargazer`
- Reference tables and figures in the text using LaTeX cross-references (i.e., `\ref{}`)
- Reference particular analytic results using in-line code (i.e., `\Sexpr{}`)

Here are some basic code examples to get you started using these datasets:

```
# basic summary statistics
head(iris) # some of the raw data
summary(iris) # variable descriptive statistics

# plots
plot(infert[,1:4]) # multivariate correlation plot
hist(infert$age) # histogram of age
plot(density(iris$Sepal.Length)) # density plot
with(iris, plot(Sepal.Length, Sepal.Width)) # scatterplot

# tables
cor(iris[,1:4])
table(infert$education)
with(infert, table(age, education))
with(infert, table(education, induced))

# models
aov(Sepal.Length ~ Species, data = iris) # ANOVA (use ‘xtable‘)
lm(Sepal.Length ~ Sepal.Width * Species, data = iris) # OLS
glm(case ~ education + age + induced + spontaneous, data = infert) # Logit
```