# The **pgfSweave** Package

Cameron Bracken and Charlie Sharpsteen

September 21, 2009

The **pgfSweave** is about speed and style. For speed, the package provides capabilities for "caching" graphics generated with **Sweave** on top of the caching funcitonality of **cacheSweave**[1]. For style The **pgfSweave** package also facilitates the use of the **tikzDevice**[2] package and **eps2pgf**[3] utility. Using these tools figure labels are converted to LaTeX strings so not only do they match the style of the document but LaTeX math symbols/equations can be put in labels.

The backbone og **pgfSweave** is a a new driver for **Sweave** (`pgfSweaveDriver`). The driver provides new chunk options `tikz`, `pgf` and `external` on top of the `cache` option provided by **cacheSweave**. This package started as a fork of **cacheSweave**. This document highlights the features and usage of **pgfSweave**. This document assumes familiarity with **Sweave**.

We'll start first with an example to entice you. The following code chunk used with **pgfSweave** will produce the graphic below.

## 1 Motivation and Background

**Sweave** is a tool for generating "reproducible" documents by embedding R or S "code chunks" directly into a LaTeX document. The problem of preforming lengthy computations in Sweave documents is not a new one. Previous attempts to tackle this problem include **cacheSweave** and the **weaver**[4]. Specifically these packages address the problem that code chunks with lengthy computations are executed every time a document is compiled. Both packages provide a `cache` option which saves R objects for quick access during successive compilations. The **cacheSweave** package stores results in a **filehash**[5] databases while the **weaver** package stores RData files. The benefit of the **cacheSweave** method is lazy loading of objects. Both methods provides significant speedup of most computations, namely only those which create objects in the global environment.

The existing methods have some drabacks:

1. Plots are not cached (since plots does not create objects in the global environment). If a plot takes a long time to generate, the same problem exists as when lengthy computations are present. Idealy we would like to reuse a plot if the code that generated it has not changed.
2. Consistency in style (font, point size) in automatically generated graphics is difficult to achieve. The default font and point size in R does not match LaTeX very well and getting this to match precisely is tricky business. The previously mentioned tools, **tikzDevice** and **eps2pgf**, counter this but using them with **Sweave** can be cumbersome.

The so called "caching" of plots is achieved with the help of three tools: the TeX package **PGF**[6] and either the command line utility **eps2pgf** or the R package **tikzDevice**. When we refer to the "caching" of a graphic we mean that if the code chunk which generated the graphic is unchanged, an image included from a file rather than regenerated from the code. The TeX package **pgf** provides the ability to "externalize graphics." The effect of externalization is that graphics get extracted an compiled separately, saving the time of compilation. The externalization chapter in the **PGF/Ti*k*Z** manual is extremely well written,

[1] http://cran.r-project.org/web/packages/cacheSweave/index.html
[2] http://cran.r-project.org/web/packages/tikzDevice/index.html
[3] http://sourceforge.net/projects/eps2pgf/
[4] http://www.bioconductor.org/packages/2.3/bioc/html/weaver.html
[5] http://cran.r-project.org/package=filehash
[6] http://sourceforge.net/projects/pgf/

so please look there for more information. Exteralization plus some clever checking on the part of **pgfSweave** makes up the chaching mechanism.

# 2 System Requirements

In general **pgfSweave** depends on:

1. A working TEXdistribution (such as TeXLive for linux and mac and MiKTex for Windows)
2. The java command line interpreter (i.e. the `java` command). This is standard on most systems and is free to download otherwise.
3. At least version 2.00 of the **PGF/Ti*k*Z** package for LATEX.

That should be it for any *nix or Mac OS X system.

## 2.1 Windows specific requirements

The **pgfSweave** package can work on windows with some special care. First of all it is strongly recommendeed that you install R in in a location that does not have spaces in its path name such as `C:\R`. This will save you much grief when using **Sweave**. Other wise do the following in the order listed.

1. Install Java.
2. Install MiKTEX.
3. Upgrade to or install PGF 2.0 if not already done.
4. Install Rtools[7]. Make sure to allow the Rtools installer to modify your PATH.

If everything is set up correctly, the commands `java` and `pdflatex` or `latex` should be available at the command prompt.

# 3 Usage

We assume a familiarity with the usage of **Sweave**, for more information see the **Sweave** manual[8]. This section will explain the usage of the `tikz`, `pgf` and `external` options and then provide a complete example.

## 3.1 The `tikz` option

The first new code chunk option, `tikz`, acts the same as the `pdf` or `eps` options but instead of resulting in an `\includegraphics{}` statement the result is an `\input{}` statement. Consider the following code:

[7] http://www.murdoch-sutherland.com/Rtools/

[8] http://www.stat.uni-muenchen.de/~leisch/Sweave/Sweave-manual.pdf

Input:

```
\begin{figure}[ht]
<<tikz-option,fig=T,tikz=T,echo=F>>=
    x <- rnorm(100)
    plot(x)
@
\caption{caption}
\label{fig:tikz-option}
\end{figure}
```

Output:

```
\begin{figure}[ht]
\input{tikz-option.tikz}
\caption{caption}
\label{fig:tikz-option}
\end{figure}
```

The `.tikz` file is generated with the **tikzDevice** package. This is the default graphics output for **pgfSweave**, the `tikz` option is set to `TRUE` by default.

## 3.2 The `pgf` option

The second new code chunk option `pgf`, acts the same as the tikz option in that the result is an `\input{}` statement. Consider the following code:

Input:

```
\begin{figure}[ht]
<<pgf-option,fig=T,pgf=T,tikz=F,echo=F>>=
    x <- rnorm(100)
    plot(x)
@
\caption{caption}
\label{fig:pgf-option}
\end{figure}
```

Output:

```
\begin{figure}[ht]
\input{pgf-option.pgf}
\caption{caption}
\label{fig:pgf-option}
\end{figure}
```

The `.pgf` file is generated with the **eps2pgf** utility. The `postscript` graphics device is used first to generate a `.eps` file. Then the command

```
$ java -jar /path/to/eps2pgf.jar -m directcopy graphic.eps
```

is run on every code chunk that has `fig=TRUE` and `pgf=TRUE`. We do not reccommend using this option in favor of the `tikz` option. Using the `tikz` option only involves one creation step (generated natively from R) and it honors the R text styles.

## 3.3 The `external` option

Input:

```
\begin{figure}[ht]
<<external,fig=T,tikz=T,external=T,echo=F>>=
    x <- rnorm(100)
    plot(x)
@
\caption{caption}
\label{fig:external-option}
\end{figure}
```

Output:

```
\begin{figure}[ht]
\beginpgfgraphicnamed{external}
\input{external.tikz}
\endpgfgraphicnamed
\caption{caption}
\label{fig:external}
\end{figure}
```

### 3.4 A complete example

At this point we will provide a complete example. The example from the **Sweave** manual is used to highlight the differences. The two frame below show the input Sweave file `example.Rnw` and the resulting tex file `example.tex`.

pgfSweave-example-Rnw.in

```
\documentclass{article}

\usepackage{tikz}
\usepackage[margin=1in]{geometry}
\pgfrealjobname{pgfSweave-example}
\title{Minimal pgfSweave Example}
\author{Cameron Bracken}

\begin{document}


<<setup,echo=F>>=
setCacheDir("cache")
@
\maketitle
This example is identical to that in the Sweave manual and is intended to
introduce pgfSweave and highlight the basic differences.  Please refer to
the pgfSweave vignette for more usage instructions.

We embed parts of the examples from the \texttt{kruskal.test} help page
into a \LaTeX{} document:

<<data,cache=T>>=
data(airquality)
kruskal.test(Ozone ~ Month, data = airquality)
@
which shows that the location parameter of the Ozone distribution varies
significantly from month to month. Finally we include a boxplot of the data:


\setkeys{Gin}{width=4in}
\begin{figure}[!ht]
\centering
%notice the new options
<<boxplot,echo=F,fig=T,tikz=T,external=T,width=4,height=4>>=
        boxplot(Ozone ~ Month, data = airquality,main='Ozone distribution',
                xlab='Month',ylab='Concentration')

@
\caption{This is from pgfSweave. Text is typset by \LaTeX\ and so matches the font of the document.}
\end{figure}


\end{document}
```

pgfSweave-example-Rnw.in

On the input file run:

```
R> library(pgfSweave)
R> pgfSweave('example.Rnw',pdf=T)
```

or

```
$ R CMD pgfsweave example.Rnw
```

And we get:

pgfSweave-example-tex.in

```
\documentclass{article}

\usepackage{tikz}
\usepackage[margin=1in]{geometry}
\pgfrealjobname{pgfSweave-example}
\title{Minimal pgfSweave Example}
\author{Cameron Bracken}

\usepackage{/Library/Frameworks/R.framework/Resources/share/texmf/Sweave}
\begin{document}


\maketitle
This example is identical to that in the Sweave manual and is intended to
introduce pgfSweave and highlight the basic differences.  Please refer to
the pgfSweave vignette for more usage instructions.

We embed parts of the examples from the \texttt{kruskal.test} help page
into a \LaTeX{} document:

\begin{Schunk}
\begin{Sinput}
> data(airquality)
> kruskal.test(Ozone ~ Month, data = airquality)
\end{Sinput}
\end{Schunk}
which shows that the location parameter of the Ozone distribution varies
significantly from month to month. Finally we include a boxplot of the data:


\setkeys{Gin}{width=4in}
\begin{figure}[!ht]
\centering
%notice the new options
\beginpgfgraphicnamed{pgfSweave-example-boxplot}
\input{pgfSweave-example-boxplot.tikz}
\endpgfgraphicnamed
\caption{This is from pgfSweave. Text is typset by \LaTeX\ and so matches the font of the document.}
\end{figure}


\end{document}
```

pgfSweave-example-tex.in

# Minimal pgfSweave Example

Cameron Bracken

September 21, 2009

This example is identical to that in the Sweave manual and is intended to introduce pgfSweave and highlight the basic differences. Please refer to the pgfSweave vignette for more usage instructions.

We embed parts of the examples from the `kruskal.test` help page into a LaTeX document:

```
> data(airquality)
> kruskal.test(Ozone ~ Month, data = airquality)
```

which shows that the location parameter of the Ozone distribution varies significantly from month to month. Finally we include a boxplot of the data:

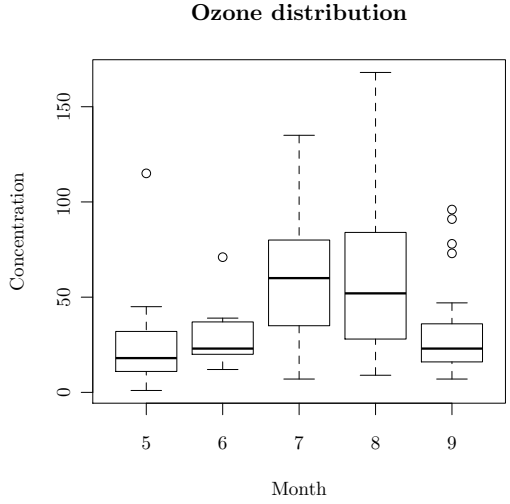


Figure 1: This is from pgfSweave. Text is typset by LaTeX and so matches the font of the document.

# 4 The Process

The process that **pgfSweave** uses when caching is turned on is outlined in the flow chart below:

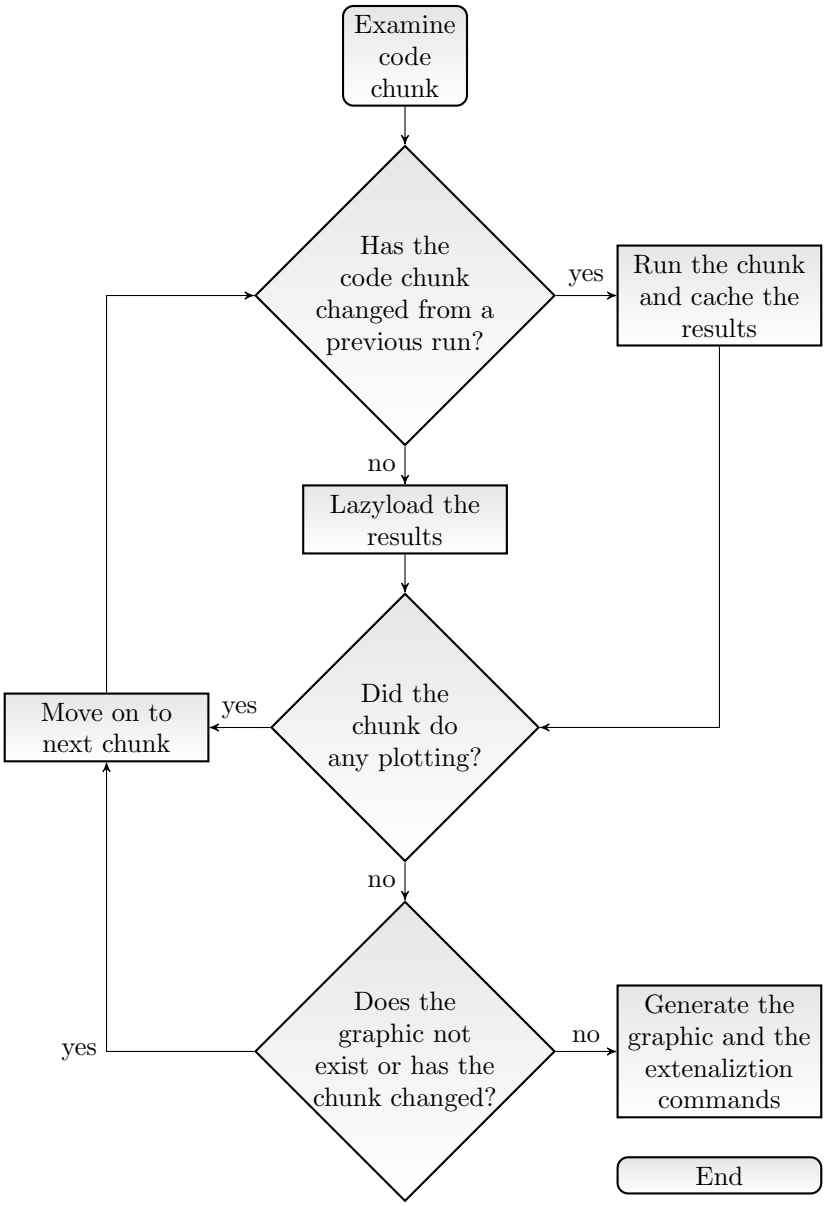


Figure 1: Flow chart of modeling procedure.

# 5 Consistency in style between graphics and text

In Figure **??** Notice the inconsistency in font and size between the default R output and the default LaTeX output. Fonts and font sizes can be changed from R but it is hard to be precise. What if you decide to change the font and and point size of your entire document? In figure **??** the text is consistent with the rest of the document.

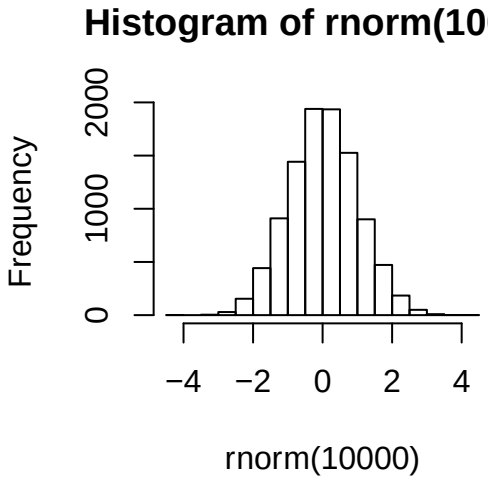


Figure 2: This is normal **Sweave**.

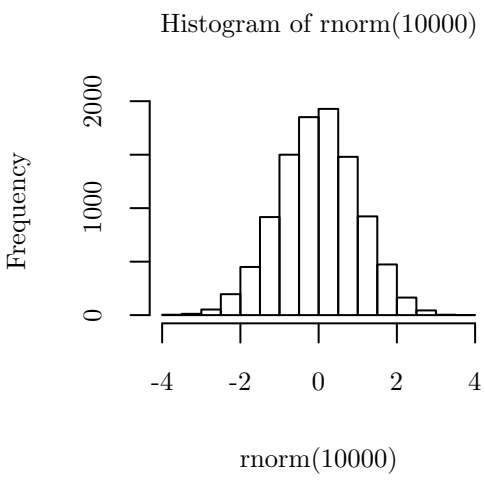


Figure 3: This is from **pgfSweave**.

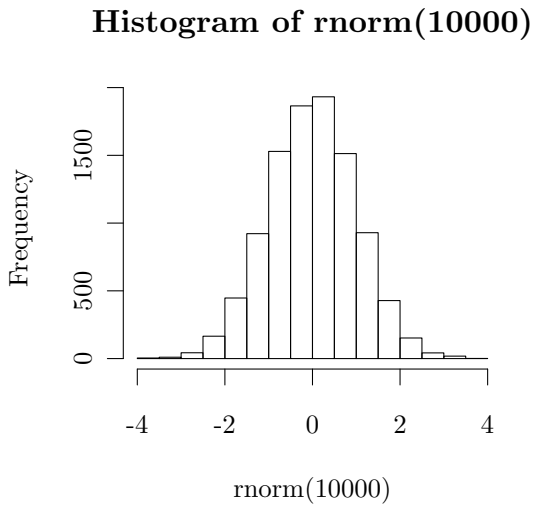


Figure 4: This is from **pgfSweave**.

The example below illustrates some of the power of **pgfSweave**. LaTeX code can be directly input into captions. This sort of thing is already available in R but again consistency in font and text size is difficult to achieve.

Every other part of **Sweave** and **cacheSweave** work the same.

# 6 Sweave graphic width defaults

The default in `Sweave.sty` is to fix the width of every image to 80% of the text width by using `\setkeys{Gin}{width=.8\textwidth}`. Say you have a 7 in text width and code chunk where you set `width=4`. The original 4 inch wide graphic will have text size matching your document but when it is included in your document it will be scaled up to 7 inched wide and the text will get bigger! This default is quite contrary to the philosophy of **pgfSweave**. There are two ways around this before each code chunk you can set `\setkeys{Gin}{width=<graphic width>}`. Alternatively (and the recommended way) you can turn off this feature globally by using `\usepackage[nogin]{Sweave}`.

# 7 Frequently Asked Questions

## Can pgfSweave be run from the command line?

Sure! Use the shell script provided in the **pgfSweave** package source in the `exec/` directory Save the script somewhere in your `PATH` and run

```
$ pgfsweave <yourfile>.Rnw
```

## How do I set subdirectories for figures and caches?

This is strait out of the **Sweave** and **cacheSweave** manuals (nothing new here). For a figures subdirectory [9] use the `prefix.string` option:

```
\SweaveOpts{prefix.string=figs/fig}
```

For a caching subdirectory use a code chunk at the beginning or your document like:

```
<<setup,echo=F>>=
setCacheDir("cache")
@
```

## Why are the width and height options being ignored?

This is another one from **Sweave**. You must use the `nogin` option in `Sweave.sty` for the width and height parameters to actually affect the size of the image in the document:

```
\usepackage[nogin]{Sweave}
```

## latex/pdflatex is not found in R.app (Mac OS X) and [Possibly] R.exe (Windows)

Your latex program is not in the default search path. Put a line such as:

```
Sys.setenv("PATH" = paste(Sys.getenv("PATH"),"/usr/texbin",sep=":"))
```

in your `.Rprofile` file.

## I get a bunch of "Incompatible list can't be unboxed" errors when compiling.

This is a problem with PGF. The workaround is to load the **atbegshi** package before PGF or TikZ:

```
\usepackage{atbegshi}
\usepackage{pgf}
```

or

```
\usepackage{atbegshi}
\usepackage{tikz}
```

[9] make sure to create the directory first!

### The vignette in /inst/doc/ does not contain any code chunks!

That is because the vignette in `/inst/doc/` is a "fake" vignette generated from the "real" vignette in `/inst/misc/vignette-src/`. The reason for this extra step is that package vignettes must be able to be compiled with `R CMD Sweave <vignette>.Rnw`, which is precisely what we don't want to use!

To compile this vignette yourself use the following:

```
$ svn checkout http://svn.rforge.net/pgfSweave/trunk pgfSweave
$ R CMD INSTALL pgfSweave
$ cd pgfSweave/inst/misc/vignette-src/
$ make
```