Refine the Beamer Template

Sweavel and Beamer Customizations

Paul Johnson[1] [2]

[1]Department of Political Science
University of Kansas

[2]Center for Research Methods and Data Analysis
University of Kansas

Conference on LyX Presentations, 2017

Outline

1. Document Preamble Changes

2. Top of Document Boilerplate

3. Final Idioms
   - Figure Customization
   - Caching Results and Calculations
   - Changes for You

4. Conclusions

Outline

1 Document Preamble Changes

2 Top of Document Boilerplate

3 Final Idioms
- Figure Customization
- Caching Results and Calculations
- Changes for You

4 Conclusions

A Boilerplate for the preamble.

1. Instead of using the R-provided Sweave.sty file to structure the output, we use Frank Harrell's Sweavel.sty (after a few changes to suit my taste).
    - Based on the LaTeX Listings package. `http://biostat.mc.vanderbilt.edu/wiki/Main/SweaveTemplate`
        - shaded backgrounds
        - line-breaking for long lines
        - more beautiful symbols (with customization)
2. Several Beamer customizations to make the output more pleasant (in my eye).
    - theme changes
    - alter the continuation
    - insert slide numbers

Preamble Changes to work with Sweavel.sty

```
\usepackage{Sweavel}
\fvset{listparameters={\setlength{\topsep}{0em}}}
\def\Sweavesize{\normalsize}
%% or
%% \def\Sweavesize{\scriptsize}
\def\Rcolor{\color{black}}
\def\Rbackground{\color[gray]{0.95}}

\usepackage{graphicx}
\usepackage{listings}
\lstset{tabsize=2, breaklines=true, style=Rstyle}
```

Preamble Changes to improve Beamer

```
\usetheme{Antibes} %% Value judgement

%% allowframebreaks beautification:
%% Change Roman I II III to dot-dot-dot
\mode<presentation>
\setbeamertemplate{frametitle continuation}[from second]
\renewcommand\insertcontinuationtext{...}

%% I want slide numbers displayed
\expandafter\def\expandafter\insertshorttitle\expandafter{%
 \insertshorttitle\hfill\insertframenumber\,/\,\
     inserttotalframenumber}
```

You need the complete LaTeX Setup, of course

- If you don't have the listings package for LaTeX, this will be an epic fail
- You need the Sweavel.sty file.
  - in your current working directory for short-term use
  - copy it into your LaTeX setup (generally under "tex/latex" will suffice) and run reconfigure (texhash).

Outline

1. Document Preamble Changes
2. Top of Document Boilerplate
3. Final Idioms
    - Figure Customization
    - Caching Results and Calculations
    - Changes for You
4. Conclusions

Changes for the beginning of the document

- Make sure an output directory exists from R’s current working directory. Tell R to not use “line breaks”, we will let Sweavel handel it.

```
> dir.create("plots", showWarnings=F)
> options(width=160, prompt=" ", continue="
      ")
```

- This is a LaTeX statement, which R and the Sweaving process will take into account. I have this at the front of the document

Changes for the beginning of the document ...

- Why that's important:
  1. all R elements are written to a string prefixed “plots/t”, folder=”plots”, common prefix letter “t”
  2. ae=F prevents R and Sweave from putting in a bunch of workarounds into the LaTeX code.
  3. default height and width of figures can be adjusted for each particular figure.
- The first R code chunk sets the defaults that cooperate with Sweavel and Listings

Changes for the beginning of the document ...

```
## Other settings I like
options(device = pdf)
options(useFancyQuotes = FALSE)
op <- par()
pjmar <- c(5.1, 5.1, 1.5, 2.1)
options(SweaveHooks=list(fig=function() par(
    mar=pjmar, ps=12)))
pdf.options(onefile=F,family="Times",
    pointsize=12)
```

Run a block of R code again

Results are interspersed with input.

```
set.seed(234234)
x1 <- rnorm(1000)
mean(x1)
```

```
[1] 0.01746376
```

```
sd(x1)
```

```
[1] 1.007385
```

Run a block of R code again ...

```
se <- 1.5
b0 <- 0.7
b1 <- 0.04
y <- b0 + b1 * x1 + se * rnorm(1000)
dat <- data.frame(x1, y)
m1 <- lm(y ~ x1, data = dat)
summary(m1)
```

```
Call:
lm(formula = y ~ x1, data = dat)

Residuals:
    Min      1Q   Median      3Q     Max
-4.5454 -1.0726 -0.0398  1.0519  4.6467

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.66429    0.04717  14.082   <2e-16 ***
```

Run a block of R code again ...

```
x1          -0.00951    0.04684  -0.203    0.839
---
 Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0
    .1 ' ' 1

 Residual standard error: 1.491 on 998 degrees of
    freedom
 Multiple R^2:  4.13e-05, Adjusted R^2:  -0.0009607
 F-statistic: 0.04122 on 1 and 998 DF,  p-value: 0.8391
```

```
dat$x2 <- rpois(1000, lambda = 2)
m2 <- lm(y ~ x1 + x2, data = dat)
summary(m2)
```

Run a block of R code again ...

```
Call:
lm(formula = y ~ x1 + x2, data = dat)

Residuals:
    Min      1Q  Median      3Q     Max
-4.5306 -1.0589 -0.0347  1.0488  4.6466

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.693501   0.082888   8.367   <2e-16 ***
x1          -0.009255   0.046865  -0.197    0.843
x2          -0.014654   0.034180  -0.429    0.668
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0
    .1 ' ' 1
```

Run a block of R code again ...

```
Residual standard error: 1.492 on 997 degrees of
    freedom
Multiple R^2:  0.0002256,  Adjusted R^2:  -0.00178
F-statistic: 0.1125 on 2 and 997 DF,  p-value: 0.8936
```

```
 anova(m2, m1, test = "F")
```

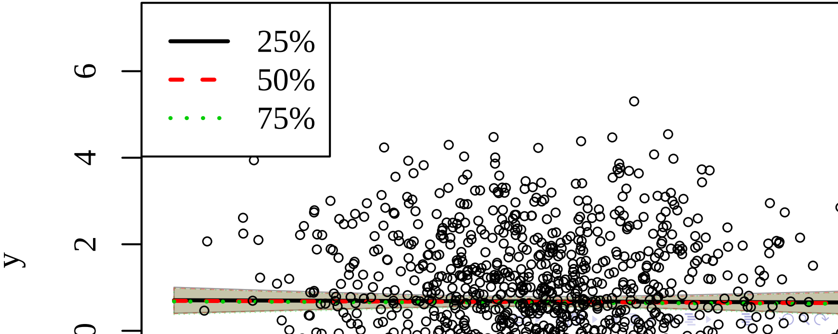
```
Analysis of Variance Table

Model 1: y ~ x1 + x2
Model 2: y ~ x1
  Res.Df    RSS Df Sum of Sq      F Pr(>F)
1    997 2219.6
2    998 2220.0 -1  -0.40922 0.1838 0.6682
```

I want a figure now

The figure comes out much too huge.

```
library(rockchalk)
plotSlopes(m2, plotx = x1, modx = x2,
    interval = "confidence")
```



Outline

1 Document Preamble Changes

2 Top of Document Boilerplate

3 Final Idioms

- Figure Customization
- Caching Results and Calculations
- Changes for You

4 Conclusions

Outline

1 Document Preamble Changes

2 Top of Document Boilerplate

3 Final Idioms
- Figure Customization
- Caching Results and Calculations
- Changes for You

4 Conclusions

Working with Figures

I want figures saved separately, and I'll insert them however I like.

- Change the chunk invocation, insert "include=F". The figure is created, but not displayed here

<<ps50, fig=T, include=F>>=
The image will be named "plots/t-ps50.pdf"

- We include the figure-generator code thus:

```
<<ps50,fig=T,include=F>>=
plotSlopes(m2, plotx = x1, modx = x2, pred-
Vals="std.dev.", interval = "confidence")
@
```

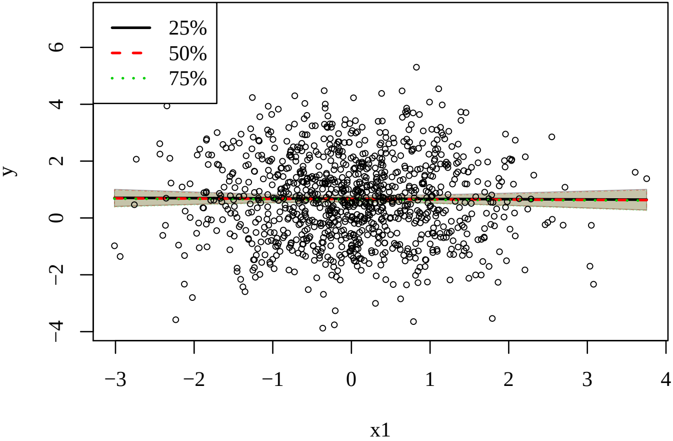
And then the standard methods of inserting the figure into the document are used

Working with Figures ...

```
\includegraphics[width=10cm]{plots/t-ps50}
```

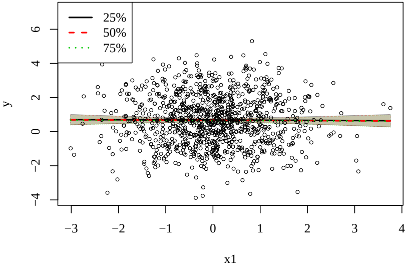
Don’t add the suffix on graphics file names.

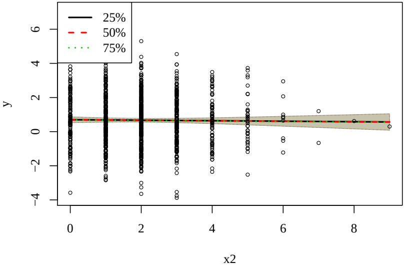
Working with Figures ...



Working with Figures ...

But if I was feeling spunky, I could put 2 pictures on one slide as well.





Same trick works with text from code chunks too

```
\input{plots/t-ps60.tex}
```

* Do insert suffix with \input
Here's a replay of chunk ps60

```
 plotSlopes(m2, plotx = x2, modx = x1,
     predVals="std.dev.", interval = "
     confidence")
```

Outline

1 Document Preamble Changes

2 Top of Document Boilerplate

3 Final Idioms
- Figure Customization
- Caching Results and Calculations
- Changes for You

4 Conclusions

Long R Jobs = Unhappy Sweaving

- If your R code takes more than a minute or two, then working with Sweaved calculations can be very frustrating.
- There are 3 ideas I've tried.
  1. Some R packages exist which are supposed to make Sweave smarter so that it does not re-calculate all results. They cache calculations from unaltered code chunks. I like the idea, but have not had great experience with it. The knitr package tries to do that as well.
  2. In the LyX document, use the Document Branches feature to separate long-running code chunks. When they run, write the results into the "plots" folder, and then Disable the branch until you need to run it again. I think this is the easiest way to manage this without straining yourself.
  3. Write two documents, one LaTeX document with the R code and the Sweaving commands.

Long R Jobs = Unhappy Sweaving ...

1. Be sure to label all of the code chunks, so they are written into “plots”.
2. Then in the presentation document, simply use the results from “plots” as you want to include them in the document.
3. This is the best of both worlds. The R run is completely replicable. And the LaTeX document does not need to wait for R to run.

Outline

1 Document Preamble Changes

2 Top of Document Boilerplate

3 Final Idioms
- Figure Customization
- Caching Results and Calculations
- Changes for You

4 Conclusions

Changes that You Should Try

1. Change your Beamer theme.
    1. I changed from the "Warsaw" theme that the LyX team preferred to the "Antibes" theme, which is a bit less intrusive (IMHO)
    2. In the R Summer Program, Pascal Deboeck and I have used a theme we adapted (which he obtained from his advisor at ND).
    3. Why don't you try that one out. I've left a copy of the beamerthemeku.sty file in this directory, so all you should need to do is to change your preamble to read

        ```
        \usetheme{KU}
        \usecolortheme{dolphin}
        ```

        This will only work if your system has the beamerthemeku.sty file, but there's a copy in the directory where you find this presentation.

Changes that You Should Try ...

    4. That theme has top and bottom banners. I recall we hacked it to make the top banner show the sections, but I grew bored with the crowding it would cause.

2. Fiddle with colors and sizes for R listings.
    1. I buried the color settings in the Preamble. At some times, I've had them in the top part of the document itself.
    2. In the preamble find this piece

```
\def\Rcolor{\color{black}}
\def\Rbackground{\color[gray]{0.95}}
```

You can change the word black to "purple" and the number 0.95 to 0.70, for example.

    1. Fiddle with the text size in the R chunk displays.
    2. In the Preamble, find this:

Changes that You Should Try ...

```
\def\Sweavesize{\normalsize}
```

And change to this

```
\def\Sweavesize{\scriptsize}
```

At some times, I have done that same thing INSIDE documents for particular pieces of output, but maybe you want to change the whole document.

If you are Going to Fiddle with a Document

- Put the document in a version management system like GIT or SVN, so you can go back in history to trace your changes
- This will happen. You have a deadline. An emergency, and
  - a big important document won't compile, and
  - the error messages are completely not understandable to you
- Here is what I do:
  1. Catch your breath. Save the document. Relax, leave your computer for a few minutes on meditate about the changes you made in the time since the document did compile and when it did not.
     - Then go back to the document, delete or comment out (or mark as Lyx-Notes) the suspicious bits and try to compile.
     - If the document is in Version Control, you can step back in history, see what you did.

If you are Going to Fiddle with a Document ...

2. If that does not work, Bisect the document.
   - Remember the error messages don't tell you where your mistake IS, they tell you when the LaTeX system finally notices them.
   - Highlight the last 95%, delete it, and see if the document compiles.
   - That narrows it down a bit.
   - Paste the deleted part back, and then delete the last 75%, try to compile. Repeat until victory obtains.

Outline

1 Document Preamble Changes

2 Top of Document Boilerplate

3 Final Idioms
- Figure Customization
- Caching Results and Calculations
- Changes for You

4 Conclusions

There Exist Helpful Customizations

- You know everything I know, now
- And you can know everything I used to know too:
  - If you find one of my presentations under `http://pj.freefaculty.org/guides` and the PDF looks different, I left the source code there so you can find out what I was doing differently way back when.
- To become truly proficient at this, all you need is the firm belief that you can eventually make it work. And to sustain that belief, all you need is an example that compiles.

There Exist Helpful Customizations

I am between 2 extremes.

- Some people understand LaTeX markup and like looking at it. They feel comfortable with that.
- Some people can't comprehend it at all and they use MS Word (fingerpainters)
- I can understand enough LaTeX to make a few customizations in these documents, but I have not studied hard enough to do a heck of a lot.

The Document Liferaft

- If you are closer to the 2'nd type of person, as most are, then here is the right path to follow.
- Find people who make documents you admire. Get their source code. Cling to the working example like a liferaft.
- Then make a small changes, one by one, to make sure it still compiles.
- DO NOT surf the Internet and find "code snippits" from other people that work for them, and paste them into your document, unless you are ready to work hard to make them work for you.

References

Leisch, F. (2002a). Sweave: Dynamic generation of statistical reports using literate data analysis. In Härdle, W. and Rönz, B., editors, *Compstat 2002 — Proceedings in Computational Statistics*, pages 575–580. Physica Verlag, Heidelberg. ISBN 3-7908-1517-9.

Leisch, F. (2002b). Sweave, part I: Mixing R and LaTeX. *R News*, 2(3):28–31.