Graphics Workflow

Paul E. Johnson[1] [2]

2018

CENTER FOR RESEARCH METHODS & DATA ANALYSIS
College of Liberal Arts & Sciences

Outline

KU

Outline

High Level plot functions in R Base

- functions provided with base R R Core Team (2017)

create a "device"

plot	hist	barplot
plot.default	boxplot	dotchart
matplot	coplot	

- Run "example(hist)", "example(barplot)", and so forth
- Run "demo(graphics)"
- The "dotchart" is a new favorite.

Low Level plotting functions

- Low Level: added accents or features
- High level functions create basic plot framework, coordinates
- In lineaRt, we were mostly interested in these low level functions

text	points	lines	box
arrows	segments	mtext	abline
axis	legend	title	polygon
rect			

- Uniform in style of usage

Special Things Worth Mentioning

- plotmath: a specialized LaTeX-like language to add math and symbols. See examples in http://pj.freefaculty.org/R/WorkingExamples
- We have detailed lectures on usage of different kinds of plots: http://pj.freefaculty.org/guides/Rcourse. See folders plot-1, plot-2, and plot-3d

Other Graphics Paradigms

- The lattice package. A recommended package distributed with R
- The ggplot2 package, created by prolific R contributor Hadley Wickham

Outline

On Screen versus In A File

- This is confusing, but . . .
  - the graph you see on the screen is not easily saveable into an image file that looks exactly right
  - flaws occur because of differences in device shape and size
- Difficulty in saving high-quality graphs was the primary reasons I wrote Rtips.html (`http://pj.freefaculty.org/R/Rtips.html`).

KU

3 "use cases" we consider

1. Create 100 plots and save into separate pages in a pdf file
2. Create 100 plots and save one into 100 separate pdf files
3. Create 1 plot and save it into one file

Naturally, we start with number 3☺

Simplest example

1. Create an open "device file" named "myfirst.pdf" in your current working directory
2. Run one graph command
3. Save the file (let R know you are done drawing in there).

```
pdf("myfirst.pdf")
hist(rnorm(1000), main =
    "1000 random normals")
dev.off()
```

- No graph will appear on screen
- File named "myfirst.pdf" should appear in the working directory

```
list.files()
```

Problem in the workflow

- You don't get to "see" the graph when it is saved.
- The best you can do is run a command, and view on screen, and then
  - run again, after
  - wrapping that command between "pdf()" and "dev.off()".
- This is the only way to assure yourself that your output file is high in quality, *but it is, admittedly, inconvenient*.

KU

2 corrections for that workflow problem

- Create an on screen device that is the "right size". Practice plots in there, so that when you save to the intended size, they will come out correctly.
  - Run `dev.new(height = 5, width = 7)`
  - Rstudio will block that, need to run a system-dependent command like
    - `windows(height = 5, width = 7)`
    - `quartz(height = 5, width = 7)`
    - `X11(height = 5, width = 7)`
- Write code to anticipate the need to save graphs into files. Thats what I call SAVEME here:

2 corrections for that workflow problem ...

```
SAVEME <- FALSE
if (SAVEME){
   pdf(file = "afilename.pdf", paper =
      "special",
      height = 5, width = 7, onefile = FALSE,
         family = "Times")
 } else {
   ## If you are not using Rstudio, this
      works in an interactive session
   if (interactive()) dev.new(height = 5,
      width = 7)
 }
plot(y ~ x, data = dat, xlab = "My super
   plot", ylab = "My other variable", main =
   "smarter than you")
lines(z ~ x, data = dat, col = "green")
if (SAVEME) dev.off()
```

2 corrections for that workflow problem ...

- When you are finished with "on screen" viewing, change `SAVEME` to `TRUE`.
- Example usages in http://pj.freefaculty.org/R/WorkingExamples
- normal versus T probability densities distributions-normalAndTCompared.R
- Gamma and normal compared distributions-GammaVersusNormal-1.R
- Logistic regression with unbalanced dichotomous predictors glm-logit-unbalanced-1.R

outdir

- I don't generally want output in the "current working directory", I want it in the "`output`" folder.

```
odir <- "output/"
chk <- dir.create(odir, recursive = TRUE)
print(chk)
```

```
[1] FALSE
```

`dir.create` returns TRUE or FALSE, indicating it succeeded.

```
pdf(file.path(odir, "myfirst.pdf"))
hist(rnorm(1000), main = "1000 random
    normals")
dev.off()
```

```
pdf
  2
```

outdir ...

- In 2016 I became worried about accidentally erasing valuable output. I wrote a function to prevent that "`dir.create.unique`" that will prevent overwrites.

```
odir2 <-
    rockchalk::dir.create.unique("output/")
odir2
```

```
[1] "output/20190828-4/"
```

- Unlike `dir.create`, `dir.create.unique` returns the name of the directory it created.
- Write a file in there

```
pdf(file = file.path(odir2, "myfirst.pdf"))
hist(rnorm(1000), main = "1000 random normals")
dev.off()
list.files(odir2)
```

The devices have other arguments

- pdf arguments I generally use

```
pdf(file = "afilename.pdf", height = 5, width
    = 7,
 paper = "special", onefile = FALSE, family =
    "Times")
```

- I change the first 3 on a case-by-case basis.
- height and width are in inches

onefile = TRUE

- Save a series of plots as "pages" in a pdf document:
  - onefile = TRUE

Example Use case

- A client has small samples of "count" data. Wonders if his sample "looks funny".
- Asks to see 100 random samples from a Poisson with 20 observations and expected value of 4

Example of onefile = TRUE

- If SAVEME is FALSE, this should draw histograms on screen, asking for your permission each time

```
SAVEME <- TRUE
if (SAVEME){
      odir <-
          rockchalk::dir.create.unique("output/")
      pdf(file = file.path(odir, "histo-100.pdf"),
          onefile = TRUE)
 } else {
   par(ask = TRUE)
 }
set.seed(234)
for(i in 1:100){
   x <- rpois(20, lambda = 4)
   hist(x, main = paste("Run number", i),
            breaks = -0.5:15.5,
```

Example of onefile = TRUE ...

```
            xlim = c(0, 15), xlab = "histogram",
          )
  }
if (SAVEME) dev.off()
```

```
pdf
  2
```

onefile = FALSE

- If you put
  - onefile = FALSE, and
  - file = "histo-%03d.pdf"

  Then you get separate plot files "histo-001.pdf", "histo-002.pdf"
- Use case for this: We are writing a paper and want to make the point that some samples generated from a Poisson distribution look peculiar. We'll pick through the separate PDF files and include some in a presentation.

KU

onefile = FALSE

```
SAVEME <- TRUE
if (SAVEME){
    odir <-
        rockchalk::dir.create.unique("output/")
    print(odir)
    pdf(file = paste0(odir, "/histo-%03d.pdf"),
        onefile = FALSE)
 } else {
   par(ask = TRUE)
 }
```

```
[1] "output/20190828-6/"
```

onefile = FALSE ...

```
set.seed(234)
for(i in 1:100){
   x <- rpois(20, lambda = 4)
   hist(x, main = paste("Run number", i),
5           breaks = -0.5:15.5,
            xlim = c(0, 15), xlab = "histogram",
          )
 }
if (SAVEME) dev.off()
```

```
pdf
  2
```

```
## the first 30 file names
list.files(odir)[1:30]
```

onefile = FALSE ...

```
 [1] "histo-001.pdf" "histo-002.pdf" "histo-003.pdf"
 [4] "histo-004.pdf" "histo-005.pdf" "histo-006.pdf"
 [7] "histo-007.pdf" "histo-008.pdf" "histo-009.pdf"
[10] "histo-010.pdf" "histo-011.pdf" "histo-012.pdf"
[13] "histo-013.pdf" "histo-014.pdf" "histo-015.pdf"
[16] "histo-016.pdf" "histo-017.pdf" "histo-018.pdf"
[19] "histo-019.pdf" "histo-020.pdf" "histo-021.pdf"
[22] "histo-022.pdf" "histo-023.pdf" "histo-024.pdf"
[25] "histo-025.pdf" "histo-026.pdf" "histo-027.pdf"
[28] "histo-028.pdf" "histo-029.pdf" "histo-030.pdf"
```

KU

My Real Workflow has a Couple of other Blandishments #1

- At the top of an R file that uses graphs, I set some default options.

```
pdf.options(onefile = FALSE, family = "Times",
    paper = "special", height = 4, width = 6)
```

- In each (SAVEME) pdf command, I don't have to type so much

```
if (SAVEME) pdf(file = "afilename.pdf")
## make a plot, silly
if (SAVEME) dev.off()
```

- The pdf arguments are still allowed if we want to customize. Graphs are tall and narrow:

My Real Workflow has a Couple of other Blandishments #1 ...

```
if (SAVEME) pdf(file = "afilename.pdf", height =
    10, width = 7)
## make a plot, silly
if (SAVEME) dev.off()
```

Outline

Saving in other image formats

- PDF is a scalable vector graphic format (the successor to "encapsulated postscript")
- You may want "Picture" formats, like "jpg" or "png" are "bitmaps"
  - Not as nice, not as scalable, but co-operate with MS Word more readily
- R does have a device to create those formats

```
if (SAVEME) png(file = "afilename.png", height =
    800, width = 800)
## draw your plot here
if (SAVEME) dev.off()
```

A picture file requires height and width in pixels, NOT INCHES. pdf and postscript require inches.

I don't do this often, I find it much nicer to export to pdf and convert to other formats from there with ImageMagick

Outline

Reproducible Output

This seems tedious, but

- In order to have a fully reproducible workflow, it is necessary to write commands that save files on disk
- Avoid
  1. "copying and pasting"
  2. pull down "File -> Save as"

Three step process

- If you are working on screen, think of making graphs as a 3 step process.
  1. Create a device (dev.new(), or windows(), or quartz())
  2. Run graphics commands (hist, plot, etc) to write on the device
  3. Close that "window" when you are done
- If you get in the habit of thinking of this as 3 steps, then it will not be such a shock when you are interested in saving graphs into files.
  1. Create an output device file (pdf, png, etc)
  2. Run same graphics commands as step 2
  3. dev.off()

KU

Expect Change

- Right now, the pdf is the default device file format
- In 1998, when I started with R, the default was postscript
- New devices will come along.
  - SVG is appealing (browser compatibility, edit with Inkscape)
  - tikZ is a LaTeX compatible framework

References

R Core Team (2017). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Session

```
sessionInfo()
```

```
R version 3.6.0 (2019-04-26)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 19.04

Matrix products: default
BLAS:   /usr/lib/x86_64-linux-gnu/atlas/libblas.so.3.10.3
LAPACK: /usr/lib/x86_64-linux-gnu/atlas/liblapack.so.3.10.3

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
     LC_TIME=en_US.UTF-8
 [4] LC_COLLATE=en_US.UTF-8     LC_MONETARY=en_US.UTF-8
     LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C                  LC_ADDRESS=C
[10] LC_TELEPHONE=C             LC_MEASUREMENT=en_US.UTF-8
    LC_IDENTIFICATION=C

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

loaded via a namespace (and not attached):
```

Session ...

```
 [1] Rcpp_1.0.1          lattice_0.20-38    MASS_7.3-51.4      grid_3.6.0
              plyr_1.8.4
 [6] nlme_3.1-140        xtable_1.8-4       stats4_3.6.0       zip_2.0.2
               carData_3.0-2
[11] rockchalk_1.8.144 minqa_1.2.4         nloptr_1.2.1
     Matrix_1.2-17      pbivnorm_0.6.0
[16] boot_1.3-22         openxlsx_4.1.0     splines_3.6.0      lme4_1.1-21
           tools_3.6.0
[21] foreign_0.8-71      kutils_1.69        compiler_3.6.0     mnormt_1.5-5
          lavaan_0.6-3
```