Style Tips

Instructor - Soumya Mukherjee

Content Credit- Dr. Matthew Beckman and Olivia Beck

Aug 1, 2023

```
rm(list = ls())
knitr::opts_chunk$set(echo = TRUE)



library(tidyverse)
```

```
Warning: package ‘tidyverse’ was built under R version 4.2.3Warning: package ‘ggplot2’ was built under R version 4.2.3Warnin
g: package ‘tibble’ was built under R version 4.2.3Warning: package ‘tidyr’ was built under R version 4.2.3Warning: package
‘readr’ was built under R version 4.2.3Warning: package ‘purrr’ was built under R version 4.2.3Warning: package ‘dplyr’ was
built under R version 4.2.3Warning: package ‘stringr’ was built under R version 4.2.3Warning: package ‘forcats’ was built un
der R version 4.2.3Warning: package ‘lubridate’ was built under R version 4.2.3── Attaching core tidyverse packages ────────
───────────────────────────────── tidyverse 2.0.0 ──
✔ dplyr     1.1.2     ✔ readr     2.1.4
✔ forcats   1.0.0     ✔ stringr   1.5.0
✔ ggplot2   3.4.2     ✔ tibble    3.2.1
✔ lubridate 1.9.2     ✔ tidyr     1.3.0
✔ purrr     1.0.1     ── Conflicts ────────────────────────────────────────────────────────── tidyverse_conflicts() ──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
ℹ Use the ]8;;http://conflicted.r-lib.org/conflicted package]8;; to force all conflicts to become errors
```

```
library(knitr)
```

```
Warning: package ‘knitr’ was built under R version 4.2.3
```

Hide

```
library(kableExtra)
```

```
Warning: package ‘kableExtra’ was built under R version 4.2.3Registered S3 methods overwritten by 'htmltools':
  method               from         
  print.html           tools:rstudio
  print.shiny.tag      tools:rstudio
  print.shiny.tag.list tools:rstudio

Attaching package: ‘kableExtra’

The following object is masked from ‘package:dplyr’:

    group_rows
```

Hide

```
library(mosaic)
```

```
Warning: package ‘mosaic’ was built under R version 4.2.3Registered S3 method overwritten by 'mosaic':
  method                           from   
  fortify.SpatialPolygonsDataFrame ggplot2

The 'mosaic' package masks several functions from core packages in order to add 
additional features.  The original behavior of these functions should not be affected by this.

Attaching package: ‘mosaic’

The following object is masked from ‘package:Matrix’:

    mean

The following objects are masked from ‘package:dplyr’:

    count, do, tally

The following object is masked from ‘package:purrr’:

    cross

The following object is masked from ‘package:ggplot2’:

    stat

The following objects are masked from ‘package:stats’:

    binom.test, cor, cor.test, cov, fivenum, IQR, median, prop.test, quantile, sd,
    t.test, var

The following objects are masked from ‘package:base’:

    max, mean, min, prod, range, sample, sum
```

Programming Style

"Good coding style is like using correct punctuation. You can manage without it, but it sure makes things easier to read"

–Hadley Wickham (Advanced R, p.63)

Style Guides (Recall Week 3)

- there isn't universal agreement on an exhaustive style guide for R
    - Tidyverse Style Guide (https://style.tidyverse.org/ (https://style.tidyverse.org/)) is the most complete style guide that I'm aware of
    - DataComputing eBook Style Guide (Appendix) is much shorter but generally agrees
- we'll focus on common elements of good programming style and prioritize those that have the greatest benefit for someone attempting to read and understand your code

Style Guides: Basics (graded)

- use **descriptive** but short object names
    - treat "descriptive" as higher priority, then shorten without sacrificing meaning
- use **white space** (for example code, see Advanced R >> Style >> Syntax >> Spacing; link: http://adv-r.had.co.nz/Style.html (http://adv-r.had.co.nz/Style.html))
    - always put spaces around infix operators (`<-` , `%>%` , `=` , `==` , `+` , `-` , etc)
    - always put a space after a comma, but never before
    - always put a space before parenthesis, **exception:** function calls should never have a space before the parenthesis
    - always put a space after the comment character: `# Helpful comment example`
- use `<-` for object assignment (not `=`)
- include **meaningful comments** in your code, place a space after the comment character

Style Guides: Object naming

- DataComputing eBook conventions
    - Data tables should start with a CAPITAL letter
    - variables should start with a **lower-case** letter
    - CamelCase: capitalize first letter of each word in name, but no spaces (e.g., `BabyNames` , `longVariableName`)
- Advanced R Text (Chapter 5)
    - snake_case: underscore instead of spaces (e.g., `Baby_names` , `long_variable_name`)
    - avoid names that correspond to existing functions (e.g. `mean` , `sum`)

- **Avoid using the same name to mean different things in different parts of your script**
    - I strongly recommend beginning (nearly) every script by cleaning your R environment: `rm(list = ls())`

Related best practices

- consistent document organization
    1. start with a clean R environment: `rm(list = ls())`
    2. load all required packages: `library( )`
    3. user-defined functions–include generous comments defining purpose and arguments function
    4. data intake (e.g. read raw data from source)
    5. body of code
        - data cleaning/processing for use
        - data analysis (wrangling, visualization, modeling, etc.)
        - in Rmd, nearly always organized across many code chunks with narrative in between
    6. outputs for external use (e.g. write resulting table to CSV, if necessary)
- "DRY" code (don't repeat yourself)
    - avoid "hard coded" conditions you plan to reference several times, or want to easily change in the future
        - e.g. see the confidence interval simulation from the class notes
    - write user-defined functions if you find yourself doing a set of operations repeatedly in the code
- use version control (e.g. Git/GitHub)
- and more! https://swcarpentry.github.io/r-novice-inflammation/06-best-practices-R/ (https://swcarpentry.github.io/r-novice-inflammation/06-best-practices-R/)

R Markdown Formatting Basics

- So far, we haven't emphasized formatting at all (on purpose)
- Ideally, the content of your work is far more important than it's appearance
- Sometimes a bit of formatting to make a document look nicer has value too (hopefully never at the expense of content)
- Markdown, LaTeX, and others provide some ability to separate content and formatting
- MS Word (and Google Docs, Mac Pages, etc) generally take a WYSIWYG approach
- We'll just demonstrate a few basics here, but there is **much** more available if your interested:
    - RStudio Cheat Sheet for R Markdown
    - R Markdown Reference Guide available from RStudio as well
    - Free Book: *R Markdown: The definitive guide* by Xie, Allaire, & Grolemund. https://bookdown.org/yihui/rmarkdown/ (https://bookdown.org/yihui/rmarkdown/)

    - Google searches, websites, and books are helpful too
- By the way, **the embedded .Rmd is much more useful than the html notes for this discussion**

R Markdown Formatting Basics

If you hang around R Markdown long enough, you'll hear a lot about "knitr", "yaml", "pandoc", and other odd terms related to configuring the resulting document produced. Even if it all feels a bit jumbled together, you can still learn to troubleshoot and accomplish what you're after in most cases. It's probably best to tackle most of these things only as the need arises.

We'll just highlight a few favorites here:

- Tables
- Math
- Chunk Options
- Global Options
- Knitting to other document types

Tables

We'll use the `mtcars` data for illustration. We used these data last week when discussing Base R, and had fit a regression model among other things. We'll pick up there again.

Here's a look at the first few rows:

Hide

```
head(mtcars)
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1
Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
Hornet Sportabout	18.7	8	360	175	3.15	3.440	17.02	0	0
Valiant	18.1	6	225	105	2.76	3.460	20.22	1	0

6 rows | 1-10 of 11 columns

The defaults have improved over the years, but if you want a little more control of the table appearance, you could wrap it in a `knitr::kable()` function from the `knitr` package:

Hide

```
knitr::kable(head(mtcars))
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225	105	2.76	3.460	20.22	1	0	3	1

Then, you can use arguments to `kable()` if you want it to round all numbers, change column alignment, include a caption, etc. Other functions exist in the `stargazer` , `xtable` , `kableExtra` , and `mmtable2` packages have additional capabilities.

Chunk Options

Chunk options let you control appearance and results produced by code chunks

- adjust dimensions for figures produced by that chunk (to do so for all figures, set a global option)
- suppress messages (like when loading packages)
- toggle whether or not to show and/or run the code chunk in the output doc (but always show code for STAT 184)
- and more

If you have several commands in sequence, sometimes the code chunk will break in the middle to print intermediate results. If you want to just print everything after the entire chunk you can use `hold` in the `results` argument where `hide` appears now. Other chunk options that you might want to play with are `include` , `eval` .

Math

RMarkdown makes it super easy to make math look nice. The syntax is based on LaTeX syntax, but there are online latex equation editors to help you get the hang of it:

- https://www.latex4technics.com/ (https://www.latex4technics.com/)
- https://www.codecogs.com/latex/eqneditor.php (https://www.codecogs.com/latex/eqneditor.php)
- google it to find others…

Note that you can easily embed mathematical content (e.g. e^{β_1}) within a sentence.

STAT 462 excerpt (inline math)…

Here's the model:

$log(\frac{\hat{p_i}}{1-\hat{p_i}}) = 3.309 - 0.288(dist_i)$

where $\hat{p}_i$ is the proportion that voted "Yes" for community *i*.

Since the relationship on a logarithmic scale is hard to interpret, we back transform before interpreting coefficients.

Note: If $e^{\beta_1} = 1$, then it means the odds are 1:1 which translates to a 50-50 chance and means there would be no relationship between the explanatory variable and the odds of "success" (however that's defined in the context of the study).

STAT 415 excerpt (aligned to `=` sign)…

$$SSE = \Sigma_i (y_i - b_0 - b_1 x_i)^2$$

$$\begin{aligned}
\frac{\partial\left(SSE\right)}{\partial b_0} &= \Sigma_i (y_i - b_0 - b_1 x_i)^2 \\
&= (-2)\,\Sigma_i\,(y_i - b_0 - b_1 x_i) \\
&= (-2)\,(\Sigma y_i - \Sigma b_0 - \Sigma b_1 x_i)
\end{aligned}$$

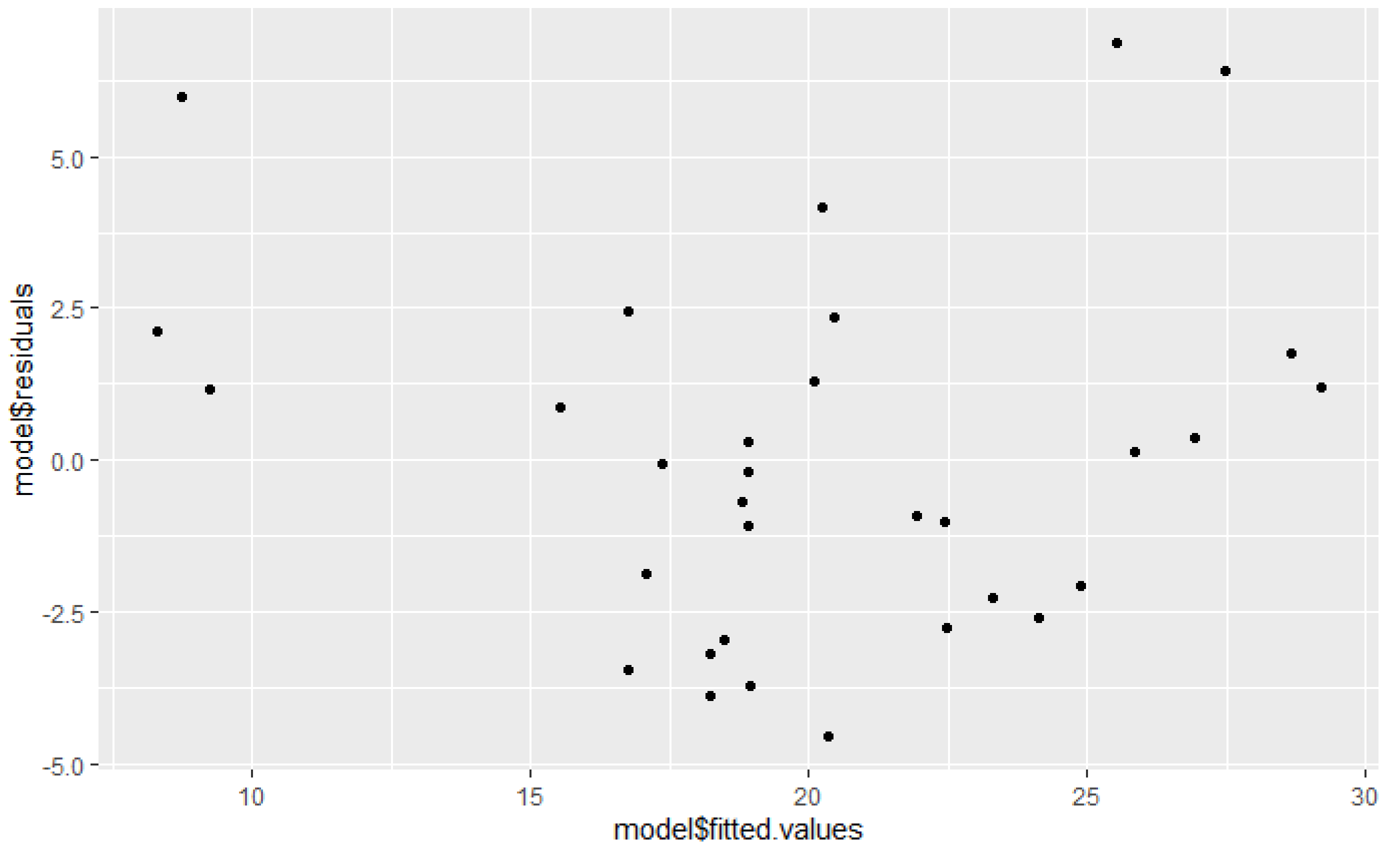
$$\begin{aligned}
\frac{\partial\left(SSE\right)}{\partial b_1} &= \Sigma_i (y_i - b_0 - b_1 x_i)^2 \\
&= \Sigma_i\,(-2x_i)\,(y_i - b_0 - b_1 x_i) \\
&= (-2)\,\Sigma_i\left(x_i y_i - x_i b_0 - b_1 x_i^2\right) \\
&= (-2)\left(\Sigma x_i y_i - \Sigma x_i b_0 - \Sigma b_1 x_i^2\right)
\end{aligned}$$

Figures

Last time we made a residual plot to accompany a simple linear regression model:

Hide

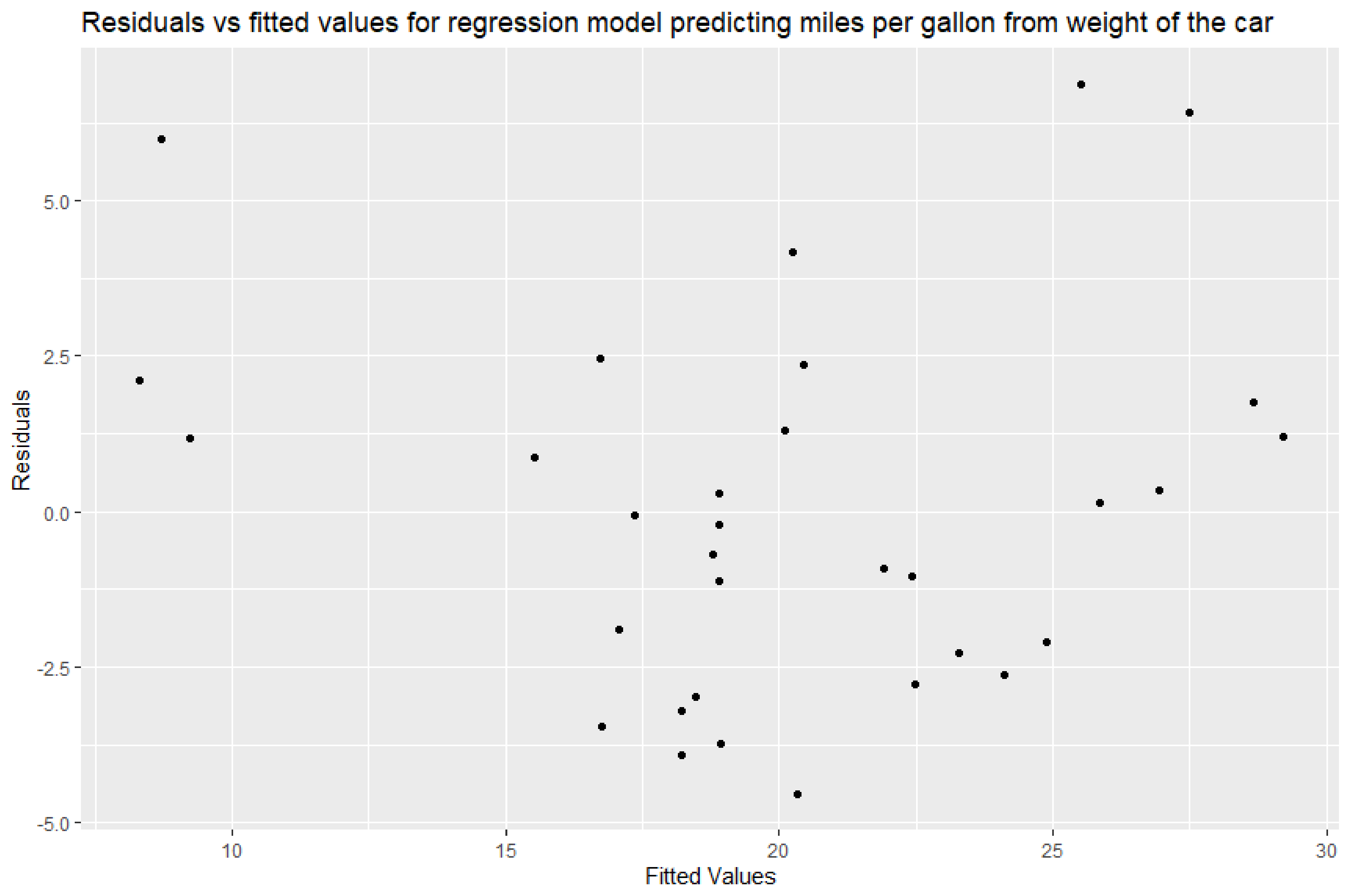
```
model <- lm(mtcars$mpg ~ mtcars$wt)   # model predicting miles per gallon from weight of the car

ggplot() +
  geom_point(aes(x = model$fitted.values, y = model$residuals))
```



If we have several graphs in the document and would like to change the size for this one, we can do that as a chunk option. I also labeled the axes & added a title (you should always do this…)

Hide

```
ggplot() +
  geom_point(aes(x = model$fitted.values, y = model$residuals)) +
  ggtitle("Residuals vs fitted values for regression model predicting miles per gallon from weight of the car") +
  xlab("Fitted Values") +
  ylab("Residuals")
```



Global arguments

You can use a code chunk to set global options as follows:

```
\```{r include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
\```
```

You can also set some global options in the “YAML header” at the top of your Rmd document by specifying key-value pairs that are interpreted when the document is rendered. The gear next to the “Knit” button in RStudio opens a window to help you to configure common global options for the document. You can find other key-value pairs for more advanced configuration in the RMarkdown cheat sheet, RMarkdown user manual, Google Searches, etc. As an example, here’s what the yaml header might look like for this document. You can see several key-value pairs dictating the global default for figure dimensions in addition to specifying the title, author, date, etc. By the way, the indenting matters… if you delete the spaces before “html_document” you won’t be able to render the document.

```
---
title: "Code Clean Up"
author: "Matthew Beckman"
date: "April 28, 2021"
output:
  html_notebook:
    fig_height: 4
    fig_width: 6
---
```

You can even impose a global formatting template by calling a separate file with a .css extension in the yaml header. Then if you want to change the formatting template, all you have to do is switch the .css file!

Knitting to other document types

- pandoc is a “universal document converter” so you can convert the same Rmd document into many different output documents
- YAML (Yet Another Markup Language; YAML Ain’t Markup Language) is a human friendly data serialization standard for all programming languages
- The “yaml header” at the top of the document provides instructions that pandoc uses to render different document types.
- We’ll look at a simple example using the default RMarkdown template document.
    - If you click the arrow next to the “Knit” button and yarn ball, you can select the different document types
    - The gear next to knit allows you to configure global options for the document
    - based on what you choose for both of the above, some code will show up in your yaml header to pass those instructions to pandoc for next time!

- You’ll need to have a LaTeX engine installed to render the PDF document types
    - In the “bad old days” this used to be a fairly significant configuration headache, but now it’s two lines of code that you only run one time in the console…

```
install.packages("tinytex")
tinytex::install_tinytex()
```