what should you do next?

@JennyBryan @jennybc

@STAT545 http://stat545.com
Data, code, figs, results living in harmony in one Big Happy Directory? NOT SUSTAINABLE

How to deal with paths and working directory in real world projects with subdirectories?
@STAT545 students running each other’s rstats pipelines. I know no better cure for:

```r
setwd("C:\Users\jenny\path\that\only\I\have")
```

In @STAT545, students run their peer’s data analysis pipeline. That solved the `setwd()` misunderstanding and problem pretty fast. rstats
How can something that feels so right be so wrong?
Your collaborators,  
future you,  
your readers,  
your VM,  
your Docker container,  
your cluster,  
.......  

will NOT share the same directory structure!
Make your path handling portable!

Build paths relative to “project root”.

http://poisotlab.io/2016/04/14/project-organization/
rmarkdown / RStudio presents a special challenge during interactive execution, working directory can be anything you want

“project root” is possibly the best choice and the path of least resistance

however, at render time, working directory = directory where the .R or .Rmd file lives
rprojroot
https://cran.r-project.org/package=rprojroot

ezknitr
https://cran.r-project.org/package=ezknitr
rprojroot

Robust, reliable and flexible paths to files below a project root. The 'root' of a project is defined as a directory that matches a certain criterion, e.g., it contains a certain regular file.
ezknitr

An extension of 'knitr' that adds flexibility in several ways. One common source of frustration with 'knitr' is that it assumes the directory where the source file lives should be the working directory, which is often not true. 'ezknitr' addresses this problem by giving you complete control over where all the inputs and outputs are, and adds several other convenient features to make rendering markdown/HTML documents easier.
source(my_personal_functions.R)

Do you do this at the top of every script?
read.csv(really_important_data.csv)

Do you do this at the top of every script?
Yes, you admit it?

It’s time to make an R package!

package = fundamental unit of R-ness

can bundle functions or data or both
http://r-pkgs.had.co.nz

https://hilaryparker.com/2014/04/29/writing-an-r-package-from-scratch/
Write your own R package

- Setting the stage
- Prepare your system for package development
- Hands-on activity
- Resources

http://stat545.com/packages00_index.html

devtools

https://cran.r-project.org/package=devtools
From earlier …

Strong recommend: use the “tidyverse”
- tibble + dplyr + tidyr

Now adding to that
- the pipe operator %>%
- the purrr package
- use of nested tibbles, list-columns
foo_foo <- little_bunny()

bop_on(
  scoop_up(
    hop_through(foo_foo, forest),
    field_mouse
  ),
  head
)

# vs
foo_foo %>%
  hop_through(forest) %>%
  scoop_up(field_mouse) %>%
  bop_on(head)
**dplyr + tidyr + purrr**
for data wrangling and aggregation

**filter()** and **select()** for targeting specific rows or variables

**mutate()** for creating or mutating variables

**group_by()** for creating conceptual groups of rows

**summarize()** for computing on groups

“shock and awe” re: nested nibbles, list-columns
dpolyr for data wrangling and aggregation

Gapminder example:

I have found friends and family love to ask seemingly innocuous questions like, “which country experienced the sharpest 5-year drop in life expectancy?”. In fact, that is a totally natural question to ask. But if you are using a language that doesn’t know about data, it’s an incredibly annoying question to answer.

http://stat545.com/block010_dpolyr-end-single-table.html
go to coding demo
dplyr + tidyr + purrr
for data wrangling and aggregation

nest() for creating meta-observations from groups of rows (nested tibbles, with data in list-columns)

complicated things, e.g., models, can go in a list-column

mutate() + purrr::map() functions can be used to post-process models

mutate(new_var = map(old_var, fun))

goal is always to get back to a “normal” data frame

http://r4ds.had.co.nz/many-models.html
df %>% group_by() %>% nest()

is not the only way to get list-columns

I often make list-columns directly

then use this same pattern

mutate(new_var = map(old_var, fun))

and work my way back to a “normal” data frame
go to coding demo
More examples of purrr usage

https://github.com/jennybc/send-email-with-r#readme

https://github.com/jennybc/analyze-github-stuff-with-r#readme

https://github.com/jennybc/manipulate-xml-with-purrr-dplyr-tidyr#readme
dpolyr + tidyr + purrr

with heavy use of list-columns

is how I do all iterative tasks

i.e. has taken the place of

- for loops  (ok I haven’t used those in very long time)
- base “apply” family of functions
- plyr package
R Markdown Notebooks

Talk from June 2016 useR! @ Stanford

Documentation
http://rmarkdown.rstudio.com/r_notebook_format.html
go to coding demo
#rstats
#bioconductor
follow people
watch package repos

https://support.bioconductor.org