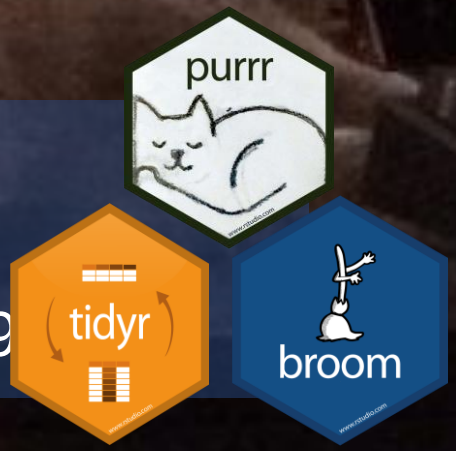
Managing many models with map functions

UGRU

Joey Burant

February 2019



Last time -- applying multiple functions to the same object (with the pipe %>%)

Today -- applying the same function to multiple objects (i.e., iteration)

Common problem, many solutions

Have you ever needed to perform the same operation(s) on a number of different objects/datasets/subsets?

(Yes, probably! If not, you will!)

Common approaches:

- ~~Copy, paste, modify~~ -- please find another way
- for loops -- repeat a sequence of instructions under certain conditions; imperative programming; good place to start
- Functional programming -- manipulate slices of data in a repetitive way by applying a named functions with one or several optional arguments; this is where R shines!
  - apply functions -- apply(), lapply(), mapply(), rapply(), etc.
  - map functions -- map(), map2(), pmap()

Functional programming … meow

- Functional programming offers a way to reduce duplication in your code (→ solve common iteration problems with *greater ease* and *fewer errors*)
- The pattern of looping over a vector (or list of objects) is so common that purr provides a whole family of functions to do just that!
- purrr enhances R's functional programming toolkit with a complete, tidy set of tools for working with functions and vectors

map functions

- map functions transform their input by applying a function to each element and return a vector the same length as the output
- As with other tidy tools, map() also has extensions for different classes of output (e.g., map_dbl(), map_chr(), etc.)
- map vs. apply functions?
- Other purrr-fect functional programming tools:
  - map2() -- iterate over two vectors in parallel
  - pmap() -- loop over p vectors in parallel (i.e., map3(), map4(), …)

Other useful functions in purrr

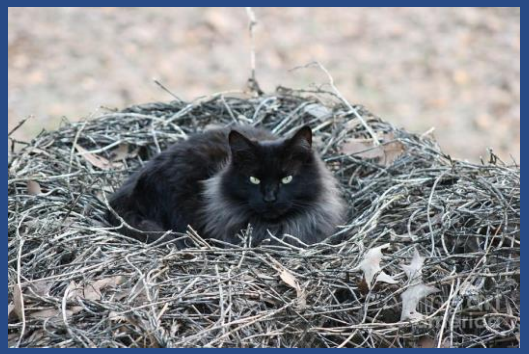
- walk() -- a variant of map() for functions called primarily for their side effects; output is returned invisibly

- pluck() -- select an element by name or index
- keep() -- select elements that pass a logical test; similar to select() and filter()
- discard() -- select elements that do not pass a logical test; similar to reject() and drop()

See “cheat sheet” for more!

tidyr

Useful functions in tidyr:



- nest() -- create a list of data frames containing all the nest variables; nest columns inside a data frame
- unnest() -- take a list column and make each element of the list its own row

Other functions: gather(), spread (), separate(), unite()

broom

Cleaning up after an analysis with untidy tools

- augment() -- takes a model object and a dataset and adds information about each observation to the dataset (e.g., extracting fitted values)
- glance() -- construct a single row summary of a model, fit, or other object
- tidy() -- turn a model object into a tidy tibble

broom

Cleaning up after an analysis with untidy tools

- augment() -- ta ect and a dataset and ad bout each observation to t , extracting fitted values)
- glance() -- cons w summary of a model, fit, or ot
- tidy() -- turn a model object into a tidy tibble

Resources

- tidyr

  - https://tidyr.tidyverse.org
  - https://cran.r-project.org/web/packages/tidyr/tidyr.pdf
  - http://vita.had.co.nz/papers/tidy-data.html

- purrr

  - https://purrr.tidyverse.org
  - https://r4ds.had.co.nz/iteration.html
  - https://github.com/rstudio/cheatsheets/blob/master/purrr.pdf

- broom

  - https://cran.r-project.org/web/packages/broom/vignettes/broom.html