Tidyverse intro

R4ds.had.co.nz

David Robinson, that guywho did sentiment analysis on Trump’s tweets, goes to these

Also Kolchak Reddy!

Flowchart:

Import -> Tidy -> Transform -> Visualize -> Model -> [Communicate, Transform]

Import is get it into format that works in R

Tidy is store in consistent format – remember the manifesto

Transform – eg compute summaries or put things in terms of variables you like better

Viz can surprise you, show you stuff you don’t expect; and refine your qs a/b the data

Human brain doesn’t scale, tho – can’t look at every possible graph w/ many vars

That’s why need modeling, once you’ve made your question sufficiently precise

A model can’t surprise you, viz can. Viz doesn’t scale well, models do. So, do both and iterate, alternate them.

Once you’ve got your final analysis, need to be able to communicate it to others

Tidyverse is set of packages designed to work well together to make steps easier, make sure you don’t need to fight w/ your tools

How do you design tools that are tidy? You do you write tidy code?

1. Share data structures across packages
2. Compose simple pieces you can easily understand individually to solve complex problems
3. Embrace functional programming to work w/ more ease, produce code that’s easier to write and easier to understand
4. Write for humans

Share data strs

Wanta data strs consistent across packages you use, not just w/in a package

1. Put each dataset in a data frame [or a data table? Data.table package is supposed to be a lot faster]
2. Put each var in a column

Only one shape of tidy data, any number of steps of messy data, so tidying data often tedious

Often leads to vars that are hidden, need to be extracted

Inconsistency makes it hard to work w/ data

So, tidyr package

Like a successor to reshape2

For making it easier to transform each var to column

Of course, sometimes you don’t have vars and obs in such a straightforward way. Matrices, xml, dates, strings. Same tidy data approach not applicable there, but do try to be consistent

What if you have a mix of object types? E.g. say you want training, test, and CV sets.

List-column is a tool for solving this.

Tibble is a type used for this

You can still store mult kinds of objects together in a single data frame this way. Good for sorting w/o causing problems

Quirks of data frames:

df <- data.frame(xyz = “a”)

What does df$x return?

It returns *a* as a factor

The reason for this: stringsAsFactors and partial name matching (takes x as partial match to xyz)

Base R is more designed for interactively exploring data than for being a great programming language

There’s also a conservative vs utopian challenge. R code written 10 years ago will generally run today. Base R is v on the conservative side, and there are some things that are hard to change w/o breaking hard code.

Tibbles

Like data frames, but more likely to complain if you do something dumb.

Tibbles will not behave like the data frame example above.

Tibbles work better w/ list-columns, too

Recall i() when need to coerce it to do stuff w/ lists, helps in some cases

Tell you the type of columns, which is helpful

Compose simple pieces

General way to solve complex problems is by combining simple, uniform pieces

Think conexes. You can put whatever you like in them, but the interface is consistent

Magrittr is good for composing fcts together. Can write like a pure sequence of statements.

Pipe is unifying way to compose simple pieces across packages

There’s an existing incompatibility b/w ggplot and dplyr, where can’t pipe in.

Ggplot2 also lacks internal consistency in some ways, around where you use “+ geom”

Interestingly, could use pipe w/ ggplot1 in way that can’t w/ ggplot2, due to the +

Shows code example of webscraping mass shooting data, makes a data frame w/ the dates from all the ones recorded in google docs on a particular site

Note that using -> to assign a piped thing to an object is often handy

Embrace Functional Programming

Whatever code solves your problem is good code.

But, relying on for loops keeps you away from learning some techniques that’ll let you solve other problems more easily

For loops emphasize the objects that’re being transformed. Can make it easy to miss the *transformation* that you’re doing, the diff b/w two similar fcts

Purrr package is implementation. Map\_dbl() lets you implement for loops in a much less verbose way – the for loop is hidden, so much easier to see what’s going on. Move up a level of abstraction.

Invoke\_map() is another powerful way to simplify, used for simulations

Example of automated report making using pwalk()

Write for humans

The goal of code is to communicate w/ other humans, not just w/ machines.

“Every project is fundamentally collaborative, with future you if no one else.” – Hadley

Most impt dplyr fcts:

Filter

Mutate

Summarise (ze also works)

Arrange

select

Goal is a pit of success, not a pinnacle of success. Make it easy to fall into success.

“My goal is a pit of success, not a pinnacle of success. A pinnacle you have to work hard to climb. A pit you just fall into.” - Hadley

Install.packages(“tidyverse”) gives you ggplot2, tibble, tidyr, readr, purr, dplyr

Diagram of other “tidyverse” packages – various import tools like jsonlite; forcats and lubridate and stringr and hms in transform; broom nad modelr in model

Uniform way of thinking a/b data to make your life easier.

There’s an excellent website for ggplot2 extensions. Great for finding new ways to visualize. Lots of addl functionality.

Ggrepel is good for neat text labeling – repels labels away from each other

Ggtree for phylo

Mentions temptation to make ggplot3 that’s identical but works w/ a pipe

If you’re going above like 10 layers of pipe, save it as an intermediate variable before doing more so you don’t get confused a/b what you’re trying to do

Shiny is in the Communicate step. And RMD of course.

I heard tonight a few recs for using d3 for interactives – scales better than Shiny. Investigate. I wanna see if can get smthng that scales well w/o learning JS.

Sparklyr – big-data package that uses same stuff as dplyr. Abstracts away operations, lets you work w/ data frame on massive scale in exact same way as would in R.

Want to minimize switching b/w mental frameworks when you don’t have to

R is v good for creating little specific frameworks for specific purposes – but good to add commonality

Ongoing work on making Rstudio work more like a notebook, in that can do things inline more. Like making an RMD document right there in-frame.

Chunk as unit of composition of a data analysis

Also, plans for a new practice to make modeling better

Note that ln() contains an entire copy of the data. There are less memory-intense ways to do this! Is to be fixed.

Key w/ big data is to move no more data around than necessary. Want to send parameterization of a linear model to Spark, not drag the data from Spark into R. And want it to work same way whether you’re doing it in spark or in R. Model-agnostic stuff like dplyr has.

Communication depends on scale you’re going for. 538 or The Upshot require more time, explanation, and context than making smthng for 3 people does.

Rapid iteration for exploration

Forcats is for making factors easier to use