Tabular data management

Jennifer Bryan
RStudio, University of British Columbia

@JennyBryan    @jennybc
data cleaning
data wrangling
descriptive stats
inferential stats
reporting
data cleaning
data wrangling
descriptive stats
inferential stats
reporting
data cleaning
data wrangling
descriptive stats
inferential stats
reporting
programming
difficulty
better exp. design → simpler stats

better data model → simpler analysis
decision fatigue
aggravation
cutting corners

mastery
efficiency
safety
decision fatigue
aggravation
cutting corners

mastery
efficiency
safety

I want this for you!
do first bit of pirates vs ninjas live coding
R objects come in a few flavours
a simple view of simple R objects that will get you pretty far

<table>
<thead>
<tr>
<th>Simple view</th>
<th>Technically correct R view</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>mode</td>
</tr>
<tr>
<td>character</td>
<td>character</td>
</tr>
<tr>
<td>logical</td>
<td>logical</td>
</tr>
<tr>
<td>numeric</td>
<td>numeric</td>
</tr>
<tr>
<td>factor</td>
<td>numeric</td>
</tr>
</tbody>
</table>
R objects come in a few flavours

A simple view of simple R objects that will get you pretty far

<table>
<thead>
<tr>
<th>Simple view</th>
<th>Technically correct R view</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>mode</td>
</tr>
<tr>
<td>character</td>
<td>character</td>
</tr>
<tr>
<td>logical</td>
<td>logical</td>
</tr>
<tr>
<td>numeric</td>
<td>numeric</td>
</tr>
<tr>
<td>factor</td>
<td>numeric</td>
</tr>
</tbody>
</table>
R objects come in a few flavours

A simple view of simple R objects that will get you pretty far

<table>
<thead>
<tr>
<th>Simple view</th>
<th>Technically correct R view</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>mode</td>
</tr>
<tr>
<td>character</td>
<td>character</td>
</tr>
<tr>
<td>logical</td>
<td>logical</td>
</tr>
<tr>
<td>numeric</td>
<td>numeric</td>
</tr>
<tr>
<td>factor</td>
<td>numeric</td>
</tr>
</tbody>
</table>
Collecting things of the same mode? yes → vector no → list

Collecting vectors of the same length? yes → data.frame no →

Collecting vectors of the same mode? yes → matrix no →
#rstats data structures via lego
atomic vectors

logical

factor

integer, double
related vectors of same length?

DATA FRAME!
Sidebar: Google “data rectangling” to see more rstats with lego

```r
minis %>%
  map2(hair, enhair) %>%
  map2(weapons, arm)
```

https://speakerdeck.com/jennybc/data-rectangling
related data frames for one experiment?

**SummarizedExperiment**

- `colData(se)`
- `rowData(se)`
- `rowRanges(se)`
- `subsetByOverlaps(se, roi)`
- `assays(se)`
- `assay(se, n = 2)`
- `assay(subsetByOverlaps(se, roi))`
- `assay(se[, se$dex == "trt"]`)
- `metadata(se)`
- `metadata(se)$modelFormula`
http://tidyverse.org
back to pirates vs ninjas live coding but with the tidyverse
If you can put it in a data frame, DO THAT.

Operate on the data frame holistically.

Pass it to other functions, pref. intact and whole.

Learn how to limit computation to specific rows or columns. Don’t create copies or excerpts lightly.

I recommend tidyverse + tibbles.
Where do tibbles come from?
Import delimited file
read_csv(), read_delim(), read_excel()...

Coerce from something else
as_tibble()  

Assemble from vector parts
tibble(...), enframe(...)  

Grow / modify an existing object
mutate()
Where do data frames come from?
Import delimited file
read.csv(), read.delim(), ...

Coerce from something else
as.data.frame()

Assemble from vector parts
data.frame(…)

Grow / modify an existing object
transform()
BioC’s SummarizedExperiment

Samples (Columns)

Features (Rows)

- colData(se)
- se[, se$dex == "trt"]
- rowRanges(se)
- rowData(se)
- subsetByOverlaps(se, roi)
- assays(se)
- assay(se, n = 2)
- assay(subsetByOverlaps(se, roi))
- assay(se[, se$dex == "trt"])

- metadata(se)
- metadata(se)$modelFormula

Good software: simple, tidy, rich

Acknowledgments

3/7
Common theme between data frames or tibbles and `SummarizedExperiment`:

Keep related things together!!!
Reduces error and tedium over doing this “by hand”

More specialized scope? Like, genomics?
Congrats, you can have more specialized object classes!

Payoffs: validity checking, receptacles to handle data of disparate type/shape, highly customized methods
Tension between data frames or tibbles and BioC / SummarizedExperiment

Under the hood, implemented with fairly different features of the R language

Different mindset:
  general tools, user recombines to fit today’s problem
  vs
  specific tools, developers anticipate the workflows

Not always trivial to move R objects or your brain back and forth
Things you need to know about tibbles:

no partial name matching with `\$`
stringsAsFactors = FALSE
df[, “X1”] will be a tibble, i.e. drop = FALSE
you can print them with wild abandon
no row names
do not munge variable names
will only recycle input of length 1