Introduction to the Tidyverse

Exploring an Opinionated Grammar of R

Nicholas R. Davis

7/29/2019

What is the 'Tidyverse'?

A set of packages developed together following a common set of principles.

- "tidy" data philosophy, where each variable has its own column, each observation has its own row
- code clarity and reproducibility through common functional structure
- use of pipe `%>%` to improve code development and readability

Packages Included

- ☐ `ggplot2`: data visualization
- ☒ `dplyr`: data manipulation
- ☐ `tidyr`: modeling and data management
- ☐ `readr`: open and organize the data
- ☐ `purrr`: code optimization and functional programming
- ☒ `tibble`: alternative to data.frame class
- ☐ `stringr`: functions for working with string data
- ☐ `forcats`: functions for working with factors
- ☒ also, by default includes `magrittr` (source of the pipe operator)

A Few Words of Caution

The Tidyverse can provide a useful set of tools, but…

- it is *not* a perfect solution to all our data problems
- it is *not* always as stable as base-R
- it is *not* used by (or even liked by) everyone
- perhaps most importantly, it is *not* a replacement for base-R

Therefore, do *not* assume…

- that it is always your best choice for building R-scripts
- that everyone will inevitably end up being “tidy”
- that you can avoid learning base-R for general research tasks

Why Be Tidy-literate?

The Tidyverse provides a powerful set of tools for working with data.

- built as a suite of “data science” tools with a focus on importing, manipulating, visualizing data
- fairly easy to mix tidy and non-tidy code/functions
- code clarity (and “piping”) useful as user-generated functions or data management tasks become more complex

Setting Up
(Entering the Tidyverse)

Install the Tidyverse

You can install everything at once (recommended)

```
> install.packages("tidyverse")
```

This package is actually many packages wrapped up together for ease of use.

Access the Tidyverse

Load the Tidyverse

```
> library(tidyverse)
```

```
── Attaching packages ─────────────────────────────────────── tidyverse 1.2.1 ──

✔ ggplot2 3.1.0       ✔ purrr   0.3.2
✔ tibble  2.1.1       ✔ dplyr   0.8.0.1
✔ tidyr   0.8.3       ✔ stringr 1.4.0
✔ readr   1.3.1       ✔ forcats 0.4.0

── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
```

Function Masking and dplyr

What about those conflicts?

- we see that there are two functions in the `dplyr` package which *mask* base-R functions of the same name
- this means if we want to access the base functions instead of the tidy ones, we need to specify the namespace
- we could do this with `base::select()`
- as a general rule, you might want to load the tidyverse *after* all other packages; this will identify the conflicts for you

magrittr
(Piping hot code)

The Pipe Operator

The `%>%` operator (from `magrittr`) has a special purpose.

- takes the object/function call result on the left and "passes" it to the right; it does *not* make assignment by itself
- functions on the right can be passed the left side by adding "." in place of the argument

```
> x <- rnorm(100)
> mean(x)
```

```
[1] -0.01533084
```

```
> x %>% mean(.)
```

```
[1] -0.01533084
```

Pipe Example

```
> # assign Prestige data to object
> prestige.data <- carData::Prestige
> # use pipe to return brief overview after removing NA values
> prestige.data %>% na.omit(.) %>% car::brief(.)
```

```
98 x 6 data.frame (93 rows omitted)
                  education income women prestige census type
                        [n]    [i]   [n]      [n]    [i]  [f]
gov.administrators      13.11  12351 11.16     68.8   1113 prof
general.managers        12.26  25879  4.02     69.1   1130 prof
accountants             12.77   9271 15.70     63.4   1171 prof
. . .
typesetters             10.00   6462 13.58     42.2   9511 bc
bookbinders              8.55   3617 70.87     35.2   9517 bc
```

tibble
(Tidy data frames)

What is a Tibble?

The tidyverse uses tibbles as an alternative to the `data.frame` class.

- tibbles, data frames have many similar properties (rectangular data)
- tibbles are intended to represent the “tidy” data principles by design
- tibbles respond well to `dplyr` data manipulation methods but coerce easily back to `data.frame` as well

Load Data as Tibble

```
> (prestige.data <- prestige.data %>% as_tibble)
```

```
# A tibble: 102 x 6
   education income women prestige census type
       <dbl>  <int> <dbl>    <dbl>  <int> <fct>
 1      13.1  12351 11.2      68.8   1113 prof
 2      12.3  25879  4.02     69.1   1130 prof
 3      12.8   9271 15.7      63.4   1171 prof
 4      11.4   8865  9.11     56.8   1175 prof
 5      14.6   8403 11.7      73.5   2111 prof
 6      15.6  11030  5.13     77.6   2113 prof
 7      15.1   8258 25.6      72.6   2133 prof
 8      15.4  14163  2.69     78.1   2141 prof
 9      14.5  11377  1.03     73.1   2143 prof
10      14.6  11023  0.94     68.8   2153 prof
# … with 92 more rows
```

Properties of Tibbles

The Good:

- automatic “brief” view; just type object name in console
- can be subsetted using all the familiar operators/indexing methods

The Bad:

- it is possible to create column classes which are tidy-specific (via `haven`)
- sometimes older functions cannot directly use tibbles
- no row names allowed!

Coercing Tibbles

It is easy to use tibbles with base-R functions which take arguments of class `data.frame`

- this is because tibbles have multiple class attributes

```
> prestige.data %>% class(.)
```

```
[1] "tbl_df"     "tbl"        "data.frame"
```

- where needed, explicit coercing is simple

```
> prestige.data %>% as.data.frame %>% class
```

```
[1] "data.frame"
```

Example

```
> prestige.data %>%
+   lm(prestige ~ income + education + women, data=.)
```

```
Call:
lm(formula = prestige ~ income + education + women, data = .)

Coefficients:
(Intercept)       income    education        women
  -6.794334     0.001314     4.186637    -0.008905
```

dplyr
(Tidy data management)

Basic dplyr Functionality

There are many useful functions for working with data in this package.

- summarize and group cases
- manipulate cases and variables
- combining and manipulating data sets

Summarize

Suppose we wanted to find the means of a few variables:

```
> prestige.data %>%
+   filter(!is.na(type)) %>%
+   summarise_at(vars(education, income, women, prestige), mean)
```

```
# A tibble: 1 x 4
  education income women prestige
      <dbl>  <dbl> <dbl>    <dbl>
1      10.8  6939.  29.0     47.3
```

Summarize by Group

What about means for each level of the factor 'type'?

```
> prestige.data %>%
+   filter(!is.na(type)) %>%
+   group_by(type) %>%
+   summarise_at(vars(education, income, women, prestige), mean)
```

```
# A tibble: 3 x 5
  type  education income women prestige
  <fct>     <dbl>  <dbl> <dbl>    <dbl>
1 bc         8.36  5374.  19.0     35.5
2 prof      14.1  10559.  25.5     67.8
3 wc        11.0   5052.  52.8     42.2
```

Manipulate Variables

Perhaps we want to create a new variable which is a transformation of `education`:

```
> prestige.data %>%
+   mutate(., educ_deviation =
+            (education - mean(education) ) / sd(education) ) %>%
+   select_at(., vars(education, educ_deviation) ) %>%
+   summary
```

```
   education      educ_deviation
 Min.   : 6.380   Min.   :-1.59726
 1st Qu.: 8.445   1st Qu.:-0.84042
 Median :10.540   Median :-0.07258
 Mean   :10.738   Mean   : 0.00000
 3rd Qu.:12.648   3rd Qu.: 0.69983
 Max.   :15.970   Max.   : 1.91756
```

Additional Resources

On the web:

- https://www.tidyverse.org/

Books:

- Wickham. H. and G. Grolemund. “R for Data Science.” Online: https://r4ds.had.co.nz/

Also see:

- Data management: https://tinyurl.com/data-transform-sheet
- Data import: https://tinyurl.com/data-import-sheet