R Design Patterns, Base-R vs. Tidyverse
With a view toward the teaching of R beginners

Norman Matloff
Dept. of Computer Science, University of California, Davis

August 18, 2022

This document enables the reader to see at a glance the difference between base-R and the tidyverse in common R design settings. I believe the base-R versions are generally simpler, thus more appropriate for R learners.

All examples use R's built-in datasets. After e.g., changing a data frame, it is restored for the next example, e.g. **data(mtcars)**. The examples are presented roughly in order of how often these operations tend to be performed by R users.

As this document is aimed at comparing base-R and the tidyverse in terms of teaching new R learners, advanced functions from either base-R or the tidyverse are excluded here.

More and more examples will be added over time.

Reading a specific cell in a data frame

```
mtcars$mpg[3]
```

```
mtcars %>%
select(mpg) %>%
   filter(row_number() == 3)
```

Comment: The Tidy version could be shortened a bit by using select(mtcars,mpg) instead of mtcars %>% select(mpg), but the latter seems to be the preferred form, e.g. on the official tidyverse page, *https://dplyr.tidyverse.org/*.

Adding a column to a data frame

```
mtcars$hwratio
  <- mtcars$hp / mtcars$wt
```

```
mtcars %>%
mutate(hwratio=hp/wt) -> mtcars
```

Comment: Of course, typically Tidy coders would use <- rather than ->. I feel that the former is more consistent with the “left to right flow” of pipes. But in any case, the point about code complexity is the same either way.

Extracting rows

```
mtc8 <-
   subset(mtcars, cyl==8)
```

```
mtcars %>%
filter(cyl == 8) -> mtc8
```

Mean by group

```
tapply(mtcars$mpg,
   mtcars$cyl, mean)
```

```
mtcars %>%
group_by(cyl) %>%
   summarize(meanMPG =
      mean(mpg,))
```

Row means

```
rowMeans(EuStockMarkets)
```

```
EuStockMarkets %>%
   as.data.frame %>%
   rowwise() %>%
   mutate(m =
      rowMeans(across(everything())))
  %>% select(m)
```

Comment: The row means operation is quite common in R usage. Here the Tidy user must go to far much trouble than in base-R.

Row operations, custom function

```
mM <- function(x)
   max(x) - min(x)
apply(EuStockMarkets,1,mM)
```

```
mM <- function(x)
    max(x) - min(x)
 EuStockMarkets %>%
   as.data.frame %>% rowwise() %>%
   mutate(m =
      mM(across(everything()))) %>%
   select(m)
```

Comment: The **apply()** function is of central importance in traditional R. Here again, the Tidy user must go to much more trouble.

Means, grouped by more than one variable

```
tapply(mtcars$mpg,
   list(mtcars$cyl,
      mtcars$am),
   mean)
```

```
mtcars %>%
   group_by(cyl,am) %>%
   summarize(m = mean(mpg))
```

Comment: There is quite a difference in type of output here:

```
> tapply(mtcars$mpg,list(mtcars$cyl,mtcars$am),mean)
       0        1
4 22.900 28.07500
6 19.125 20.56667
8 15.050 15.40000
> mtcars %>% group_by(cyl,am) %>% summarize(m = mean(mpg))
# Groups:   cyl [3]
    cyl    am     m
  <dbl> <dbl> <dbl>
1     4     0  22.9
2     4     1  28.1
3     6     0  19.1
4     6     1  20.6
5     8     0  15.0
6     8     1  15.4
```

The base-R form returns a 3x2 table, which is often what one needs for reports, research papers and so. The Tidy version is less useful in such contexts.

Quick look at bivariate data

```
plot(x=mtcars$wt,
   y=mtcars$mpg)
```

```
ggplot(data = mtcars,
   aes(x = wt, y = mpg)) +
   geom_point()
```

Binary categorization on a vector

```
NileHiLow <-
  ifelse(Nile >= 1000,
    'high','low')
```

```
Nile %>% as.data.frame %>%
  mutate(
    HighLow = case_when
    (x < 1000~'low',
     x >= 1000~'high')
  ) %>%
  select(HighLow) %>%
  as.vector -> HighLow
```

Comment: The step of conversion back to a vector at the end is needed for many R packages in which vector input is required.

Deleting columns from a data frame

```
mtcars[c('drat','carb')]
  <- NULL
```

```
select(mtcars,-c(drat,carb))
  -> mtcars
```