Advanced Verbs

sunsik

By attaching "all, if, at" at the end of the verb, more complicated but subtle expression can be delievered to R. Verbs in dplyr package whose advanced usage will be covered are following:

- filter
- group_by
- summarize
- mutate
- select

0. Basic vocabularies

(1) arguments

Before we start, let's look into some of the important argument that will be in frequent use:

1. **predicate : designate columns**
   - This is the part that we designate columns to apply a verb by the **feature of their values**.
   - We designate columns to apply a verb by one of **logical vector, function**(wheter it is built-in or manual) and **formula**.
   - Concept on *formula* will be explained shortly.
2. **vars : designate columns**
   - This is the part that we designate columns to apply a verb by the **features of column name**.
   - That is, we can use **column name** or **column index** directly, but we can call **vars() function** to wrap more flexible options in it.
   - Such flexible options are called *select helpers* and the list of them will be presented shortly.
3. **vars_predicate : select rows of the designated columns**
   - This is the part that we select rows of designated columns to apply a verb by specifying **all_vars(), any_vars()** with *expression*.
   - What *expression* is and how to use it also will be explained shortly.
4. **funs : optional**
   - This is the optional part that we can fill if we want to execute certain function to the value of designated column(s).
   - Function to specify in this space can only take designated column(s) as its input. Thus, calculation between column is not allowed.

(2) select helpers

The list of *select helpers* that we can wrap in the vars() function is:

- **starts_with** : designates columns that starts with specified string.
- **ends_with** : designates columns that ends with specified string.
- **contains** : designates columns that contains specified string.
- **matches** : designates columns that matches specified regular expression.
- **num_range** : designates columns that matches prefix and following list of numbers.

(3) etc.,

1) expression

This is **logical statement wrapped by all_vars() or any_vars()** of vars_predicate. Period(.) will refer to the value of designated column(s). all_vars(expression) will filter row(s) whose values in every designated

column(s) satisfy expression. any_vars(expression) will filter row(s) whose values in at least one designated column(s) satisfy expression.

2) formula

Formula can be said as **a manual function that is not declared to the environment**. Indicator of formula is ~, and value of the input variable is called by period(.) too.

Basically:

- **verb_all** designates **every column**
- **verb_if** designates column **by predicate**
- **verb_at** designates column **by vars**

That's it for every concept that needs explanation.

```
library(dplyr)
```

1. filter

```
filter_all(.tbl, .vars_predicate)
filter_if(.tbl, .predicate, .vars_predicate)
filter_at(.tbl, .vars, .vars_predicate)
```

```
# filter rows that at least one of the designated columns(all) satisfy ". == 1.1"
iris %>%
  filter_all(any_vars(. == 1.1))
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
## 1          4.3         3.0          1.1         0.1     setosa
## 2          5.6         2.5          3.9         1.1 versicolor
## 3          5.5         2.4          3.8         1.1 versicolor
## 4          5.1         2.5          3.0         1.1 versicolor
```

```
# designate column that contains at least one 7.9(designated Sepal.Length)
# filter rows that every value of the designated column satisfies ". == 1.1"
iris %>%
  filter_if(~ any(. == 7.9),
            all_vars(. > 7.6))
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
## 1          7.7         3.8          6.7         2.2 virginica
## 2          7.7         2.6          6.9         2.3 virginica
## 3          7.7         2.8          6.7         2.0 virginica
## 4          7.9         3.8          6.4         2.0 virginica
## 5          7.7         3.0          6.1         2.3 virginica
```

```
# designate column that contains ".W"(designated 2 columns)
# filter rows that every value of the designated column satisfies ". > 2.4"
iris %>%
  filter_at(vars(matches("\\.W")),
            all_vars(. > 2.4))
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
## 1          6.3         3.3          6.0         2.5 virginica
## 2          7.2         3.6          6.1         2.5 virginica
## 3          6.7         3.3          5.7         2.5 virginica
```

2. group_by

```
group_by_all(.tbl, .funs = list(), ...)
group_by_if(.tbl, .predicate, .funs = list(), ..., .add = FALSE)
group_by_at(.tbl, .vars, .funs = list(), ..., .add = FALSE)
```

There is optional argument "add". This takes logical value and it decides whether to prevent overwriting(therefore adding) on preceding groups or not.

```
# designates every column to be group variable. Seems somewhat useless.
iris %>%
  group_by_all() %>%
  head()
```

```
## # A tibble: 6 x 5
## # Groups:   Sepal.Length, Sepal.Width, Petal.Length, Petal.Width, Species
## #   [6]
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##          <dbl>       <dbl>        <dbl>       <dbl> <fct>
## 1          5.1         3.5          1.4         0.2 setosa
## 2          4.9         3            1.4         0.2 setosa
## 3          4.7         3.2          1.3         0.2 setosa
## 4          4.6         3.1          1.5         0.2 setosa
## 5          5           3.6          1.4         0.2 setosa
## 6          5.4         3.9          1.7         0.4 setosa
```

```
# designates every column that is factor(designate Species) and apply as.character function to the valu
iris %>%
  group_by_if(is.factor,
              as.character) %>%
  head()
```

```
## # A tibble: 6 x 5
## # Groups:   Species [1]
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##          <dbl>       <dbl>        <dbl>       <dbl> <chr>
## 1          5.1         3.5          1.4         0.2 setosa
## 2          4.9         3            1.4         0.2 setosa
## 3          4.7         3.2          1.3         0.2 setosa
## 4          4.6         3.1          1.5         0.2 setosa
## 5          5           3.6          1.4         0.2 setosa
## 6          5.4         3.9          1.7         0.4 setosa
```

```
# designate the column "Species" and apply str_to_title in stringr package.
iris %>%
  group_by_at("Species",
              list(stringr::str_to_title)) %>%
  summarize(count = n())
```

```
## # A tibble: 3 x 2
##   Species    count
##   <chr>      <int>
## 1 Setosa        50
## 2 Versicolor    50
## 3 Virginica     50
```

3. summarize

```
summarize_all(.tbl, .funs, ...)
summarize_if(.tbl, .predicate, .funs, ...)
summarize_at(.tbl, .vars, .funs, ...)
```

```
# tip : if you are in rush to finish the work, it is safer to define a manual function than struggling
is.int <- function(x) return(all(x %% 1 == 0))
# designate integer columns to form groups and then apply mean to rest of the column.
mtcars %>%
  group_by_if(is.int) %>%
  summarize_all(list(mean)) %>%
  head()
```

```
## # A tibble: 6 x 11
## # Groups:   cyl, hp, vs, am, gear [6]
##     cyl    hp    vs    am  gear  carb   mpg  disp  drat    wt  qsec
##   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1     4    52     1     1     4     2  30.4  75.7  4.93  1.62  18.5
## 2     4    62     1     0     4     2  24.4 147.   3.69  3.19  20
## 3     4    65     1     1     4     1  33.9  71.1  4.22  1.84  19.9
## 4     4    66     1     1     4     1  29.8  78.8  4.08  2.07  19.2
## 5     4    91     0     1     5     2  26   120.   4.43  2.14  16.7
## 6     4    93     1     1     4     1  22.8 108    3.85  2.32  18.6
```

```
# designate column whose name starts with a or v and ends with only one more word(vs, am) as a group co
# Then designate columns whose mean is larger than 20, and apply mean function.
mtcars %>%
  group_by_at(vars(matches("^[av]\\w$"))) %>%
  summarize_if(~ mean(.) > 20,
               list(mean))
```

```
## # A tibble: 4 x 5
## # Groups:   vs [?]
##      vs    am   mpg  disp    hp
##   <dbl> <dbl> <dbl> <dbl> <dbl>
## 1     0     0  15.0 358.  194.
## 2     0     1  19.8 206.  181.
## 3     1     0  20.7 175.  102.
## 4     1     1  28.4  89.8  80.6
```

```
# designate cyl column as group column
# Then designate columns which contains at least 4 characters, and apply mean function.
mtcars %>%
  group_by_at("cyl") %>%
  summarize_at(vars(matches("\\w{4}")),
               list(mean))
```

```
## # A tibble: 3 x 6
##     cyl  disp  drat  qsec  gear  carb
##   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1     4  105.  4.07  19.1  4.09  1.55
## 2     6  183.  3.59  18.0  3.86  3.43
## 3     8  353.  3.23  16.8  3.29  3.5
```

4. mutate

```
mutate_all(.tbl, .funs, ...)
mutate_if(.tbl, .predicate, .funs, ...)
mutate_at(.tbl, .vars, .funs, ...)
```

```
# standardize every value in every column
mtcars %>%
  mutate_all(list(~ (. - mean(.)) / sd(.))) %>%
  head()
```

```
##          mpg        cyl        disp         hp       drat           wt
## 1  0.1508848 -0.1049878 -0.57061982 -0.5350928  0.5675137 -0.610399567
## 2  0.1508848 -0.1049878 -0.57061982 -0.5350928  0.5675137 -0.349785269
## 3  0.4495434 -1.2248578 -0.99018209 -0.7830405  0.4739996 -0.917004624
## 4  0.2172534 -0.1049878  0.22009369 -0.5350928 -0.9661175 -0.002299538
## 5 -0.2307345  1.0148821  1.04308123  0.4129422 -0.8351978  0.227654255
## 6 -0.3302874 -0.1049878 -0.04616698 -0.6080186 -1.5646078  0.248094592
##         qsec         vs         am       gear       carb
## 1 -0.7771651 -0.8680278  1.1899014  0.4235542  0.7352031
## 2 -0.4637808 -0.8680278  1.1899014  0.4235542  0.7352031
## 3  0.4260068  1.1160357  1.1899014  0.4235542 -1.1221521
## 4  0.8904872  1.1160357 -0.8141431 -0.9318192 -1.1221521
## 5 -0.4637808 -0.8680278 -0.8141431 -0.9318192 -0.5030337
## 6  1.3269868  1.1160357 -0.8141431 -0.9318192 -1.1221521
```

```
# designate column in factor(Species) and mutate that column as "species" and "upper"
substr13 <- function(x) return(substr(x, 1, 3))
iris %>%
  mutate_if(is.factor,
            list(species = substr13,
                 upper = toupper)) %>%
  head()
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species species  upper
## 1          5.1         3.5          1.4         0.2  setosa     set SETOSA
## 2          4.9         3.0          1.4         0.2  setosa     set SETOSA
## 3          4.7         3.2          1.3         0.2  setosa     set SETOSA
## 4          4.6         3.1          1.5         0.2  setosa     set SETOSA
## 5          5.0         3.6          1.4         0.2  setosa     set SETOSA
## 6          5.4         3.9          1.7         0.4  setosa     set SETOSA
```

```
# designate columns whose name contains Petal and round standardized value of them
iris %>%
  mutate_at(vars(contains("Petal")),
            list(~ round((. - mean(.)) / sd(.), 4))) %>%
  head()
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5      -1.3358     -1.3111  setosa
## 2          4.9         3.0      -1.3358     -1.3111  setosa
## 3          4.7         3.2      -1.3924     -1.3111  setosa
## 4          4.6         3.1      -1.2791     -1.3111  setosa
## 5          5.0         3.6      -1.3358     -1.3111  setosa
## 6          5.4         3.9      -1.1658     -1.0487  setosa
```

5. select

In this family especially, funs are specified to apply functions to column's name, not value.

```
select_all(.tbl, .funs = list(), ...)
select_if(.tbl, .predicate, .funs = list(), ...)
select_at(.tbl, .vars, .funs = list(), ...)
```

```
# this may be useful if we want to apply certain function to every column's name...
iris %>%
  select_all(toupper) %>%
  head()
```

```
##   SEPAL.LENGTH SEPAL.WIDTH PETAL.LENGTH PETAL.WIDTH SPECIES
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

```
# select columns that will have only 2 levels if converted to factor
mtcars %>%
  select_if(~ nlevels(as.factor(.)) == 2) %>%
  head()
```

```
##                   vs am
## Mazda RX4          0  1
## Mazda RX4 Wag      0  1
## Datsun 710         1  1
## Hornet 4 Drive     1  0
## Hornet Sportabout  0  0
## Valiant            1  0
```

```
set.seed(2007)
randomdata <- as.data.frame(matrix(rnorm(10 * 6), nrow = 10))
names(randomdata) <- paste0("column", 1:ncol(randomdata))
str(randomdata)
```

```
## 'data.frame':    10 obs. of  6 variables:
##  $ column1: num  0.4595 -1.2636 -0.215 1.5681 -0.0729 ...
##  $ column2: num  0.91 -0.66 0.558 -2.206 -0.231 ...
##  $ column3: num  0.533 0.41 0.109 -1.506 0.809 ...
##  $ column4: num  -0.5011 -0.7352 0.2565 -1.1163 0.0905 ...
##  $ column5: num  0.1348 -1.2081 -1.6041 0.0427 -0.7588 ...
##  $ column6: num  -0.158 0.867 -1.751 -1.302 -0.472 ...
```

```
# select columns that number after prefix(column) is one of 1, 3, 5
randomdata %>%
  select_at(vars(num_range("column", c(1, 3, 5)))) %>%
  head(5)
```

```
##       column1    column3     column5
## 1  0.45953516  0.5331768  0.13475436
## 2 -1.26355380  0.4096190 -1.20807348
## 3 -0.21499616  0.1090323 -1.60414221
## 4  1.56808839 -1.5063224  0.04269577
## 5 -0.07289493  0.8087937 -0.75880687
```