

Factors

Factors

A **factor** is a special character vector where the elements have pre-defined groups or 'levels'. You can think of these as qualitative or categorical variables:

```
x <- c("yellow", "red", "red", "blue", "yellow", "blue")  
class(x)
```

```
## [1] "character"
```

```
x_fact <- factor(x) # factor() is a function  
class(x_fact)
```

```
## [1] "factor"
```

Factors

Factors have **levels** (character types do not).

```
x
## [1] "yellow" "red"    "red"    "blue"   "yellow" "blue"

x_fact
## [1] yellow red    red    blue   yellow blue
## Levels: blue red yellow
```

Note that levels are, by default, in **alphanumerical** order.

Factors

Extract the levels of a factor vector using `levels()`:

```
levels(x_fact)
```

```
## [1] "blue"  "red"   "yellow"
```

forcats package

A package called forcats is really helpful for working with factors.



factor() vs as_factor()

`factor()` is from base R and `as_factor()` is from `forcats`

Both can change a variable to be of class factor.

- `factor()` will order **alphanumerically** unless told otherwise.
- `as_factor()` will order by **first appearance** unless told otherwise.

If you are assigning your levels manually either function is fine!

as_factor() function

```
x <- c("yellow", "red", "red", "blue", "yellow", "blue")
x_fact_2 <- as_factor(x)
x_fact_2
```

```
## [1] yellow red    red    blue   yellow blue
## Levels: yellow red blue
```

```
# Compare to factor() method:
x_fact
```

```
## [1] yellow red    red    blue   yellow blue
## Levels: blue red yellow
```

A Factor Example

We will use a slightly different version of the data on heat-related visits to the ER from the State of Colorado.

For today, we are looking at data that reports ER visits by age category.

```
er_visits_age <- read_csv("https://daseh.org/data/CO_ER_heat_visits_by_age.csv")

## Rows: 60 Columns: 6
## — Column specification —————
## Delimiter: ","
## chr (1): age
## dbl (5): year, rate, lower95cl, upper95cl, visits
##
## [ Use `spec()` to retrieve the full column specification for this data.
## [ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```


The data

```
head(er_visits_age)
```

```
## # A tibble: 6 × 6
##   year age          rate lower95cl upper95cl visits
##   <dbl> <chr>      <dbl>      <dbl>      <dbl>   <dbl>
## 1  2011 0-4 years    3.52        1.82        6.16     12
## 2  2011 15-34 years 7.34        5.95        8.74    106
## 3  2011 35-64 years 5.84        4.80        6.88    121
## 4  2011 5-14 years  5.20        3.50        6.90     36
## 5  2011 65+ years  8.34        5.98       10.7     48
## 6  2012 0-4 years    3.58        1.85        6.25     12
```

Notice that `age` is a `chr` variable. This indicates that the values are **character** strings.

R does not realize that there is any order related to the `AGE` values. It will assume that it is **alphanumeric** (for numbers, this means ascending order).

However, we know that the order is: **0-4 years old, 5-14 years old, 15-34 years old, 35-64 years old, and 65+ years old.**

For the next steps, let's take a subset of data.

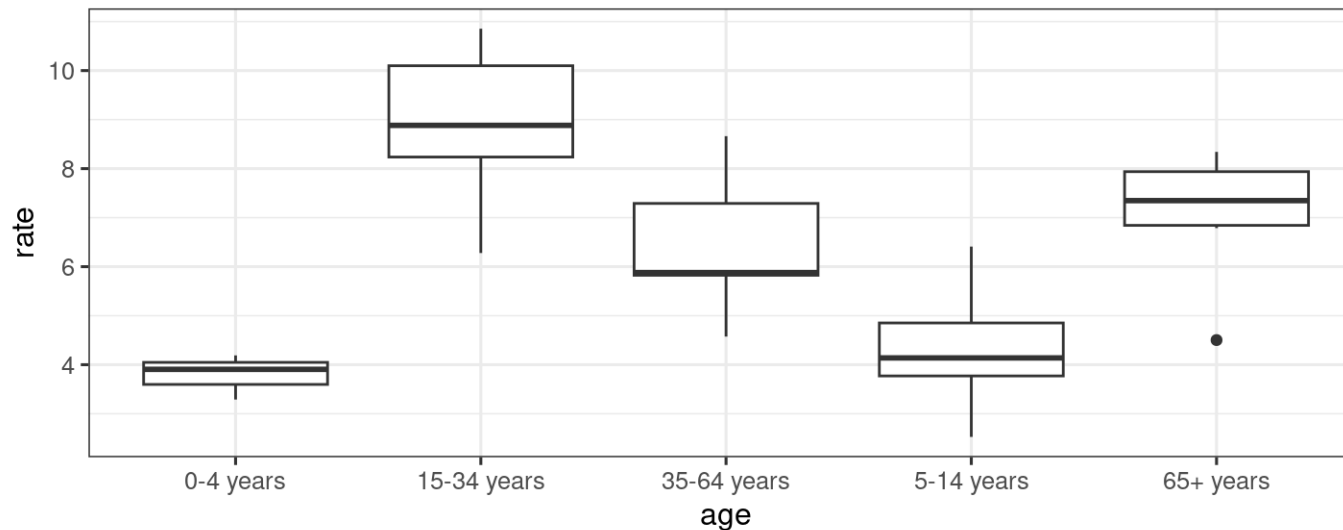
Use `set.seed()` to take the same random sample each time.

```
set.seed(123)  
er_visits_age_subset <- slice_sample(er_visits_age, n = 32)
```

Plot the data

Let's make a plot first.

```
er_visits_age_subset %>%  
  ggplot(mapping = aes(x = age, y = rate)) +  
  geom_boxplot() +  
  theme_bw(base_size = 12) # make all labels size 12
```



OK this is very useful, but it is a bit difficult to read. We expect the values to be plotted by the order that we know, not by alphabetical order.

Change to factor

Currently `age` is class `character` but let's change that to class `factor` which allows us to specify the levels or order of the values.

```
er_visits_age_fct <-  
  er_visits_age_subset %>%  
  mutate(age = factor(age,  
    levels = c("0-4 years old", "5-14 years old", "15-34 years old", "35-64 year  
  ))
```

```
er_visits_age_fct %>%  
  pull(age) %>%  
  levels()
```

```
## [1] "0-4 years old"      "5-14 years old"    "15-34 years old"  "35-64 years old"  
## [5] "65+ years old"
```

Change to a factor

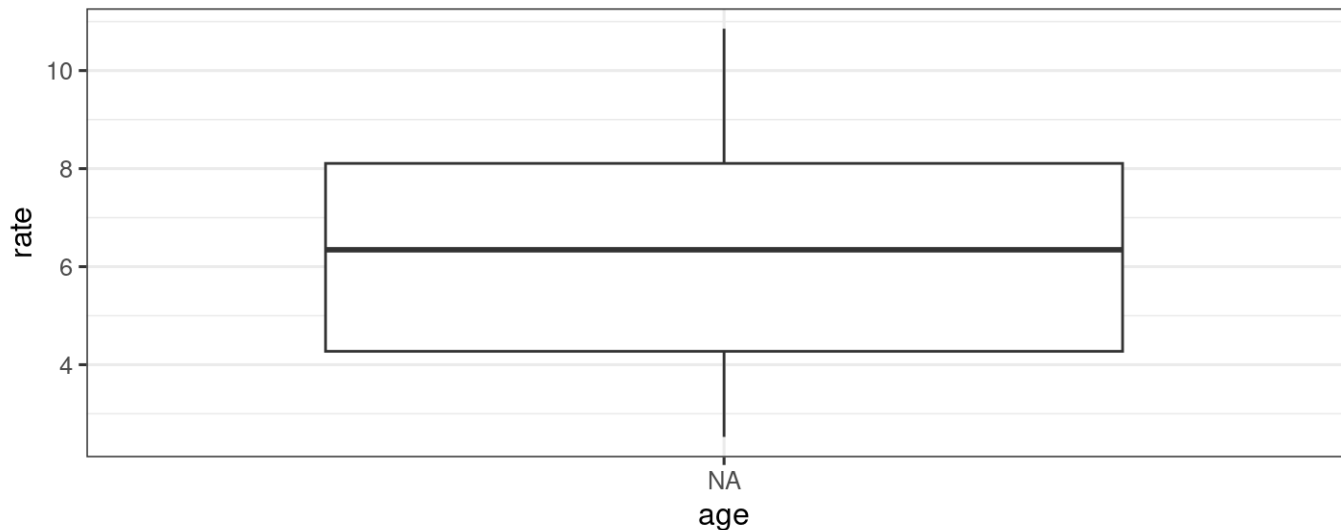
```
head(er_visits_age_fct)
```

```
## # A tibble: 6 × 6
##   year age    rate lower95cl upper95cl visits
##   <dbl> <fct> <dbl>    <dbl>    <dbl>    <dbl>
## 1  2017 <NA>    3.29     1.64     5.89     11
## 2  2013 <NA>    4.50     2.86     6.14     29
## 3  2021 <NA>    NA       NA       NA       NA
## 4  2013 <NA>    5.51     3.78     7.23     39
## 5  2011 <NA>    5.84     4.80     6.88    121
## 6  2019 <NA>    8.34     6.94     9.73    137
```

Plot again

Now let's make our plot again:

```
er_visits_age_fct %>%  
  ggplot(mapping = aes(x = age, y = rate)) +  
  geom_boxplot() +  
  theme_bw(base_size = 12)
```



Now that's more like it! Notice how the data is automatically plotted in the order we would like.

What about if we `arrange()` the data by age?

Character data is arranged alphabetically (if letters) or by ascending first number (if numbers).

```
er_visits_age_subset %>%  
  arrange(age)
```

```
## # A tibble: 32 × 6  
##   year age      rate lower95cl upper95cl visits  
##   <dbl> <chr>    <dbl>    <dbl>    <dbl>    <dbl>  
## 1  2017 0-4 years   3.29     1.64     5.89     11  
## 2  2021 0-4 years   NA        NA        NA        NA  
## 3  2016 0-4 years   4.19     2.29     7.03     14  
## 4  2018 0-4 years   3.91     2.08     6.68     13  
## 5  2019 15-34 years  8.34     6.94     9.73    137  
## 6  2018 15-34 years 10.1      8.60    11.7    165  
## 7  2022 15-34 years 10.0      8.52    11.6    167  
## 8  2016 15-34 years 10.9      9.23    12.5    171  
## 9  2012 15-34 years  8.88     7.36    10.4    130  
## 10 2014 15-34 years  6.28     5.02     7.54     95  
## #   22 more rows
```

Notice that the order is not what we would hope for!

Arranging Factors

Factor data is arranged by level.

```
er_visits_age_fct %>%  
  arrange(age)
```

```
## # A tibble: 32 × 6  
##   year age    rate lower95cl upper95cl visits  
##   <dbl> <fct> <dbl>    <dbl>    <dbl>    <dbl>  
## 1  2017 <NA>   3.29     1.64     5.89     11  
## 2  2013 <NA>   4.50     2.86     6.14     29  
## 3  2021 <NA>   NA        NA        NA        NA  
## 4  2013 <NA>   5.51     3.78     7.23     39  
## 5  2011 <NA>   5.84     4.80     6.88    121  
## 6  2019 <NA>   8.34     6.94     9.73    137  
## 7  2020 <NA>   8.02     6.14     9.90     70  
## 8  2019 <NA>   7.19     6.07     8.30    159  
## 9  2018 <NA>  10.1     8.60    11.7    165  
## 10 2022 <NA>  10.0     8.52    11.6    167  
## #   22 more rows
```

Nice! Now this is what we would want!

Making tables with characters

Tables grouped by a character are arranged alphabetically (if letters) or by ascending first number (if numbers).

```
er_visits_age_subset %>%  
  group_by(age) %>%  
  summarize(total_visits = sum(visits, na.rm = T))
```

```
## # A tibble: 5 × 2  
##   age          total_visits  
##   <chr>          <dbl>  
## 1 0-4 years           38  
## 2 15-34 years        986  
## 3 35-64 years        983  
## 4 5-14 years         215  
## 5 65+ years          296
```

Making tables with factors

Tables grouped by a factor are arranged by level.

```
er_visits_age_fct %>%  
  group_by(age) %>%  
  summarize(total_visits = sum(visits, na.rm = T))
```

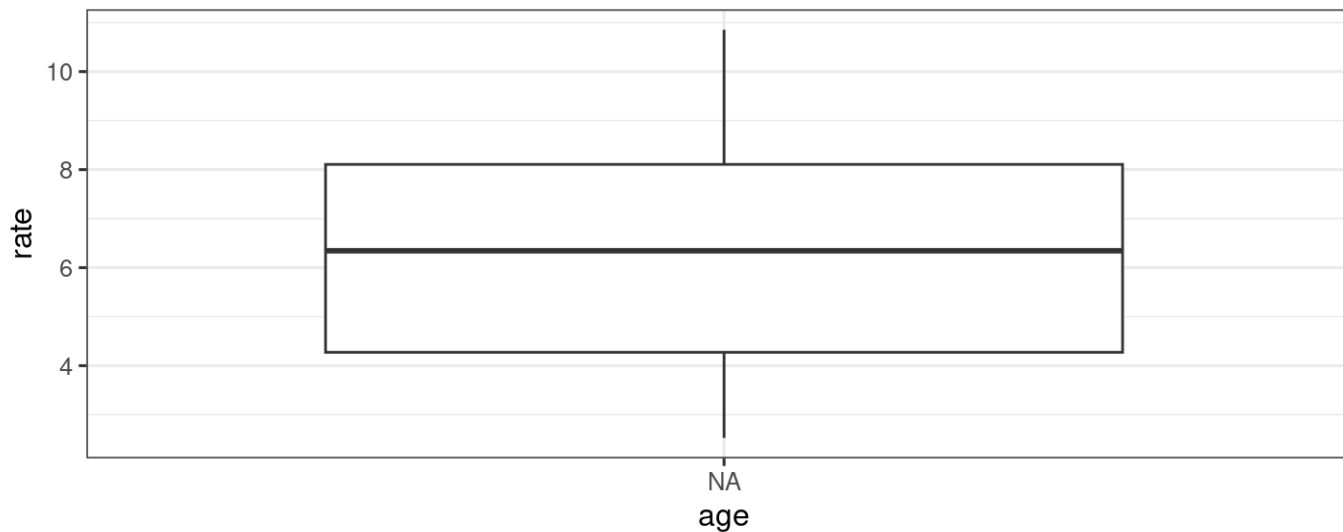
```
## # A tibble: 1 × 2  
##   age    total_visits  
##   <fct>         <dbl>  
## 1 <NA>          2518
```

forcats for ordering

What if we wanted to order **age** by increasing **rate**?

```
library(forcats)
```

```
er_visits_age_fct %>%  
  ggplot(mapping = aes(x = age, y = rate)) +  
  geom_boxplot() +  
  theme_bw(base_size = 12)
```



This would be useful for identifying easily which age group to focus on.

forcats for ordering

We can order a factor by another variable by using the `fct_reorder()` function of the `forcats` package.

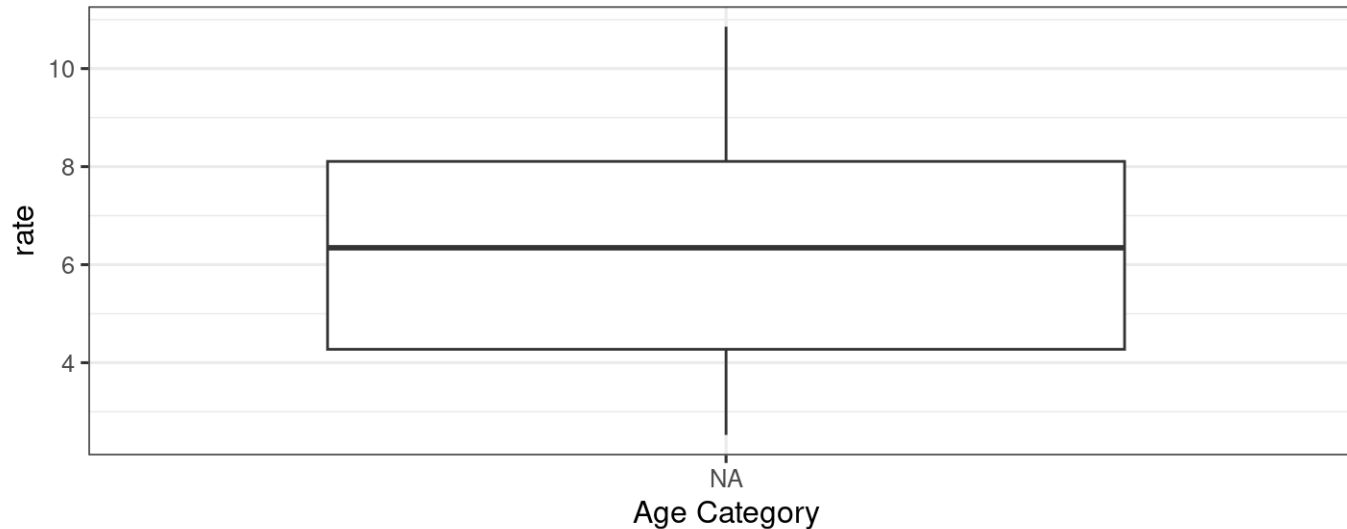
```
fct_reorder({column getting changed}, {guiding column}, {summarizing function})
```

forcats for ordering

We can order a factor by another variable by using the `fct_reorder()` function of the `forcats` package.

```
library(forcats)
```

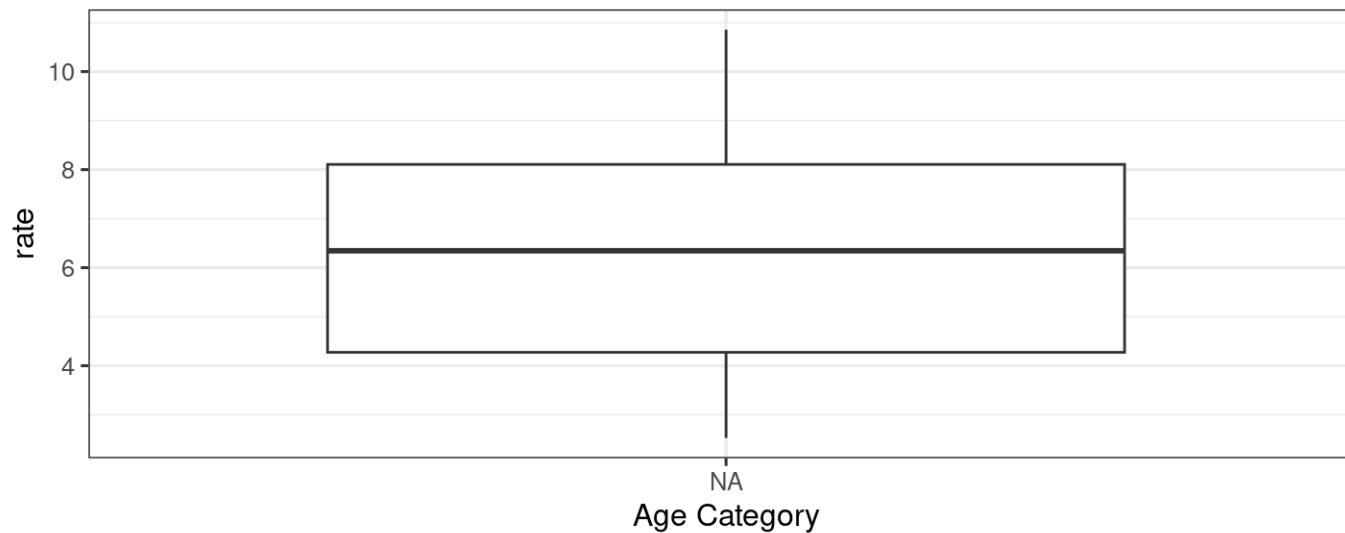
```
er_visits_age_fct %>%  
  ggplot(mapping = aes(x = fct_reorder(age, rate, mean), y = rate)) +  
  geom_boxplot() +  
  labs(x = "Age Category") +  
  theme_bw(base_size = 12)
```



forcats for ordering.. with `.desc` = argument

```
library(forcats)
```

```
er_visits_age_fct %>%  
  ggplot(mapping = aes(x = fct_reorder(age, rate, mean, .desc = TRUE), y = rate)) +  
  geom_boxplot() +  
  labs(x = "Age Category") +  
  theme_bw(base_size = 12)
```



forcats for ordering... can be used to sort datasets

```
er_visits_age_fct %>% pull(age) %>% levels() # By year order
```

```
## [1] "0-4 years old"    "5-14 years old"   "15-34 years old"  "35-64 years old"  
## [5] "65+ years old"
```

```
er_visits_age_fct <- er_visits_age_fct %>%  
  mutate(  
    age = fct_reorder(age, rate, mean)  
  )
```

```
er_visits_age_fct %>% pull(age) %>% levels() # by increasing mean visits
```

```
## [1] "0-4 years old"    "5-14 years old"   "15-34 years old"  "35-64 years old"  
## [5] "65+ years old"
```

Checking Proportions with `fct_count()`

The `fct_count()` function of the `forcats` package is helpful for checking that the proportions of each level for a factor are similar. Need the `prop = TRUE` argument otherwise just counts are reported.

```
er_visits_age_fct %>%  
  pull(age) %>%  
  fct_count(prop = TRUE)
```

```
## # A tibble: 6 × 3  
##   f                n      p  
##   <fct>          <int> <dbl>  
## 1 0-4 years old      0      0  
## 2 5-14 years old     0      0  
## 3 15-34 years old    0      0  
## 4 35-64 years old    0      0  
## 5 65+ years old     0      0  
## 6 <NA>             32     1
```


GUT CHECK: Why is it useful to have the factor class as an option?

- A. It helps us check the factual accuracy of our datasets.
- B. It helps us change the order of variables in case the order has meaning.

GUT CHECK: What does the `fct_reorder()` function do?

- A. It helps us reorder a factor based on the values of another variable.
- B. It helps us reorder a factor based on a random change in the order.

Summary

- the factor class allows us to have a different order from alphanumeric for categorical data
- we can change data to be a factor variable using `mutate` and a factor creating function like `factor()` or `as_factor`
- the `as_factor()` is from the `forcats` package (first appearance order by default)
- the `factor()` base R function (alphanumeric order by default)
- with `factor()` we can specify the levels with the `levels` argument if we want a specific order
- the `fct_reorder({variable_to_reorder}, {variable_to_order_by}, {summary function})` helps us reorder a variable by the values of another variable
- arranging, tabulating, and plotting the data will reflect the new order

Lab

- ▯ [Class Website](#)
- ▯ [Lab.](#) ▯ [Day 6 Cheatsheet](#) ▯ [Posit's forcats cheatsheet](#)



Image by [Gerd Altmann](#) from [Pixabay](#)