

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 **alphabetically** 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 data on heat-related visits to the ER from the State of Colorado, separated by age category, for 2011-2022. More on this data can be found here: <https://coepht.colorado.gov/heat-related-illness>

You can download the data from the DaSEH website here:
https://daseh.org/data/CO_ER_heat_visits_by_age_data.csv

This dataset is also available in the **dasehr** package.

We will limit the data to only one of the **gender** categories - we will choose “Both genders” because of data missingness.

```
library(dasehr)
er_visits_age <- CO_heat_ER_byage

#er_visits_age <- read_csv("https://daseh.org/data/CO_ER_heat_visits_by_age_data.csv")

er_visits_age <- er_visits_age %>%
  filter(str_detect(GENDER, "Both genders"))
```


The data

```
head(er_visits_age)
```

```
## # A tibble: 6 × 7
##   YEAR GENDER AGE RATE L95CL U95CL VISITS
##   <dbl> <chr> <chr> <dbl> <dbl> <dbl> <dbl>
## 1  2011 Both genders 0-4 years old 3.52 1.82 6.16 12
## 2  2011 Both genders 15-34 years old 7.34 5.95 8.74 106
## 3  2011 Both genders 35-64 years old 5.84 4.80 6.88 121
## 4  2011 Both genders 5-14 years old 5.20 3.50 6.90 36
## 5  2011 Both genders 65+ years old 8.34 5.98 10.7 48
## 6  2011 Both genders All ages 6.30 5.62 6.99 323
```

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 **alphabetical** (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**, **65+ years old**, and **All ages**.

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

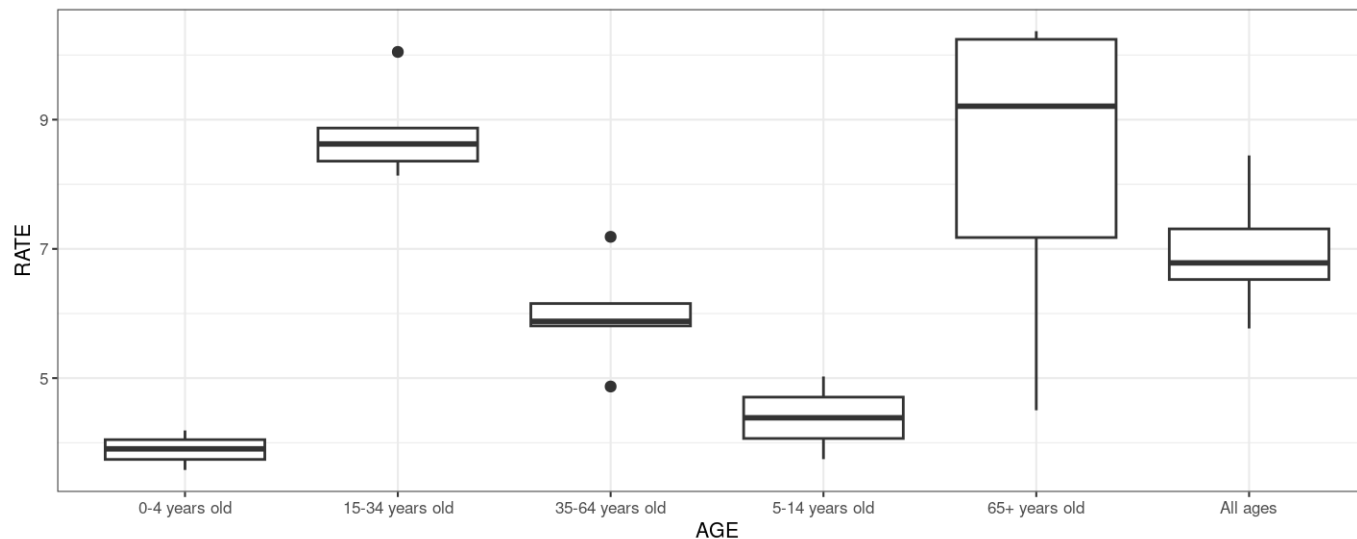
Use `set.seed()` to take the same random sample each time.

```
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 = 8) # make all labels size 8
```



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"     "All ages"
```

Change to a factor

```
head(er_visits_age_fct)
```

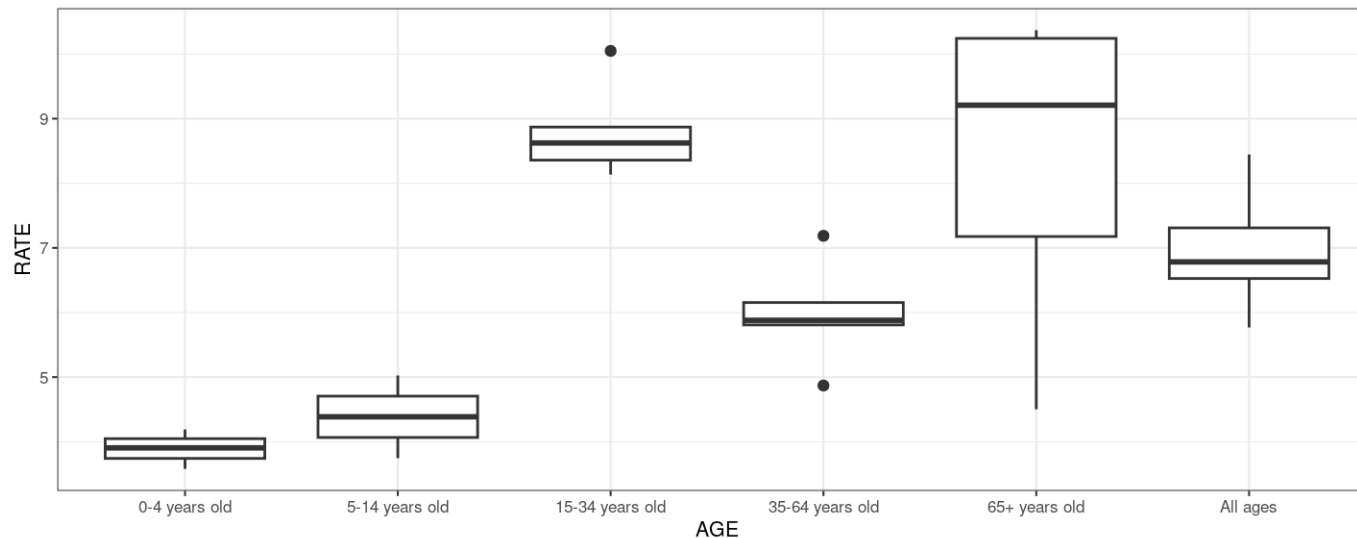
```
## # A tibble: 6 × 7
```

```
##   YEAR GENDER AGE RATE L95CL U95CL VISITS
##   <dbl> <chr> <fct> <dbl> <dbl> <dbl> <dbl>
## 1  2016 Both genders 0-4 years old 4.19 2.29 7.03 14
## 2  2019 Both genders 35-64 years old 7.19 6.07 8.30 159
## 3  2013 Both genders 15-34 years old 8.13 6.69 9.58 121
## 4  2022 Both genders 0-4 years old NA NA NA NA
## 5  2017 Both genders All ages 5.77 5.14 6.40 323
## 6  2019 Both genders 15-34 years old 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 = 8)
```



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 grade?

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

```
er_visits_age_subset %>%  
  arrange(AGE)
```

```
## # A tibble: 32 × 7  
##   YEAR GENDER AGE RATE L95CL U95CL VISITS  
##   <dbl> <chr> <chr> <dbl> <dbl> <dbl> <dbl>  
## 1 2016 Both genders 0-4 years old 4.19 2.29 7.03 14  
## 2 2022 Both genders 0-4 years old NA NA NA NA  
## 3 2018 Both genders 0-4 years old 3.91 2.08 6.68 13  
## 4 2015 Both genders 0-4 years old NA NA NA NA  
## 5 2021 Both genders 0-4 years old NA NA NA NA  
## 6 2012 Both genders 0-4 years old 3.58 1.85 6.25 12  
## 7 2020 Both genders 0-4 years old NA NA NA NA  
## 8 2014 Both genders 0-4 years old NA NA NA NA  
## 9 2013 Both genders 15-34 years old 8.13 6.69 9.58 121  
## 10 2019 Both genders 15-34 years old 8.34 6.94 9.73 137  
## # 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 × 7  
##   YEAR GENDER AGE RATE L95CL U95CL VISITS  
##   <dbl> <chr> <fct> <dbl> <dbl> <dbl> <dbl>  
## 1 2016 Both genders 0-4 years old 4.19 2.29 7.03 14  
## 2 2022 Both genders 0-4 years old NA NA NA NA  
## 3 2018 Both genders 0-4 years old 3.91 2.08 6.68 13  
## 4 2015 Both genders 0-4 years old NA NA NA NA  
## 5 2021 Both genders 0-4 years old NA NA NA NA  
## 6 2012 Both genders 0-4 years old 3.58 1.85 6.25 12  
## 7 2020 Both genders 0-4 years old NA NA NA NA  
## 8 2014 Both genders 0-4 years old NA NA NA NA  
## 9 2022 Both genders 5-14 years old 3.75 2.31 5.19 26  
## 10 2015 Both genders 5-14 years old 5.03 3.38 6.67 36  
## # 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: 6 × 2  
##   AGE                total_visits  
##   <chr>              <dbl>  
## 1 0-4 years old      39  
## 2 15-34 years old   831  
## 3 35-64 years old   649  
## 4 5-14 years old    62  
## 5 65+ years old     389  
## 6 All ages         1943
```

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))
```

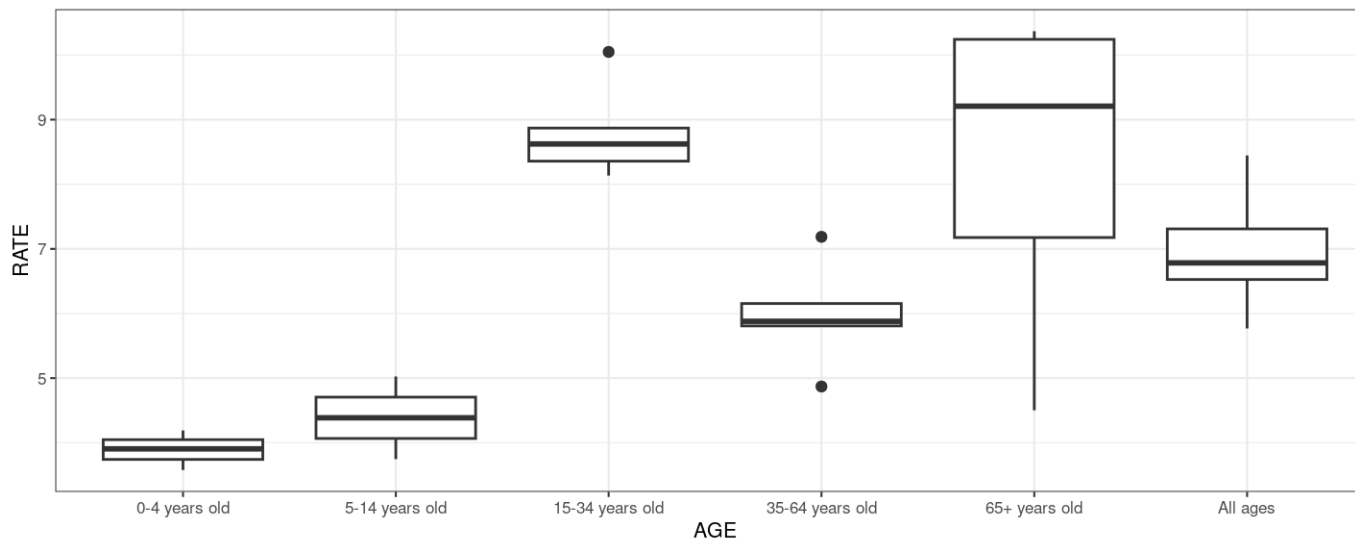
```
## # A tibble: 6 × 2  
##   AGE                total_visits  
##   <fct>              <dbl>  
## 1 0-4 years old      39  
## 2 5-14 years old    62  
## 3 15-34 years old   831  
## 4 35-64 years old   649  
## 5 65+ years old     389  
## 6 All ages         1943
```

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 = 8)
```



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.

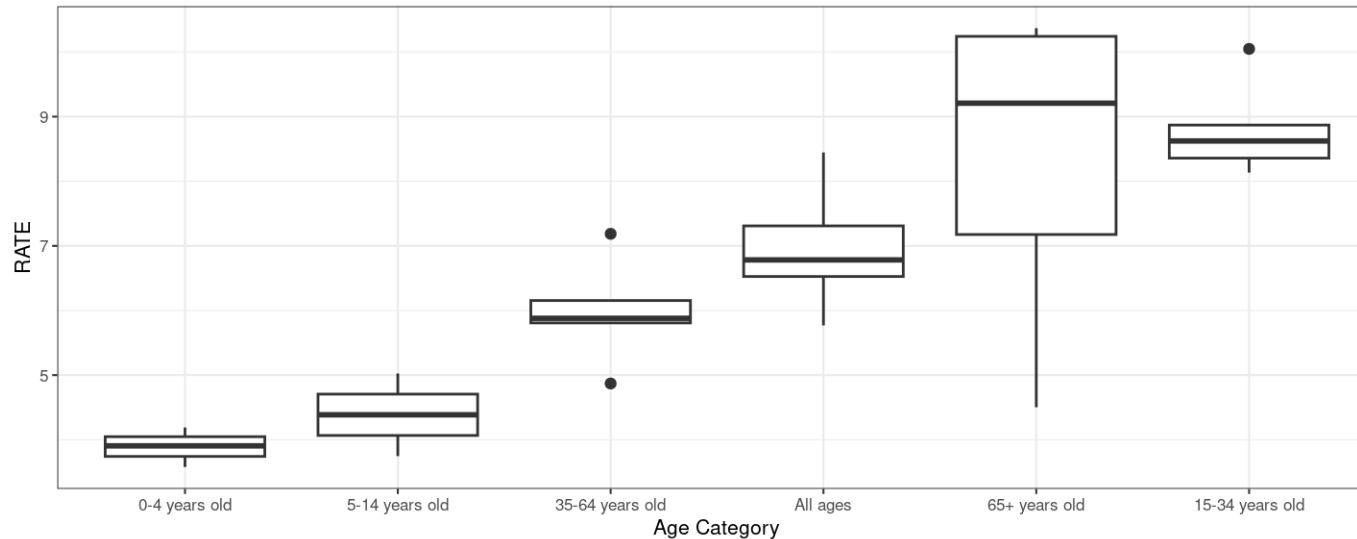
```
fct_reorder({column getting changed}, {guiding column}, {summarizing function})
```

forcats for ordering

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

```
library(forcats)
```

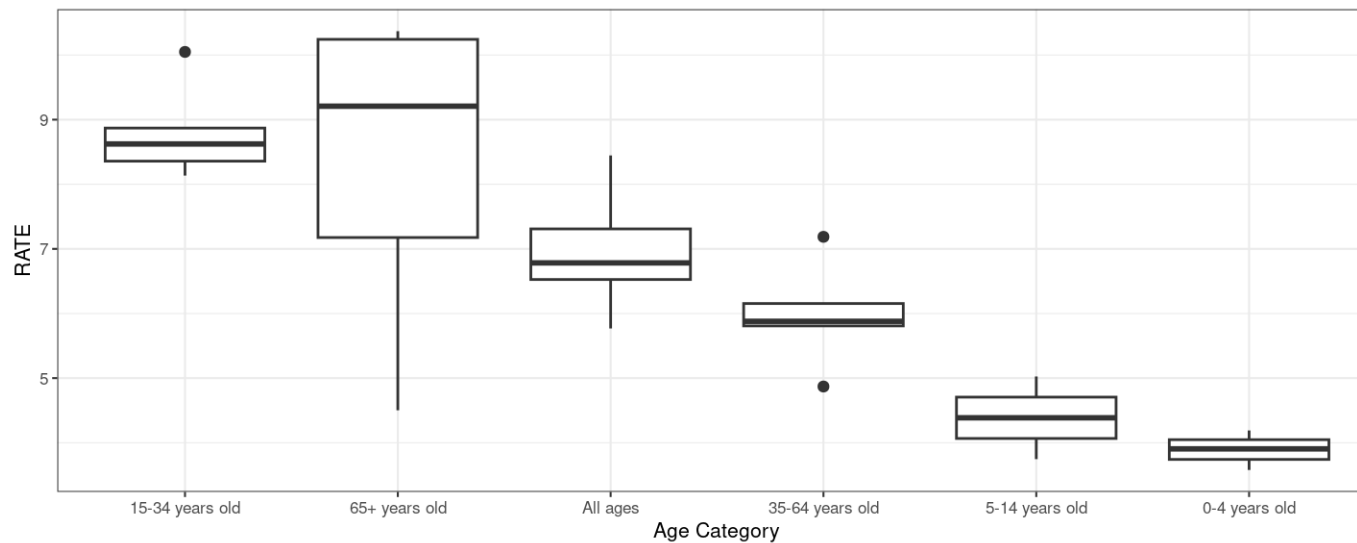
```
er_visits_age_fct %>%  
  ggplot(mapping = aes(x = fct_reorder(AGE, RATE, mean), y = RATE)) +  
  geom_boxplot() +  
  labs(x = "Age Category") +  
  theme_bw(base_size = 8)
```



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 = 8)
```



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"   "All ages"
```

```
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 dropouts
```

```
## [1] "0-4 years old"    "5-14 years old"  "35-64 years old" "All ages"  
## [5] "65+ years old"   "15-34 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      8 0.25  
## 2 5-14 years old     2 0.0625  
## 3 35-64 years old     5 0.156  
## 4 All ages          5 0.156  
## 5 65+ years old      6 0.188  
## 6 15-34 years old     6 0.188
```


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 (alphabetical 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](#)



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