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("red", "red", "blue", "yellow", "blue")
class(x)
```

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

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

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

```
x_fact
```

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

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

Factors

You can learn what are the unique levels of a `factor` vector

```
levels(x_fact)
```

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

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

A Factor Example

First we will create some data about absences of students. We will have information about the number of days absent and the grade for individual students.

- We will use the `tibble()` function to create the data.
- We will use the `sample()` function to create a random sequence of integers from 0 to 7 (for a range of absence values) with replacements for 32 hypothetical students.
- Since there are four grades and 8*4 is 32, we will repeat the 4 grade values 8 times.
- We use the `set.seed()` function so that the random sample from 0 to 7 is the same each time the code is run.

```
set.seed(123)
data_highschool <- tibble(
  absences = sample(0:7, size = 32, replace = TRUE),
  grade = rep(c("Sophomore", "Freshman", "Junior", "Senior"), 8)
)
```

The data

```
head(data_highschool)
```

```
## # A tibble: 6 × 2
##   absences grade
##      <int> <chr>
## 1        6 Sophomore
## 2        6 Freshman
## 3        2 Junior
## 4        5 Senior
## 5        2 Sophomore
## 6        1 Freshman
```

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

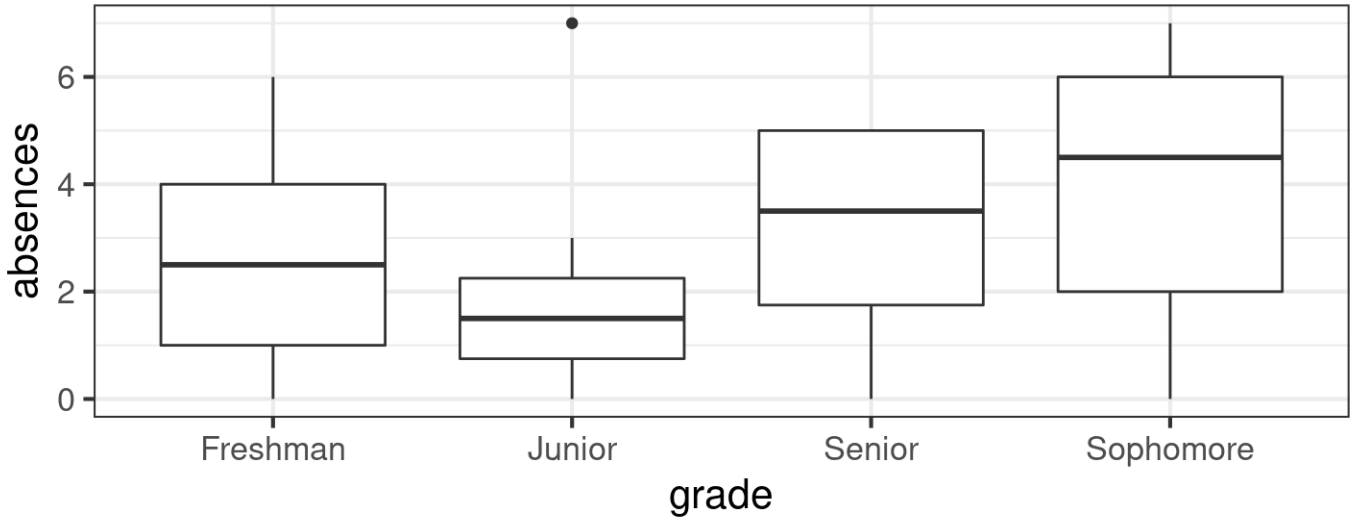
R does not realize that there is any order related to the `grade` values. It will assume that it is **alphabetical**.

However, we know that the order is: **freshman**, **sophomore**, **junior**, **senior**.

Plot the data

Let's make a plot first:

```
data_highschool %>%
  ggplot(mapping = aes(x = grade, y = absences)) +
  geom_boxplot() +
  theme_bw(base_size = 16) # make all labels size 16
```



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

```
levels(pull(data_highschool, grade))
```

```
## NULL
```

```
data_highschool_fct <- data_highschool %>%
  mutate(grade = factor(grade,
    levels = c("Freshman", "Sophomore", "Junior", "Senior")
  ))

levels(pull(data_highschool_fct, grade))
```

```
## [1] "Freshman"  "Sophomore" "Junior"    "Senior"
```

Change to a factor

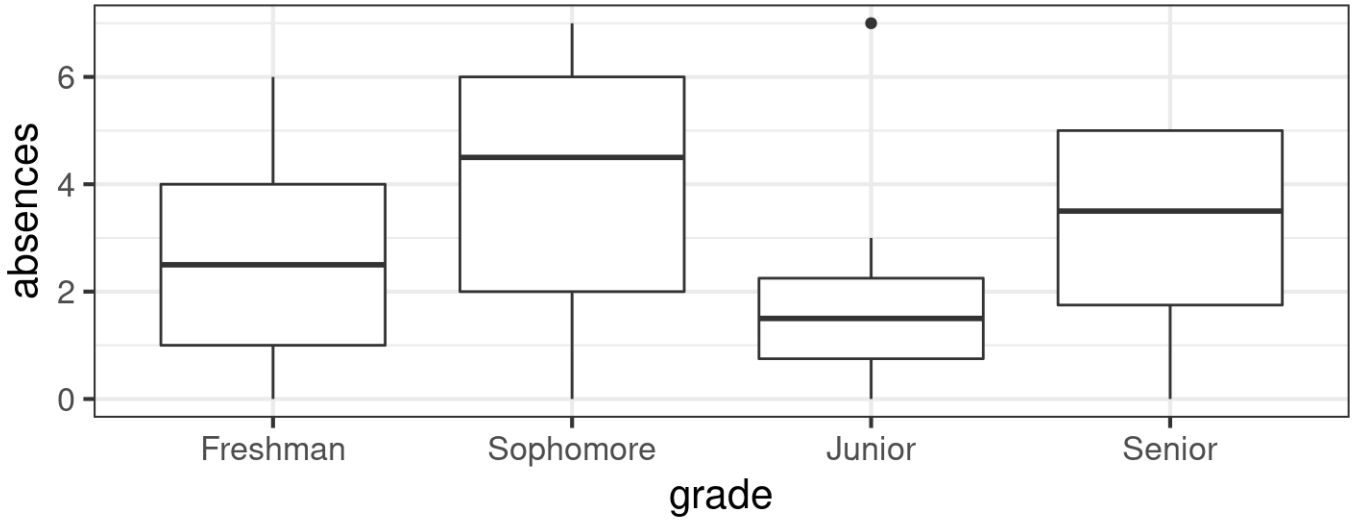
```
head(data_highschool_fct)
```

```
## # A tibble: 6 × 2
##   absences grade
##      <int> <fct>
## 1        6 Sophomore
## 2        6 Freshman
## 3        2 Junior
## 4        5 Senior
## 5        2 Sophomore
## 6        1 Freshman
```

Plot again

Now let's make our plot again:

```
data_highschool_fct %>%
  ggplot(mapping = aes(x = grade, y = absences)) +
  geom_boxplot() +
  theme_bw(base_size = 16)
```



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 ?

```
data_highschool %>% # the data with the character version of grade
  arrange(grade) %>%
  head(n = 19) # print just enough to see the 1st three grades
```

```
## # A tibble: 19 × 2
##    absences grade
##       <int> <chr>
##  1        6 Freshman
##  2        1 Freshman
##  3        4 Freshman
##  4        0 Freshman
##  5        4 Freshman
##  6        3 Freshman
##  7        2 Freshman
##  8        1 Freshman
##  9        2 Junior
## 10        1 Junior
## 11        3 Junior
## 12        1 Junior
## 13        2 Junior
## 14        0 Junior
## 15        7 Junior
## 16        0 Junior
## 17        5 Senior
## 18        5 Senior
## 19        5 Senior
```

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

Arranging Factors

```
data_highschool_fct %>% # the data with the factor version of grade
  arrange(grade) %>%
  head(19)
```

```
## # A tibble: 19 × 2
##    absences grade
##       <int> <fct>
##  1        6 Freshman
##  2        1 Freshman
##  3        4 Freshman
##  4        0 Freshman
##  5        4 Freshman
##  6        3 Freshman
##  7        2 Freshman
##  8        1 Freshman
##  9        6 Sophomore
## 10        2 Sophomore
## 11        2 Sophomore
## 12        5 Sophomore
## 13        7 Sophomore
## 14        0 Sophomore
## 15        4 Sophomore
## 16        6 Sophomore
## 17        2 Junior
## 18        1 Junior
## 19        3 Junior
```

Nice! Now this is what we would want!

Making tables with characters

```
data_highschool %>%
  group_by(grade) %>%
  summarize(total_absences = sum(absences))
```

```
## # A tibble: 4 × 2
##   grade     total_absences
##   <chr>              <int>
## 1 Freshman              21
## 2 Junior                16
## 3 Senior                25
## 4 Sophomore             32
```

Making tables with factors

```
data_highschool_fct %>%
  group_by(grade) %>%
  summarize(total_absences = sum(absences))
```

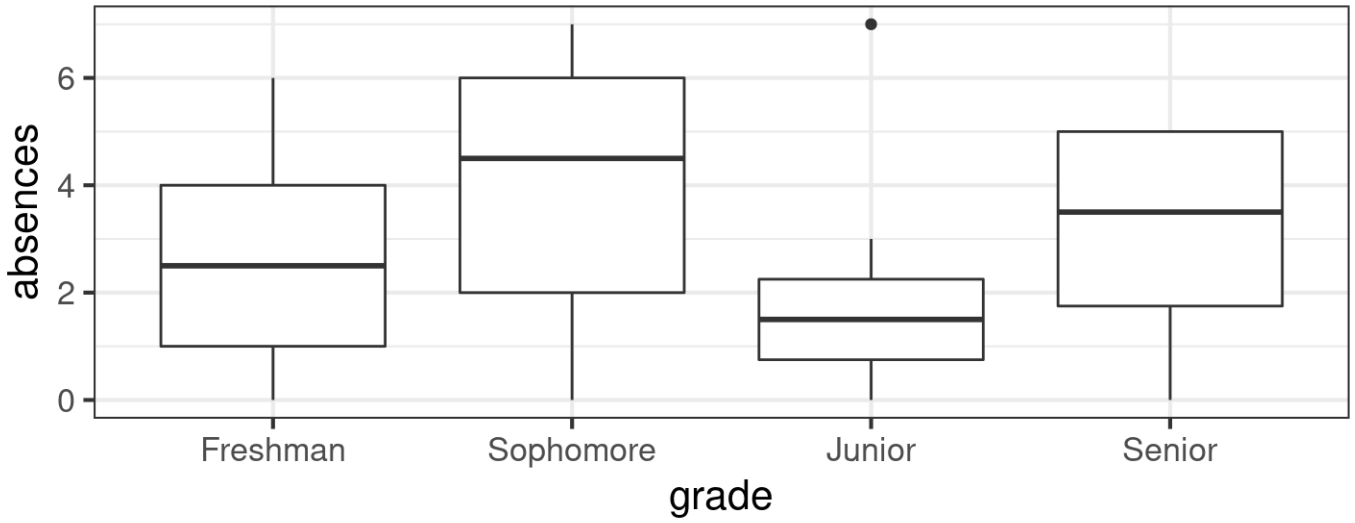
```
## # A tibble: 4 × 2
##   grade     total_absences
##   <fct>              <int>
## 1 Freshman              21
## 2 Sophomore             32
## 3 Junior                16
## 4 Senior                25
```

Here we see that the sum is calculated in the order we would like only for the version of the data that has absences coded as a factor!

`forcats` for ordering

What if we wanted to order `grade` by the amount of `absences`?

```
library(forcats)

data_highschool_fct %>%
  ggplot(mapping = aes(x = grade, y = absences)) +
  geom_boxplot() +
  theme_bw(base_size = 16)
```

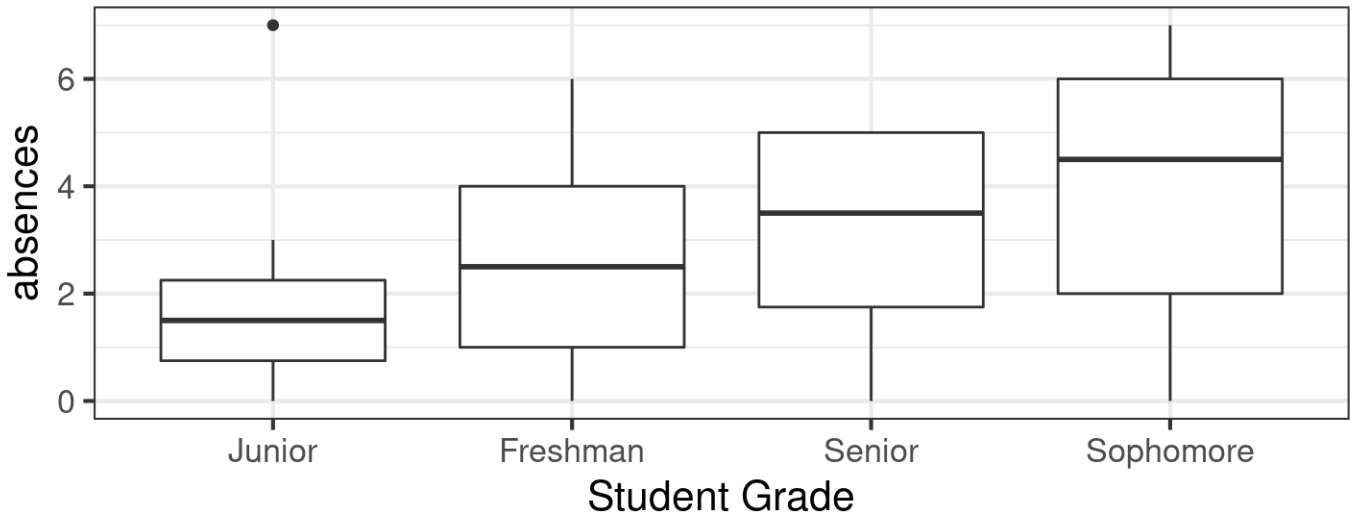


This would be useful for identifying easily which grade to focus on.

forcats for ordering

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

```
library(forcats)

data_highschool_fct %>%
  ggplot(mapping = aes(x = forcats::fct_reorder(grade, absences), y = absences)) +
  geom_boxplot() +
  labs(x = "Student Grade") +
  theme_bw(base_size = 16) # make all labels size 16
```



Adding another variable

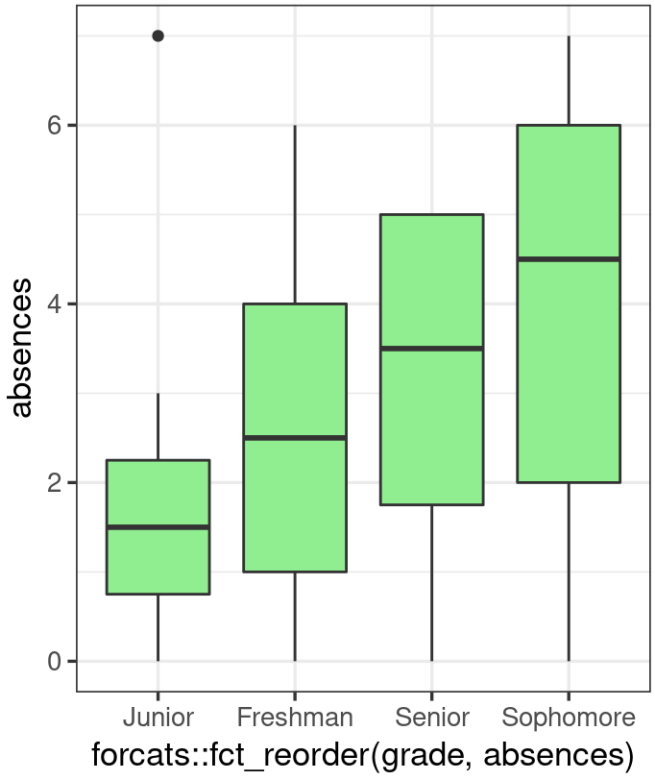
Let's say that we also want to assess which grade has the most incidences of being tardy (another word for late) to class. - Now we will add another simulated variable of random values from 0 to 7 and of 32 values total. - We set a seed again so that our results will be consistent each time we run this code.

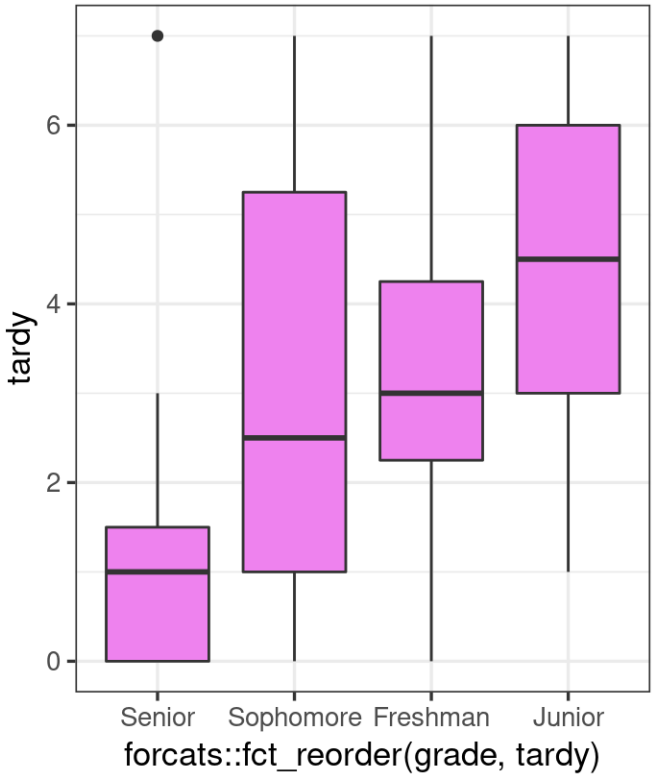
```
set.seed(1956)
data_highschool_fct <-
  data_highschool_fct %>%
  mutate("tardy" = sample(0:7, size = 32, replace = TRUE))

head(data_highschool_fct)
```

```
## # A tibble: 6 × 3
##   absences grade     tardy
##      <int> <fct>     <int>
## 1        6 Sophomore     1
## 2        6 Freshman      3
## 3        2 Junior        5
## 4        5 Senior        1
## 5        2 Sophomore     4
## 6        1 Freshman      0
```

Plotting new variable

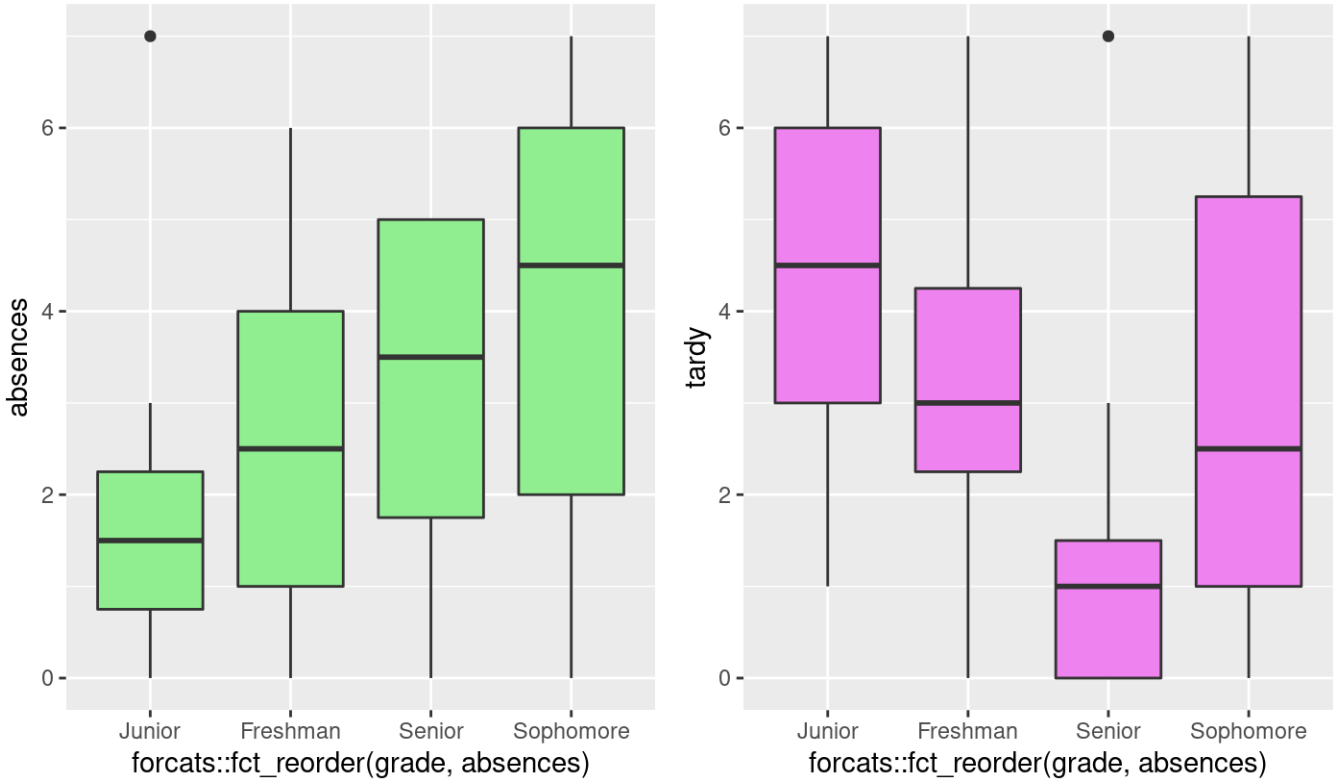
Now let's plot each of our variables of interest (absences and tardy) on the y axis and grade on the x axis. Let's arrange grade by the amount of each.





Plot with more reordering

The last plot is informative, but what if we are mostly interested in absences and we are secondarily interested in tardiness. Then it might help to order grade for both plots by the amount of absences for each grade.



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.

```
data_highschool_fct %>%
  pull(grade) %>%
  fct_count(prop = TRUE)
```

```
## # A tibble: 4 × 3
##   f             n     p
##   <fct>     <int> <dbl>
## 1 Freshman      8  0.25
## 2 Sophomore     8  0.25
## 3 Junior        8  0.25
## 4 Senior        8  0.25
```

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` , the `as_factor()` (of `forcats` package) or `factor()` function and specifying the levels with the `levels` argument
- the `fct_reorder({variable_to_reorder}, {variable_to_order_by})` 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

The End