Intro to R

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 student dropouts from the State of California during the 2016-2017 school year. More on this data can be found here: https://www.cde.ca.gov/ds/ad/filesdropouts.asp

To preserve school anonymity, “CDS_CODE” is used in place of the individual school’s name.

You can download the data from the JHU website here: http://jhudatascience.org/intro_to_r/data/dropouts.txt

```
dropouts <- read_delim("http://jhudatascience.org/intro_to_r/data/dropouts.txt", delim = "\t")
```

```
## Rows: 59599 Columns: 20
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: "\t"
## chr  (2): CDS_CODE, GENDER
## dbl (18): ETHNIC, E7, E8, E9, E10, E11, E12, EUS, ETOT, D7, D8, D9, D10, D11...
##
##   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.
```

```
dropouts
```

```
## # A tibble: 59,599 × 20
##    CDS_CODE  ETHNIC GENDER    E7    E8    E9   E10   E11   E12   EUS  ETOT    D7
##    <chr>      <dbl> <chr>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
##  1 01100170…      1 M          0     0     0     0     1     1     0     2     0
##  2 01100170…      1 F          0     0     1     0     2     0     0     3     0
##  3 01100170…      2 M          0     0     0     0     0     1     0     1     0
##  4 01100170…      2 F          0     0     2     2     2     1     0     7     0
##  5 01100170…      3 M          0     0     0     1     0     0     0     1     0
##  6 01100170…      3 F          0     0     1     1     2     0     0     4     0
##  7 01100170…      4 M          0     0     0     1     0     0     0     1     0
##  8 01100170…      5 M          0     0    31    32    17    22     0   102     0
##  9 01100170…      5 F          0     0    26    34    30    20     0   110     0
## 10 01100170…      6 M          0     0    19    20    17    13     0    69     0
## # … with 59,589 more rows, and 8 more variables: D8 <dbl>, D9 <dbl>, D10 <dbl>,
```

Preparing the data

Aggregate (sum) across ethnicity and gender:

```
dropouts <-
  dropouts %>%
  group_by(CDS_CODE) %>%
  summarize(
    Freshman = sum(D9),
    Sophomore = sum(D10),
    Junior = sum(D11),
    Senior = sum(D12)
  )
dropouts
```

```
## # A tibble: 5,507 × 5
##    CDS_CODE       Freshman Sophomore Junior Senior
##    <chr>             <dbl>     <dbl>  <dbl>  <dbl>
##  1 01100170112607        1         1      0      1
##  2 01100170123968        0         0      0      0
##  3 01100170130401        3         5     12     24
##  4 01100170130419        1         2     13     36
##  5 01100170131581        0         0      0      0
##  6 01100176002000        0         0      0      0
##  7 01316090131755        0         0      0      0
##  8 01316170131763        0         0      0      2
##  9 01611190000001        0         1      0      1
## 10 01611190106401        0         0      0      0
## # … with 5,497 more rows
```

Preparing the data

Pivot to long format:

```
dropouts <-
  dropouts %>%
  pivot_longer(
    !CDS_CODE,
    names_to = "grade",
    values_to = "n_dropouts"
  )
dropouts
```

```
## # A tibble: 22,028 × 3
##    CDS_CODE       grade     n_dropouts
##    <chr>          <chr>          <dbl>
##  1 01100170112607 Freshman           1
##  2 01100170112607 Sophomore          1
##  3 01100170112607 Junior             0
##  4 01100170112607 Senior             1
##  5 01100170123968 Freshman           0
##  6 01100170123968 Sophomore          0
##  7 01100170123968 Junior             0
##  8 01100170123968 Senior             0
##  9 01100170130401 Freshman           3
## 10 01100170130401 Sophomore          5
## # … with 22,018 more rows
```

The data

```
head(dropouts)
```

```
## # A tibble: 6 × 3
##   CDS_CODE       grade     n_dropouts
##   <chr>          <chr>          <dbl>
## 1 01100170112607 Freshman           1
## 2 01100170112607 Sophomore          1
## 3 01100170112607 Junior             0
## 4 01100170112607 Senior             1
## 5 01100170123968 Freshman           0
## 6 01100170123968 Sophomore          0
```

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**.

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

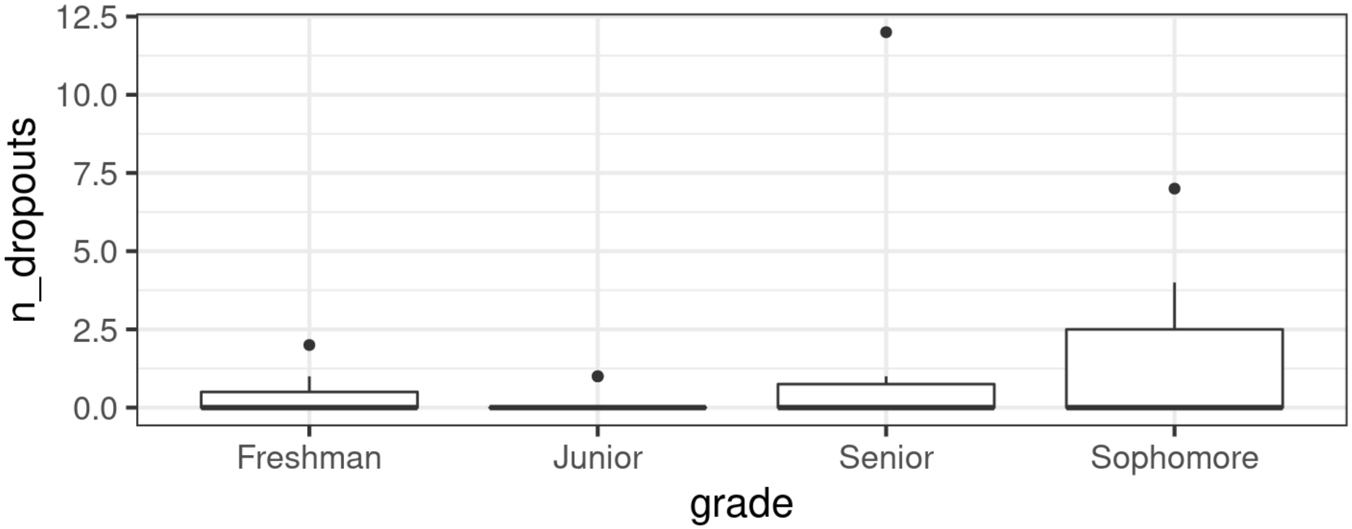
Use `set.seed()` to take the same random sample each time.

```r
set.seed(123)
dropouts_subset <- slice_sample(dropouts, n = 32)
```

Plot the data

Let's make a plot first.

```
dropouts_subset %>%
  ggplot(mapping = aes(x = grade, y = n_dropouts)) +
  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.

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

dropouts_fct %>%
  pull(grade) %>%
  levels()
```

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

Change to a factor

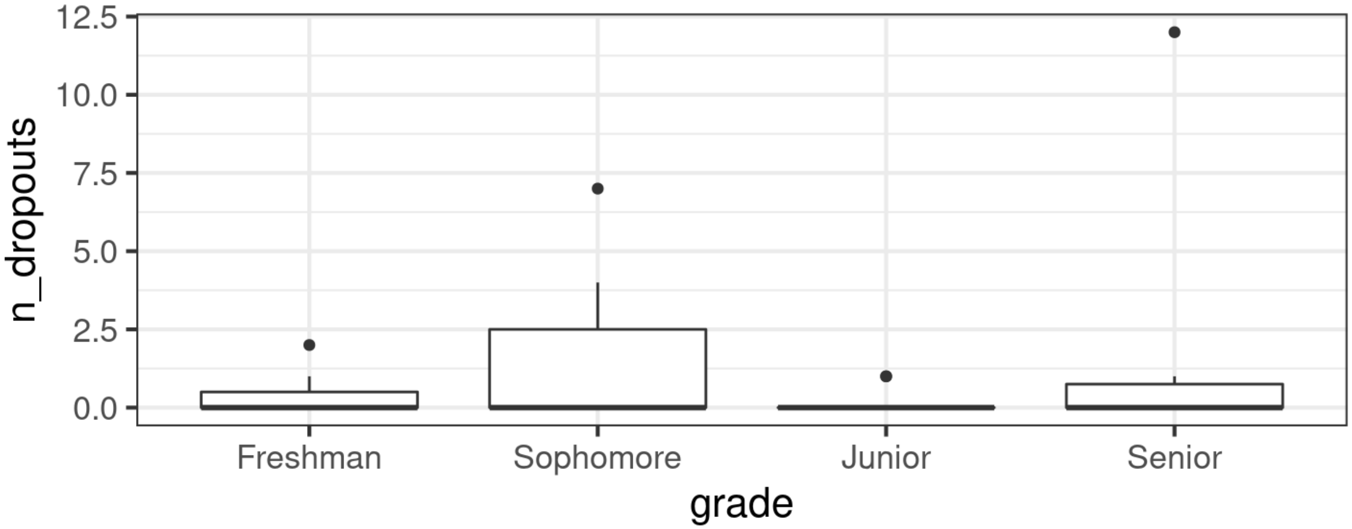
```
head(dropouts_fct)
```

```
## # A tibble: 6 × 3
##   CDS_CODE       grade      n_dropouts
##   <chr>          <fct>           <dbl>
## 1 45699716050231 Junior              0
## 2 45700036050330 Junior              0
## 3 12630401230069 Sophomore           0
## 4 09100900930131 Sophomore           1
## 5 15633216009179 Junior              0
## 6 33670330113647 Sophomore           0
```

Plot again

Now let's make our plot again:

```
dropouts_fct %>%
  ggplot(mapping = aes(x = grade, y = n_dropouts)) +
  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 ?

Character data is arranged alphabetically.

```
dropouts_subset %>%
  arrange(grade)
```

```
## # A tibble: 32 × 3
##    CDS_CODE       grade    n_dropouts
##    <chr>          <chr>         <dbl>
##  1 19643941931823 Freshman          1
##  2 11626616007611 Freshman          0
##  3 23656150128280 Freshman          0
##  4 30664313030616 Freshman          0
##  5 54719935432414 Freshman          2
##  6 22655326025050 Freshman          0
##  7 37683386039911 Freshman          0
##  8 45699716050231 Junior            0
##  9 45700036050330 Junior            0
## 10 15633216009179 Junior            0
## # … with 22 more rows
```

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

Arranging Factors

Factor data is arranged by level.

```
dropouts_fct %>%
  arrange(grade)
```

```
## # A tibble: 32 × 3
##    CDS_CODE       grade     n_dropouts
##    <chr>          <fct>          <dbl>
##  1 19643941931823 Freshman           1
##  2 11626616007611 Freshman           0
##  3 23656150128280 Freshman           0
##  4 30664313030616 Freshman           0
##  5 54719935432414 Freshman           2
##  6 22655326025050 Freshman           0
##  7 37683386039911 Freshman           0
##  8 12630401230069 Sophomore          0
##  9 09100900930131 Sophomore          1
## 10 33670330113647 Sophomore          0
## # … with 22 more rows
```

Nice! Now this is what we would want!

Making tables with characters

Tables grouped by a character are arranged alphabetically.

```
dropouts_subset %>%
  group_by(grade) %>%
  summarize(total_dropouts = sum(n_dropouts))
```

```
## # A tibble: 4 × 2
##   grade     total_dropouts
##   <chr>              <dbl>
## 1 Freshman               3
## 2 Junior                 2
## 3 Senior                13
## 4 Sophomore             12
```

Making tables with factors

Tables grouped by a factor are arranged by level.

```
dropouts_fct %>%
  group_by(grade) %>%
  summarize(total_dropouts = sum(n_dropouts))
```

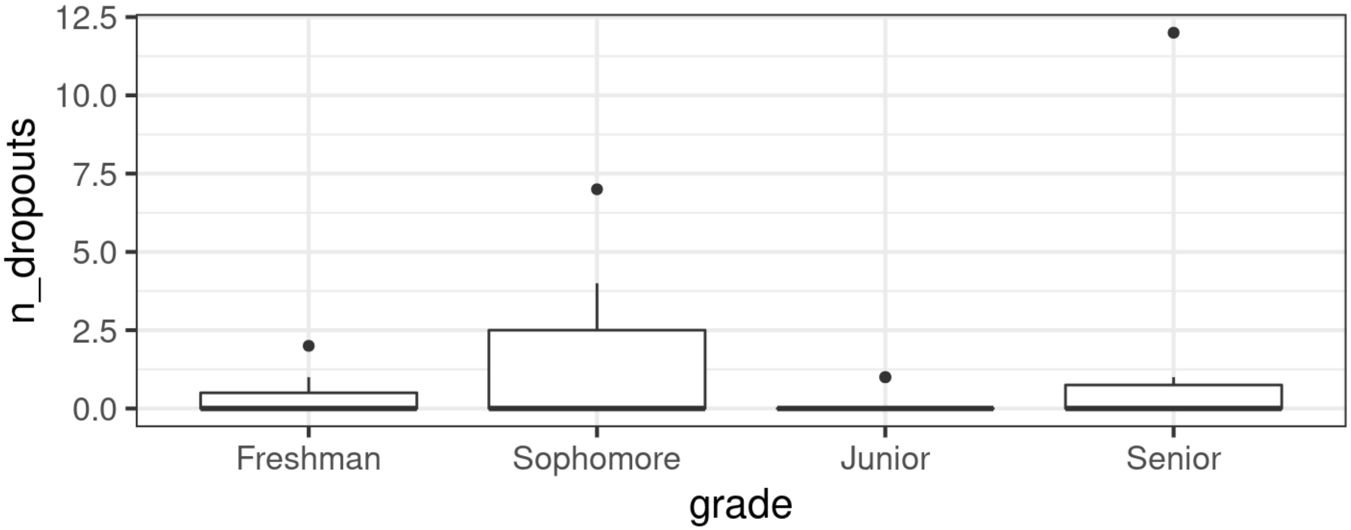
```
## # A tibble: 4 × 2
##   grade     total_dropouts
##   <fct>              <dbl>
## 1 Freshman               3
## 2 Sophomore             12
## 3 Junior                 2
## 4 Senior                13
```

`forcats` for ordering

What if we wanted to order `grade` by increasing `n_dropouts`?

```
library(forcats)

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



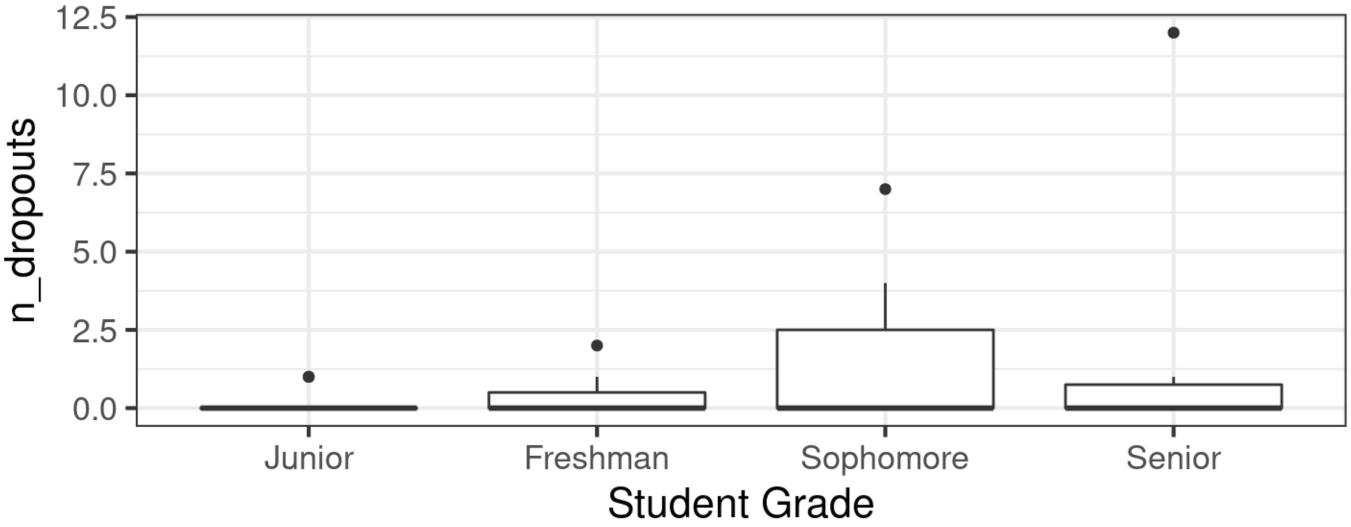
This would be useful for identifying easily which grade to focus on.

forcats for ordering

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

```
library(forcats)

dropouts_fct %>%
  ggplot(mapping = aes(x = fct_reorder(grade, n_dropouts, mean), y = n_dropouts)) +
  geom_boxplot() +
  labs(x = "Student Grade") +
  theme_bw(base_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)
dropouts_fct <-
  dropouts_fct %>%
  mutate("tardy" = sample(0:7, size = 32, replace = TRUE))

head(dropouts_fct)
```

```
## # A tibble: 6 × 4
##   CDS_CODE       grade     n_dropouts tardy
##   <chr>          <fct>          <dbl> <int>
## 1 45699716050231 Junior             0     1
## 2 45700036050330 Junior             0     3
## 3 12630401230069 Sophomore          0     5
## 4 09100900930131 Sophomore          1     1
## 5 15633216009179 Junior             0     4
## 6 33670330113647 Sophomore          0     0
```

Plotting new variable

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

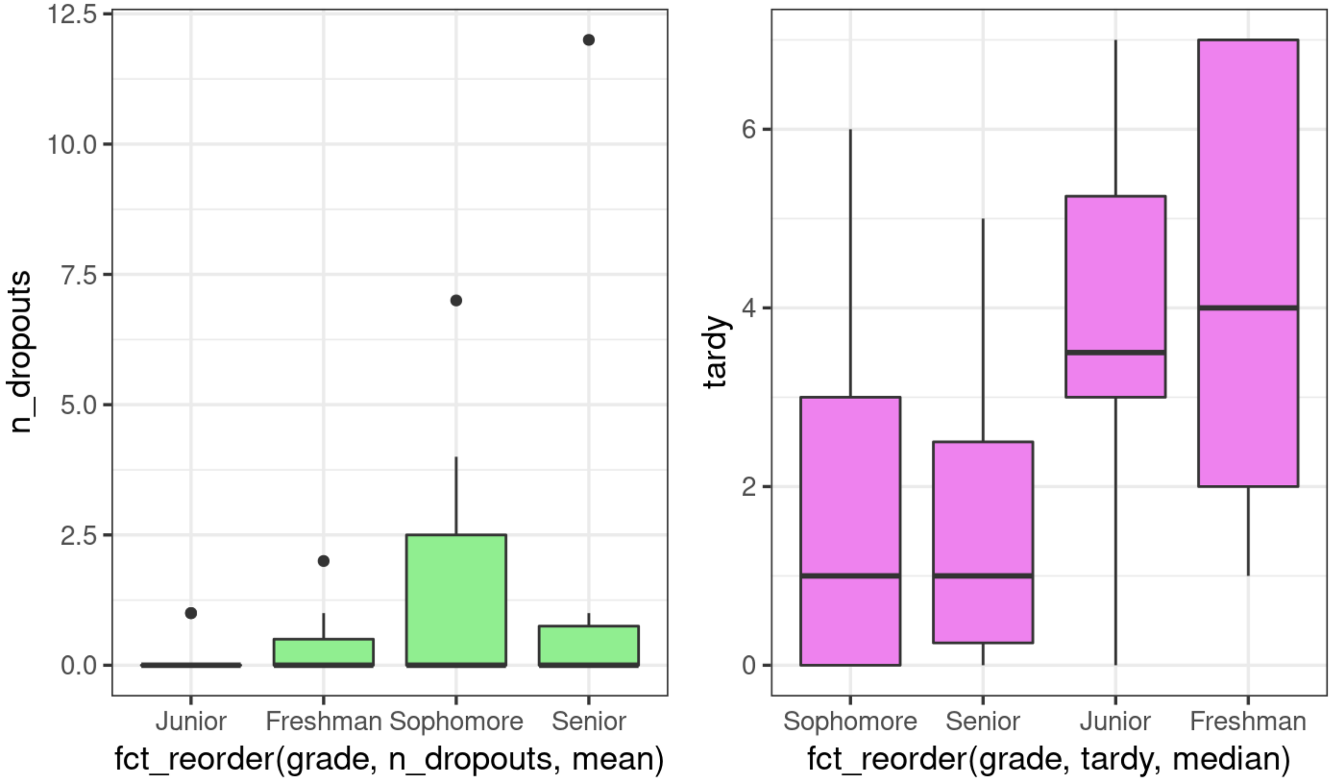
```
dropouts_plot <- dropouts_fct %>%
  ggplot(mapping = aes(
    x = fct_reorder(grade, n_dropouts, mean),
    y = n_dropouts
  )) +
  geom_boxplot(fill = "lightgreen") +
  theme_bw(base_size = 13)

tardy_plot <- dropouts_fct %>%
  ggplot(mapping = aes(
    x = fct_reorder(grade, tardy, median),
    y = tardy
  )) +
  geom_boxplot(fill = "violet") +
  theme_bw(base_size = 13)

library(patchwork)
dropouts_plot + tardy_plot
```

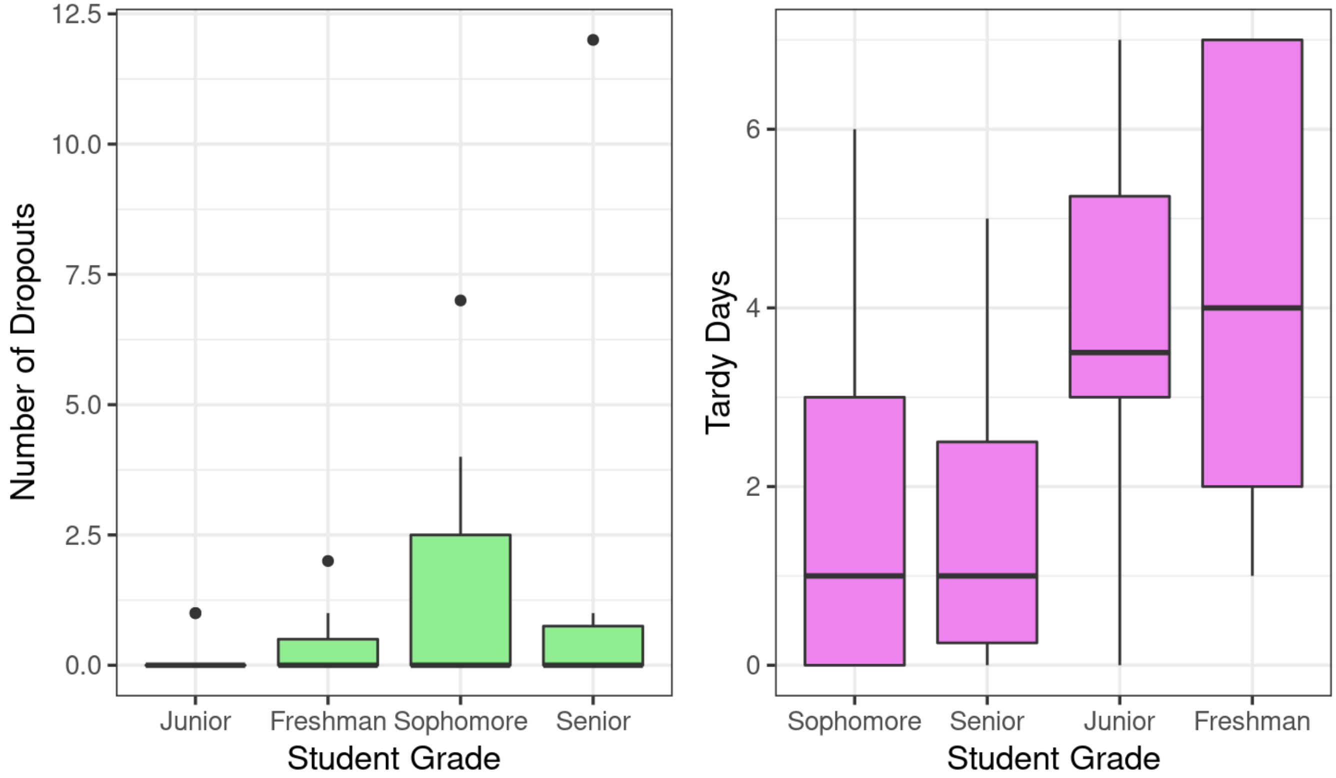
Plotting new variable

The result!



Cleaning up the plots

```
dropouts_plot +
  labs(x = "Student Grade", y = "Number of Dropouts") +
  tardy_plot +
  labs(x = "Student Grade", y = "Tardy Days")
```



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.

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

```
## # A tibble: 4 × 3
##   f             n     p
##   <fct>     <int> <dbl>
## 1 Freshman      7 0.219
## 2 Sophomore     7 0.219
## 3 Junior       12 0.375
## 4 Senior        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