# 3-ish forcats tricks

## for @WeAreRLadies

Suzan Baert

# Tip 1: Lumping levels together in Other

You can use the `fct_lump()` function from `forcats` to keep the 3 most frequent levels and get everything else coded as 'Other'.

## Current version

```
starwars %>%
  count(eye_color)
```

```
## # A tibble: 15 x 2
##    eye_color         n
##    <chr>         <int>
##  1 black            10
##  2 blue             19
##  3 blue-gray         1
##  4 brown            21
##  5 dark              1
##  6 gold              1
##  7 green, yellow     1
##  8 hazel             3
##  9 orange            8
## 10 pink              1
## 11 red               5
## 12 red, blue         1
## 13 unknown           3
## 14 white             1
```

## Using fct_lump version

```
library(forcats)

starwars %>%
  mutate(eye_color = fct_lump(eye_color, n=3)) %>%
  count(eye_color)
```

```
## # A tibble: 4 x 2
##   eye_color     n
##   <fct>     <int>
## 1 blue         19
## 2 brown        21
## 3 yellow       11
## 4 Other        36
```

# Tip 1b: Lumping levels together in Other - but under your control

You can use the `fct_other()` function to control yourself what to keep and what to drop inside "other"

## Control what to keep separate

```
to_keep <- c("brown", "blue", "hazel")

starwars %>%
  mutate(eye_color = fct_other(eye_color,
                               keep = to_keep)) %>%
  count(eye_color)
```

```
## # A tibble: 4 x 2
##   eye_color     n
##   <fct>     <int>
## 1 blue         19
## 2 brown        21
## 3 hazel         3
## 4 Other        44
```

## Control what to drop into "other"

```
to_other <- c("white", "pink", "gold", "unknown", "blu

starwars %>%
  mutate(eye_color = fct_other(eye_color,
                               drop = to_other)) %>%
  count(eye_color)
```

```
## # A tibble: 8 x 2
##   eye_color     n
##   <fct>     <int>
## 1 black        10
## 2 blue         19
## 3 brown        21
## 4 hazel         3
## 5 orange        8
## 6 red           5
## 7 yellow       11
## 8 Other        10
```

# Tip 2: Moving elements of a factor around

You can use the `fct_relevel()` to change the order of some elements.

- The original vector:

```
weekdays_factor
```

```
## [1] monday    tuesday   wednesday thursday  friday    saturday  sunday
## 7 Levels: monday < tuesday < wednesday < thursday < ... < sunday
```

- To change the starting order, add the elements you want to come first in fct_relevel.
  Note: What does not get mentioned will be added at the end in the current order.

```
fct_relevel(weekdays_factor, "sunday")
```

```
## [1] monday    tuesday   wednesday thursday  friday    saturday  sunday
## 7 Levels: sunday < monday < tuesday < wednesday < thursday < ... < saturday
```

# Tip 2b: Moving elements of a factor around

- To move elements to other positions add the extra "after" element:

```
fct_relevel(weekdays_factor, "sunday", after = 2)
```

```
## [1] monday    tuesday   wednesday thursday  friday    saturday  sunday
## 7 Levels: monday < tuesday < sunday < wednesday < thursday < ... < saturday
```

- To move something to the very end:

```
fct_relevel(weekdays_factor, "sunday", after = Inf)
```

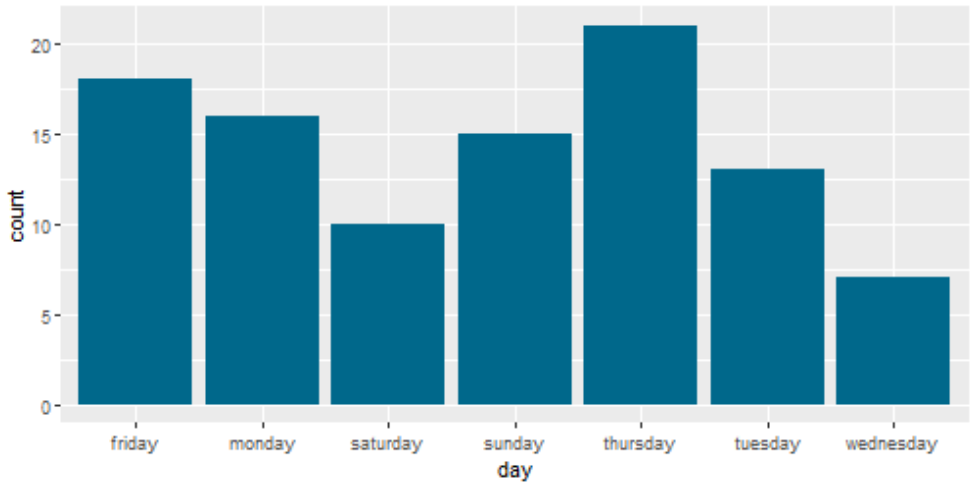
```
## [1] monday    tuesday   wednesday thursday  friday    saturday  sunday
## 7 Levels: monday < tuesday < wednesday < thursday < ... < sunday
```

# Tip 3: Changing factor levels based on count

Using `fct_infreq()` to change the factor levels based on the frequency it occurs.

## No ordering

```
ggplot(weekdays_df) +
  geom_bar(aes(day), fill = "deepskyblue4")
```




## After fct_infreq

```
weekdays_df %>%
  mutate(day = fct_infreq(day)) %>%
  ggplot() +
  geom_bar(aes(day), fill = "deepskyblue4")
```