Getting things into the right order

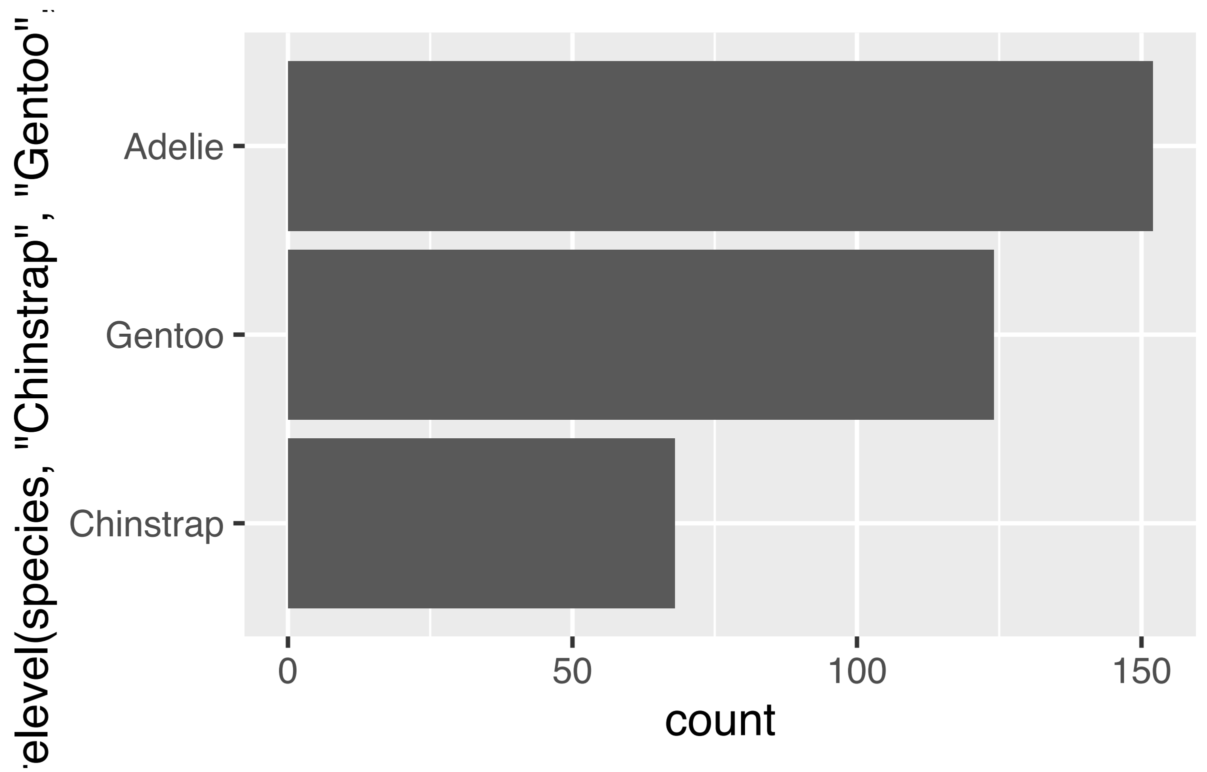
Claus O. Wilke

last updated: 2021-02-26

Remember from "Visualizing amounts"

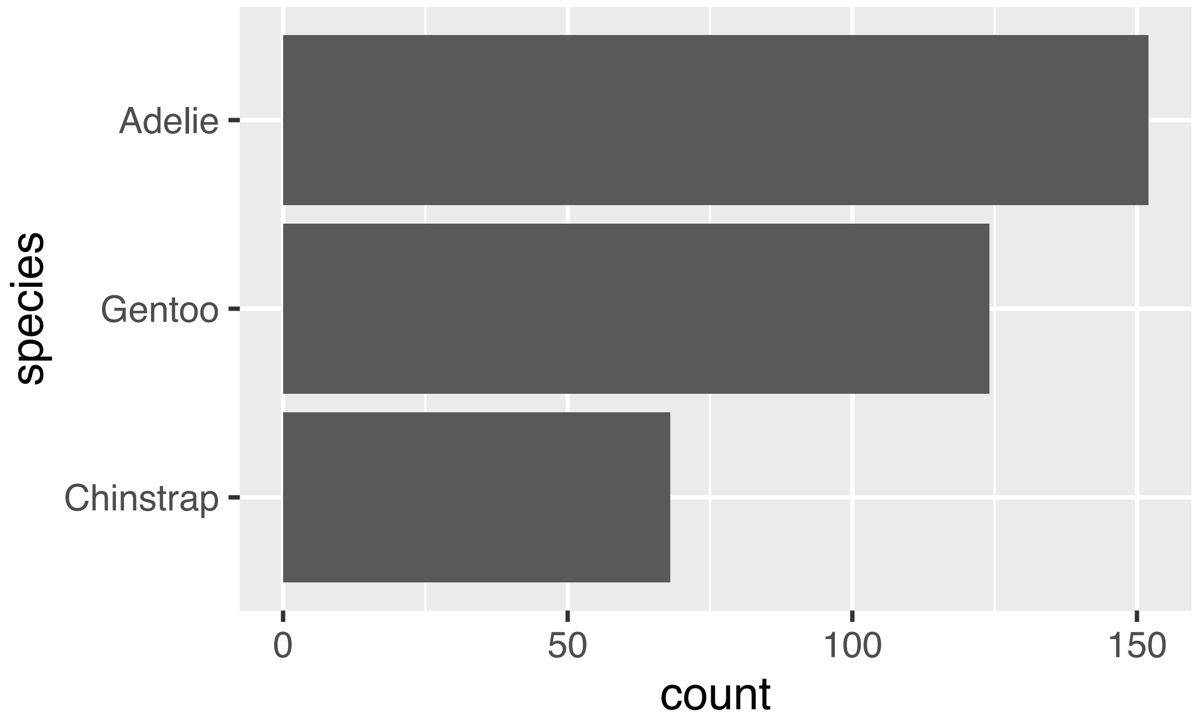
We can use `fct_relevel()` to manually order the bars in a bar plot

```
ggplot(penguins, aes(y = fct_relevel(species, "Chinstrap", "Gentoo", "Ad
  geom_bar()
```



Somewhat cleaner: mutate first, then plot

```
penguins %>%
  mutate(species = fct_relevel(species, "Chinstrap", "Gentoo", "Adelie"
  ggplot(aes(y = species)) +
  geom_bar()
```



We order things in ggplot with factors

```
penguins %>%
  mutate(species = fct_relevel(species, "Chinstrap", "Gentoo", "Adelie")
  slice(1:30) %>%    # get first 30 rows
  pull(species)      # pull out just the `species` column
```

```
 [1] Adelie Adelie Adelie Adelie Adelie Adelie Adelie Adelie Adelie Adelie
[11] Adelie Adelie Adelie Adelie Adelie Adelie Adelie Adelie Adelie Adelie
[21] Adelie Adelie Adelie Adelie Adelie Adelie Adelie Adelie Adelie Adelie
Levels: Chinstrap Gentoo Adelie
```

- The column `species` is a factor
- A factor is a categorical variable with defined categories called levels
- For factors, ggplot generally places visual elements in the order defined by the levels

Manual ordering of factor levels: `fct_relevel()`

```r
penguins %>%
  mutate(species = fct_relevel(species)) %>%
  slice(1:30) %>%   # get first 30 rows
  pull(species)     # pull out just the `species` column
```

```
 [1] Adelie Adelie Adelie Adelie Adelie Adelie Adelie Adelie Adelie Adelie
[11] Adelie Adelie Adelie Adelie Adelie Adelie Adelie Adelie Adelie Adelie
[21] Adelie Adelie Adelie Adelie Adelie Adelie Adelie Adelie Adelie Adelie
Levels: Adelie Chinstrap Gentoo
```

Default: alphabetic order

Manual ordering of factor levels: `fct_relevel()`

```
penguins %>%
  mutate(species = fct_relevel(species, "Gentoo")) %>%
  slice(1:30) %>%    # get first 30 rows
  pull(species)      # pull out just the `species` column
```

```
 [1] Adelie Adelie Adelie Adelie Adelie Adelie Adelie Adelie Adelie Adelie
[11] Adelie Adelie Adelie Adelie Adelie Adelie Adelie Adelie Adelie Adelie
[21] Adelie Adelie Adelie Adelie Adelie Adelie Adelie Adelie Adelie Adelie
Levels: Gentoo Adelie Chinstrap
```

Move `"Gentoo"` in front, rest alphabetic

Manual ordering of factor levels: `fct_relevel()`

```r
penguins %>%
  mutate(species = fct_relevel(species, "Chinstrap", "Gentoo")) %>%
  slice(1:30) %>%    # get first 30 rows
  pull(species)      # pull out just the `species` column
```

```
 [1] Adelie Adelie Adelie Adelie Adelie Adelie Adelie Adelie Adelie Adelie
[11] Adelie Adelie Adelie Adelie Adelie Adelie Adelie Adelie Adelie Adelie
[21] Adelie Adelie Adelie Adelie Adelie Adelie Adelie Adelie Adelie Adelie
Levels: Chinstrap Gentoo Adelie
```

Move `"Chinstrap"` in front, then `"Gentoo"`, rest alphabetic

Manual ordering of factor levels: `fct_relevel()`

```
penguins %>%
  mutate(species = fct_relevel(species, "Chinstrap", "Adelie", "Gentoo")
  slice(1:30) %>%    # get first 30 rows
  pull(species)      # pull out just the `species` column
```

```
 [1] Adelie Adelie Adelie Adelie Adelie Adelie Adelie Adelie Adelie Adelie
[11] Adelie Adelie Adelie Adelie Adelie Adelie Adelie Adelie Adelie Adelie
[21] Adelie Adelie Adelie Adelie Adelie Adelie Adelie Adelie Adelie Adelie
Levels: Chinstrap Adelie Gentoo
```

Use order `"Chinstrap"`, `"Adelie"`, `"Gentoo"`

Manual ordering of factor levels: `fct_relevel()`

```r
penguins %>%
  mutate(species = fct_relevel(species, "Gentoo", "Chinstrap", "Adelie")
  slice(1:30) %>%    # get first 30 rows
  pull(species)      # pull out just the `species` column
```

```
 [1] Adelie Adelie Adelie Adelie Adelie Adelie Adelie Adelie Adelie Adelie
[11] Adelie Adelie Adelie Adelie Adelie Adelie Adelie Adelie Adelie Adelie
[21] Adelie Adelie Adelie Adelie Adelie Adelie Adelie Adelie Adelie Adelie
Levels: Gentoo Chinstrap Adelie
```

Use order `"Gentoo"`, `"Chinstrap"`, `"Adelie"`

The order of the y axis is from bottom to top

```
penguins %>%
  mutate(species = fct_relevel(species, "Chinstrap", "Gentoo", "Adelie")
  ggplot(aes(y = species)) +
  geom_bar()
```

Reorder based on frequency: `fct_infreq()`

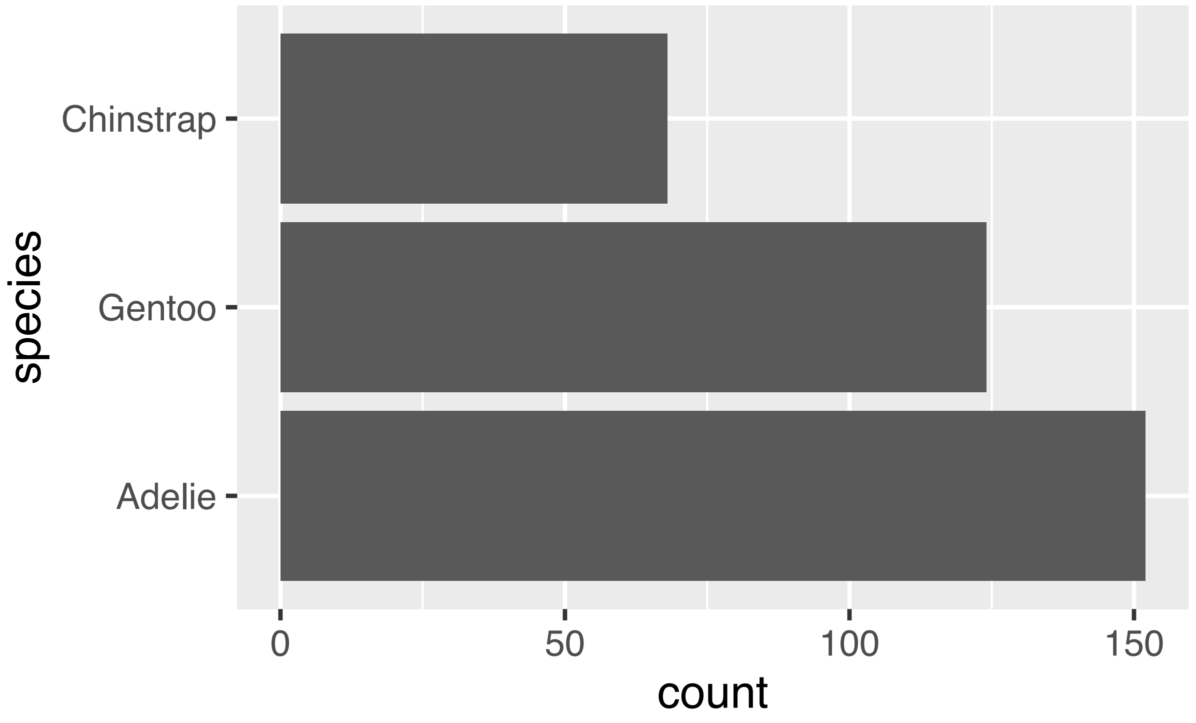
```r
penguins %>%
  mutate(species = fct_infreq(species)) %>%
  slice(1:30) %>%    # get first 30 rows
  pull(species)      # pull out just the `species` column
```

```
 [1] Adelie Adelie Adelie Adelie Adelie Adelie Adelie Adelie Adelie Adelie
[11] Adelie Adelie Adelie Adelie Adelie Adelie Adelie Adelie Adelie Adelie
[21] Adelie Adelie Adelie Adelie Adelie Adelie Adelie Adelie Adelie Adelie
Levels: Adelie Gentoo Chinstrap
```

- Use the order defined by the number of penguins of different species
- The order is descending, from most frequent to least frequent

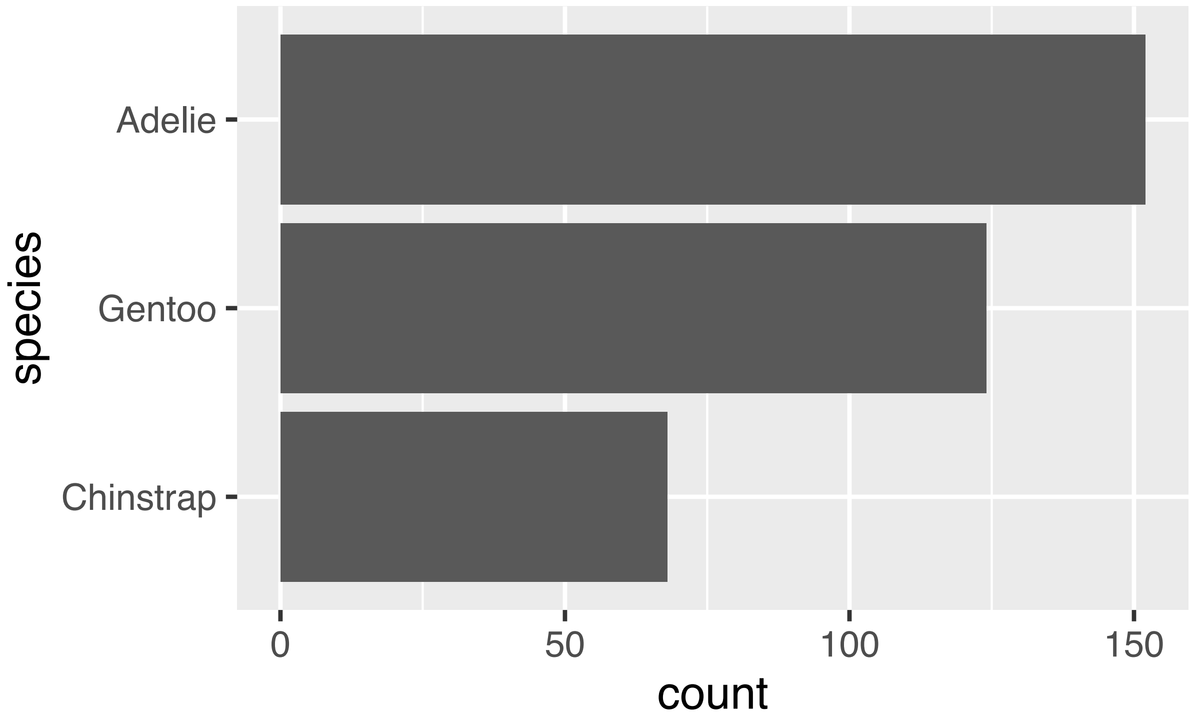
Reorder based on frequency: `fct_infreq()`

```r
penguins %>%
  mutate(species = fct_infreq(species)) %>%
  ggplot(aes(y = species)) + geom_bar()
```



Reverse order: `fct_rev()`

```
penguins %>%
  mutate(species = fct_rev(fct_infreq(species))) %>%
  ggplot(aes(y = species)) + geom_bar()
```



Reorder based on numeric values: `fct_reorder()`

```r
penguins %>%
  count(species)
```

```
# A tibble: 3 x 2
  species       n
  <fct>     <int>
1 Adelie      152
2 Chinstrap    68
3 Gentoo      124
```

```r
penguins %>%
  count(species) %>%
  mutate(species = fct_reorder(species, n)) %>%
  pull(species)     # pull out just the `species` column
```

```
[1] Adelie    Chinstrap Gentoo
Levels: Chinstrap Gentoo Adelie
```

The order is ascending, from smallest to largest value

Reorder based on numeric values: `fct_reorder()`

```r
penguins %>%
  count(species) %>%
  mutate(species = fct_reorder(species, n)) %>%
  ggplot(aes(n, species)) + geom_col()
```

Compare to see the difference

```
penguins %>%
  count(species) %>% # summarize da
  mutate(species = fct_reorder(spec
```

```
# A tibble: 3 x 2
  species       n
  <fct>     <int>
1 Adelie      152
2 Chinstrap    68
3 Gentoo      124
```

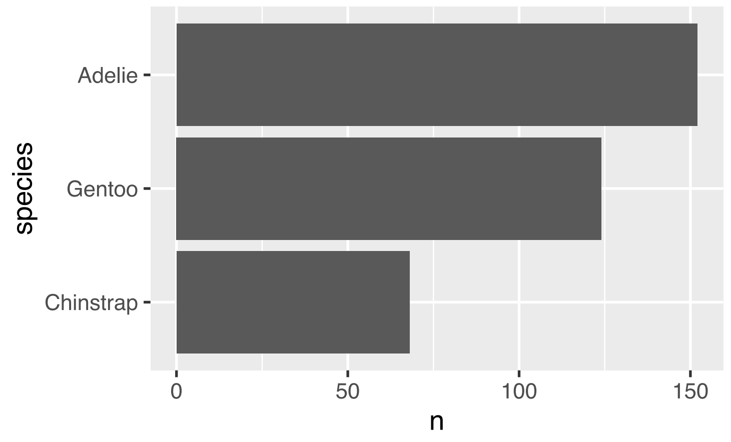
```
penguins %>%
  # modify the original dataset, no
  mutate(species = fct_infreq(speci
```

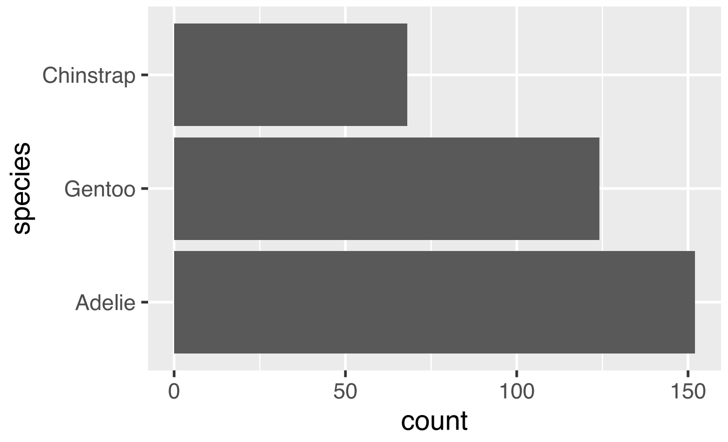
```
# A tibble: 344 x 8
   species island bill_length_mm bill_depth_mm
   <fct>   <fct>           <dbl>         <dbl>
 1 Adelie  Torge…           39.1          18.7
 2 Adelie  Torge…           39.5          17.4
 3 Adelie  Torge…           40.3          18
 4 Adelie  Torge…           NA            NA
 5 Adelie  Torge…           36.7          19.3
 6 Adelie  Torge…           39.3          20.6
 7 Adelie  Torge…           38.9          17.8
 8 Adelie  Torge…           39.2          19.6
 9 Adelie  Torge…           34.1          18.1
10 Adelie  Torge…           42            20.2
# … with 334 more rows, and 2 more variables: s
```

Compare to see the difference

```
penguins %>%
  count(species) %>% # summarize da
  mutate(species = fct_reorder(spec
  ggplot(aes(n, species)) + geom_co
```

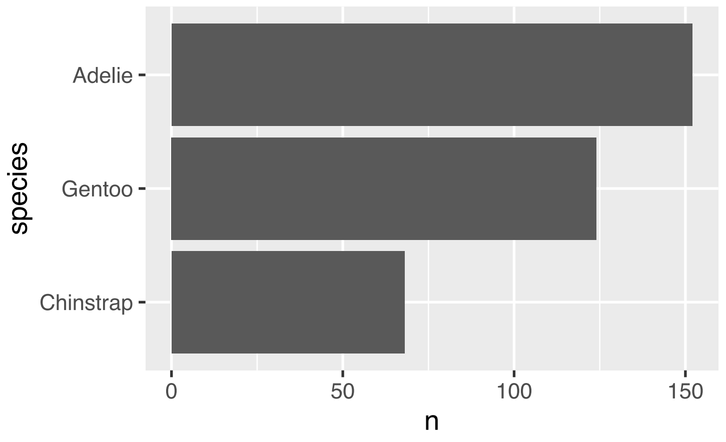
```
penguins %>%
  # modify the original dataset, no
  mutate(species = fct_infreq(specie
  ggplot(aes(y = species)) + geom_ba
```



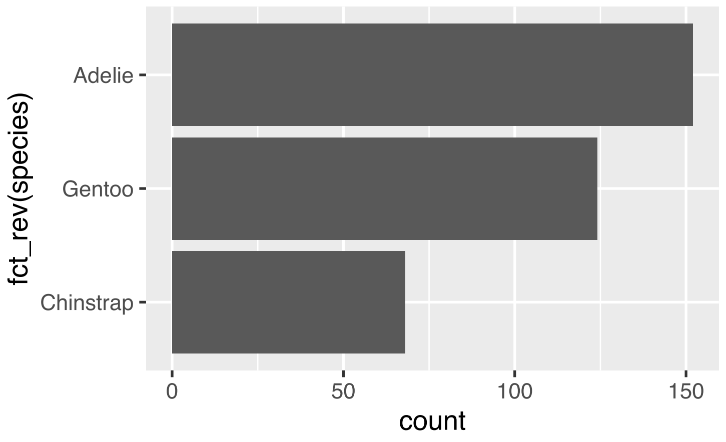


Compare to see the difference

```r
penguins %>%
  count(species) %>% # summarize da
  mutate(species = fct_reorder(speci
  ggplot(aes(n, species)) + geom_col
```



```r
penguins %>%
  # modify the original dataset, no
  mutate(species = fct_infreq(specie
  ggplot(aes(y = fct_rev(species)))
```



We can do more than just order bars

The gapminder dataset: Life expectancy data

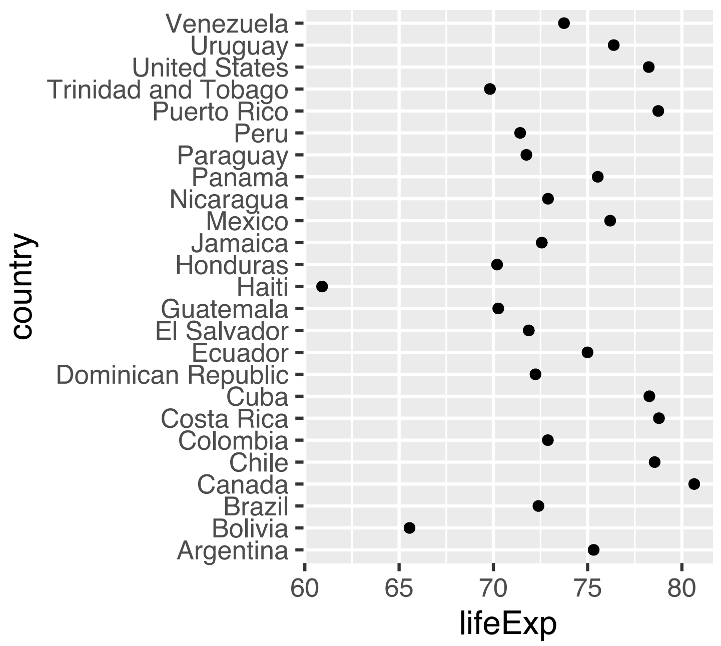
```r
library(gapminder)

gapminder
```

```
# A tibble: 1,704 x 6
   country     continent  year lifeExp      pop gdpPercap
   <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
 1 Afghanistan Asia       1952    28.8  8425333      779.
 2 Afghanistan Asia       1957    30.3  9240934      821.
 3 Afghanistan Asia       1962    32.0 10267083      853.
 4 Afghanistan Asia       1967    34.0 11537966      836.
 5 Afghanistan Asia       1972    36.1 13079460      740.
 6 Afghanistan Asia       1977    38.4 14880372      786.
 7 Afghanistan Asia       1982    39.9 12881816      978.
 8 Afghanistan Asia       1987    40.8 13867957      852.
 9 Afghanistan Asia       1992    41.7 16317921      649.
10 Afghanistan Asia       1997    41.8 22227415      635.
# … with 1,694 more rows
```

Life expectancy in the Americas in 2007

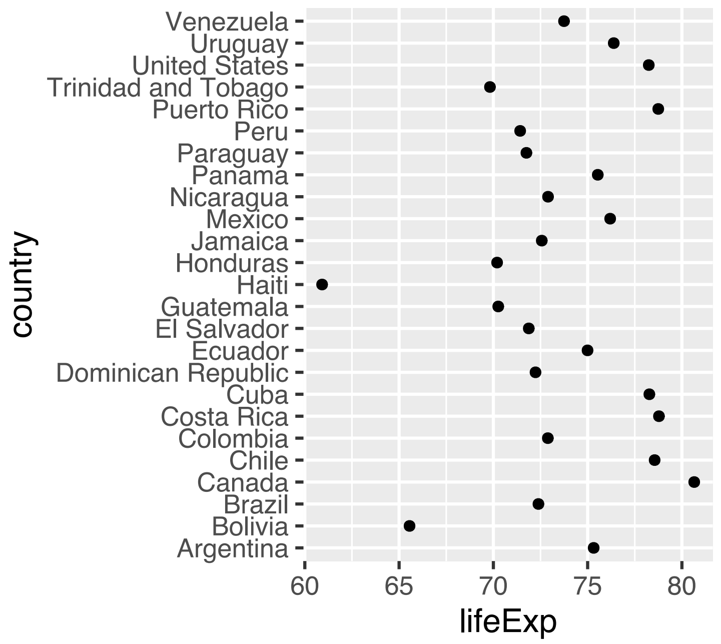
```
gapminder %>%
  filter(
    year == 2007,
    continent == "Americas"
  ) %>%
  ggplot(aes(lifeExp, country)) +
  geom_point()
```



Life expectancy in the Americas in 2007

```r
gapminder %>%
  filter(
    year == 2007,
    continent == "Americas"
  ) %>%
  ggplot(aes(lifeExp, country)) +
  geom_point()
```

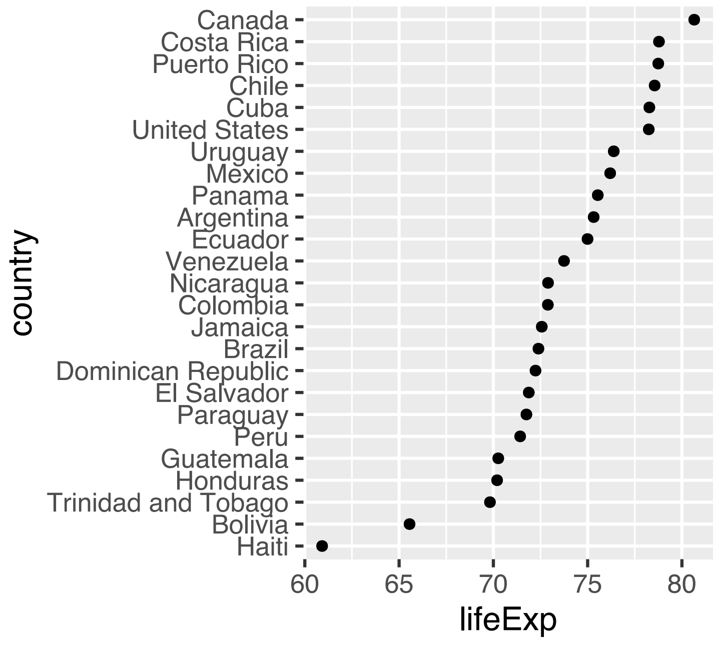
Reminder:
Default order is alphabetic, from bottom to top



Life expectancy, ordered from highest to lowest

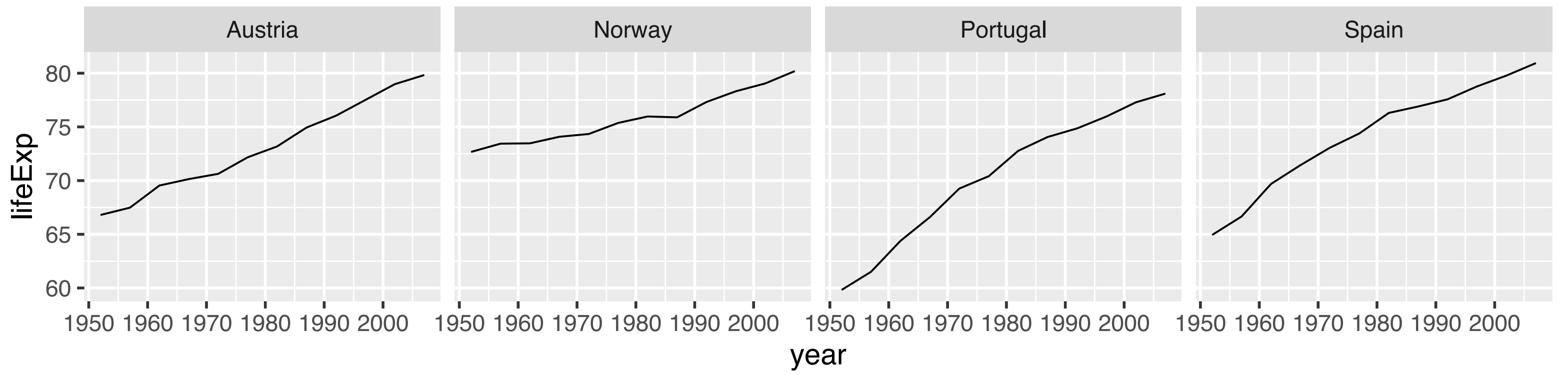
```r
gapminder %>%
  filter(
    year == 2007,
    continent == "Americas"
  ) %>%
  mutate(
    country = fct_reorder(country
  ) %>%
  ggplot(aes(lifeExp, country)) +
  geom_point()
```

Order is ascending from bottom to top



We can also order facets

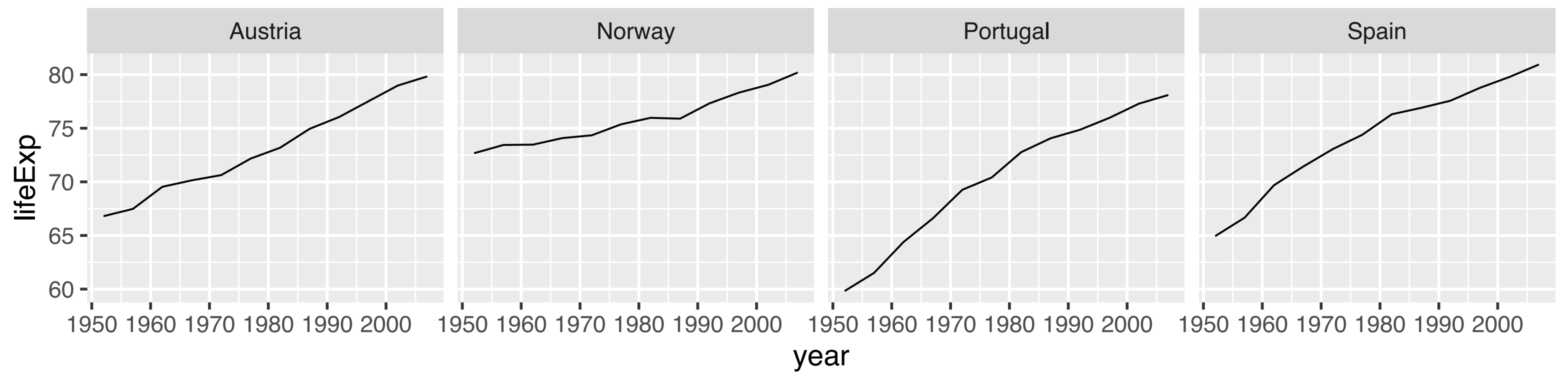
```r
gapminder %>%
  filter(country %in% c("Norway", "Portugal", "Spain", "Austria")) %>%
  ggplot(aes(year, lifeExp)) + geom_line() +
  facet_wrap(vars(country), nrow = 1)
```



- Default ordering is alphabetic; there's no good reason for this ordering

We can also order facets

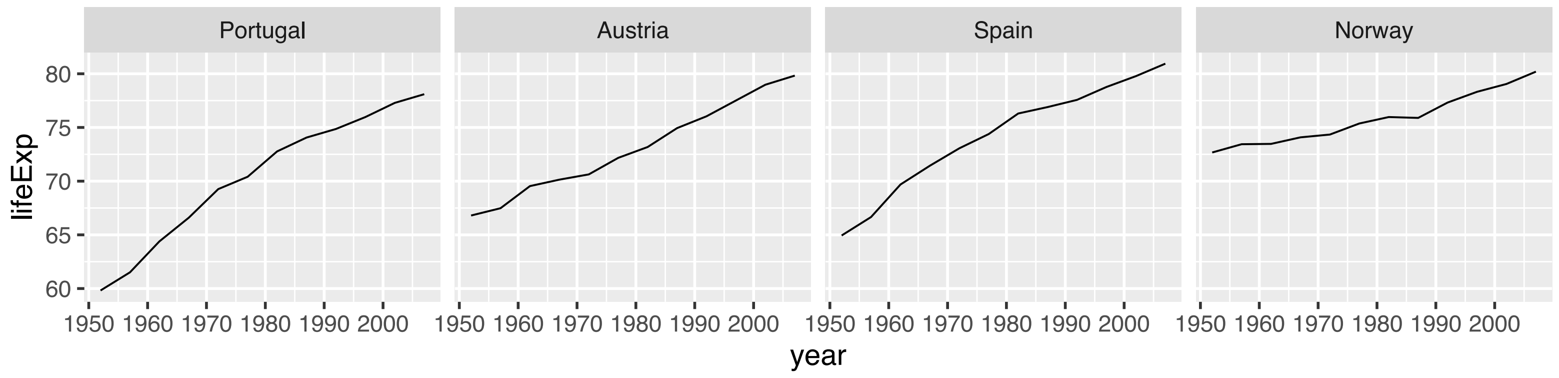
```r
gapminder %>%
  filter(country %in% c("Norway", "Portugal", "Spain", "Austria")) %>%
  ggplot(aes(year, lifeExp)) + geom_line() +
  facet_wrap(vars(country), nrow = 1)
```



- Let's apply `fct_reorder()` and see what happens

We can also order facets

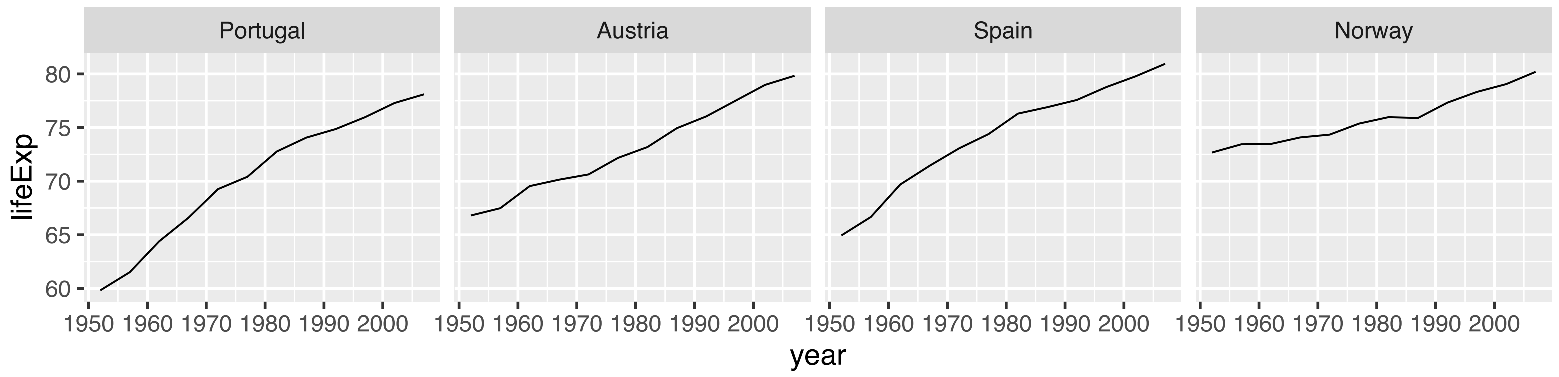
```
gapminder %>%
  filter(country %in% c("Norway", "Portugal", "Spain", "Austria")) %>%
  mutate(country = fct_reorder(country, lifeExp)) %>% # default: order
  ggplot(aes(year, lifeExp)) + geom_line() +
  facet_wrap(vars(country), nrow = 1)
```



- When the levels of a factor occur more than once, `fct_reorder()` applies a summary function

We can also order facets

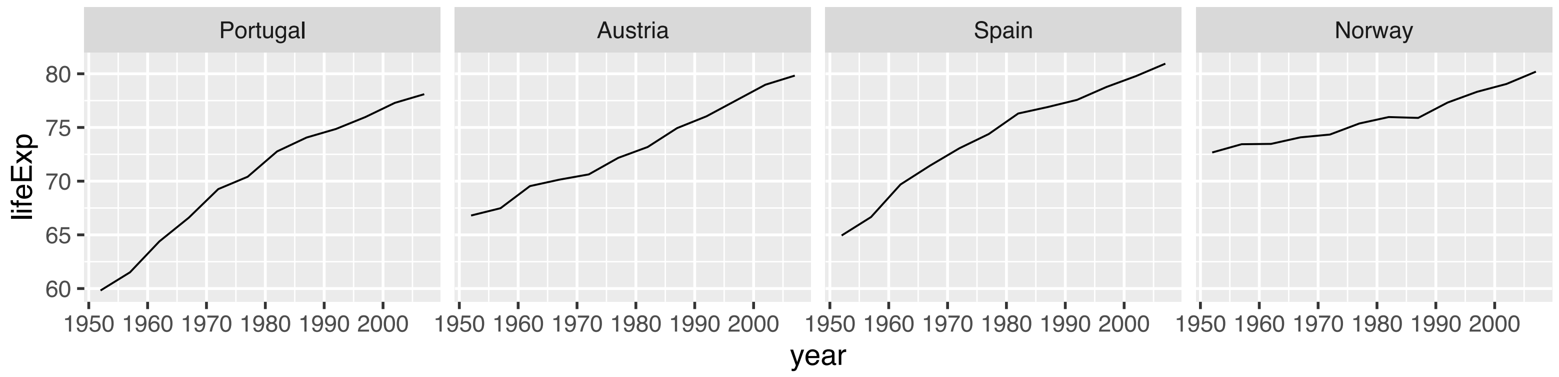
```r
gapminder %>%
  filter(country %in% c("Norway", "Portugal", "Spain", "Austria")) %>%
  mutate(country = fct_reorder(country, lifeExp)) %>% # default: order 
  ggplot(aes(year, lifeExp)) + geom_line() +
  facet_wrap(vars(country), nrow = 1)
```



- The default summary function is `median()`

We can also order facets

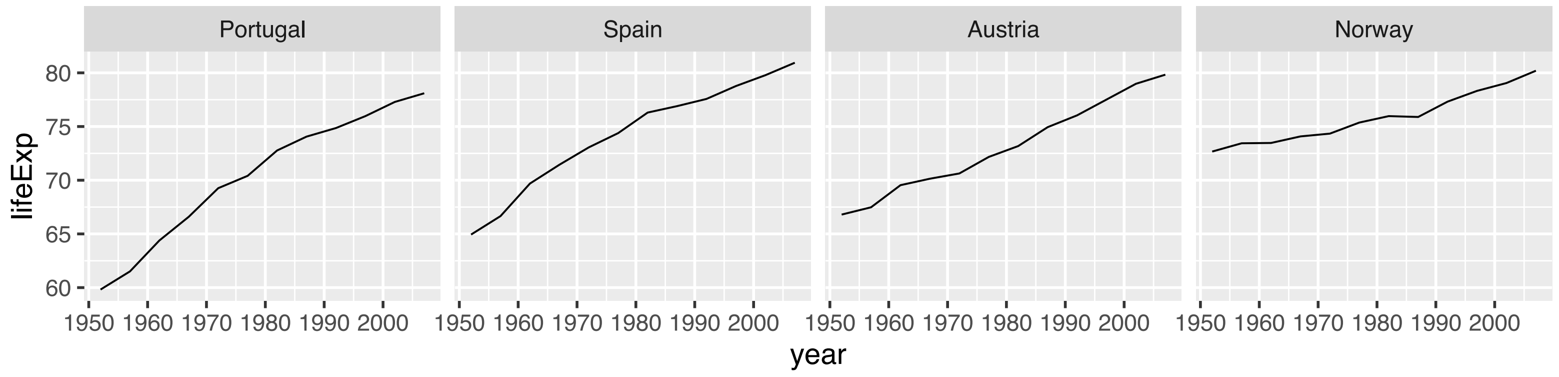
```r
gapminder %>%
  filter(country %in% c("Norway", "Portugal", "Spain", "Austria")) %>%
  mutate(country = fct_reorder(country, lifeExp, median)) %>% # order by
  ggplot(aes(year, lifeExp)) + geom_line() +
  facet_wrap(vars(country), nrow = 1)
```



- We can also set the summary function explicitly

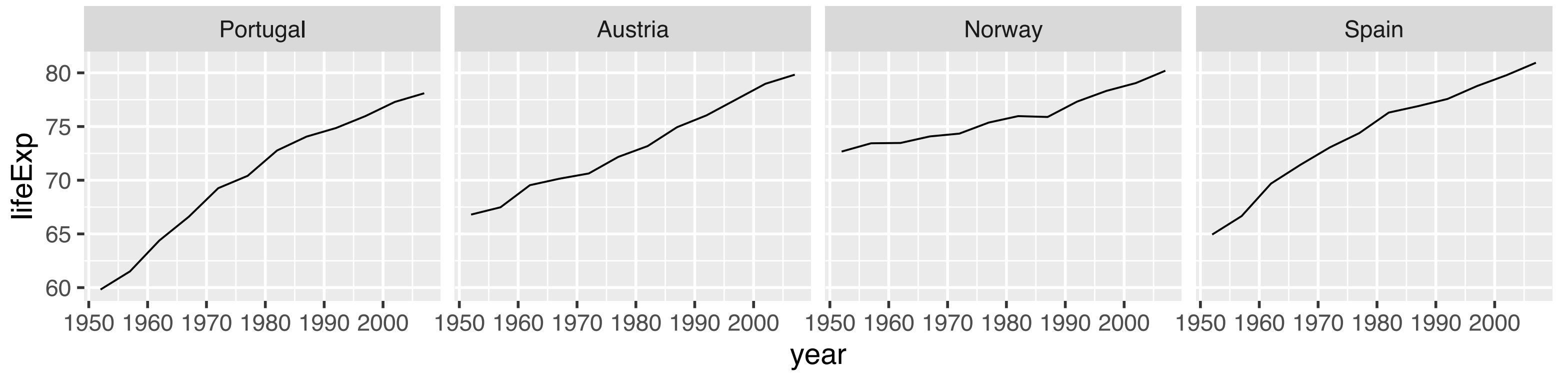
Alternative orderings: By smallest value per facet

```
gapminder %>%
  filter(country %in% c("Norway", "Portugal", "Spain", "Austria")) %>%
  mutate(country = fct_reorder(country, lifeExp, min)) %>% # order by m
  ggplot(aes(year, lifeExp)) + geom_line() +
  facet_wrap(vars(country), nrow = 1)
```



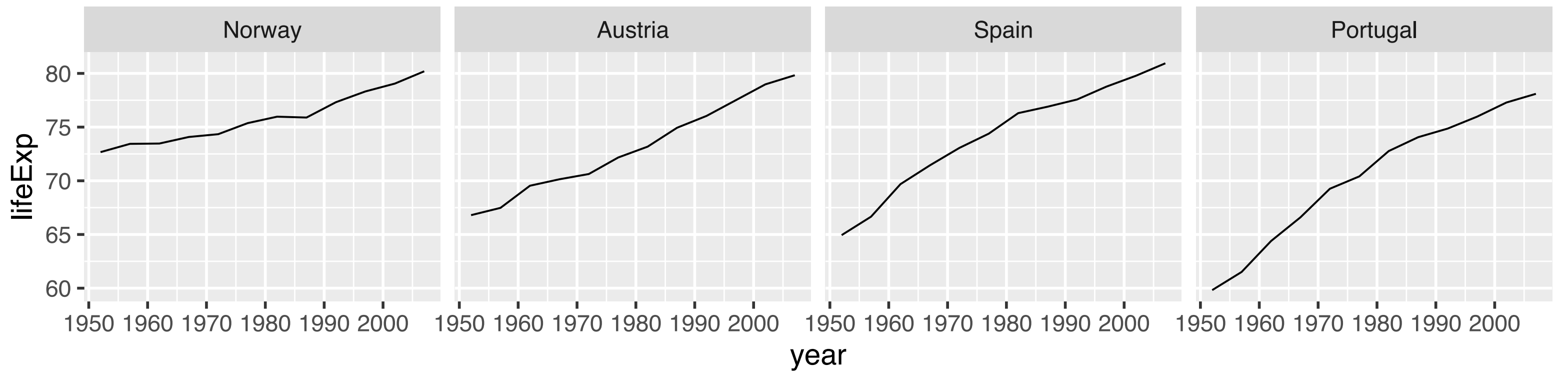
Alternative orderings: By largest value per facet

```r
gapminder %>%
  filter(country %in% c("Norway", "Portugal", "Spain", "Austria")) %>%
  mutate(country = fct_reorder(country, lifeExp, max)) %>% # order by ma
  ggplot(aes(year, lifeExp)) + geom_line() +
  facet_wrap(vars(country), nrow = 1)
```



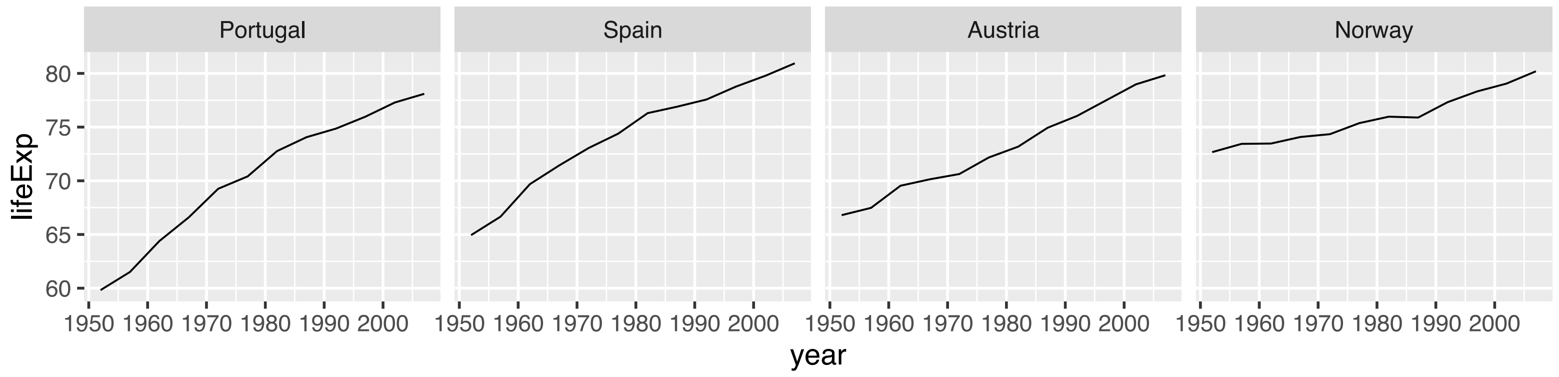
Alternative orderings: By smallest difference

```
gapminder %>%
  filter(country %in% c("Norway", "Portugal", "Spain", "Austria")) %>%
  # order by custom function: here, difference between max and min
  mutate(country = fct_reorder(country, lifeExp, function(x) { max(x) -
  ggplot(aes(year, lifeExp)) + geom_line() +
  facet_wrap(vars(country), nrow = 1)
```



Alternative orderings: By largest difference

```
gapminder %>%
  filter(country %in% c("Norway", "Portugal", "Spain", "Austria")) %>%
  # order by custom function: here, difference between min and max
  mutate(country = fct_reorder(country, lifeExp, function(x) { min(x) -
  ggplot(aes(year, lifeExp)) + geom_line() +
  facet_wrap(vars(country), nrow = 1)
```



Final example: Lumping factor levels together

Dataset: Flights out of New York City in 2013

```
library(nycflights13)

flight_data <- flights %>% # take data on individual flights
  left_join(airlines) %>%  # add in full-length airline names
  select(name, carrier, flight, year, month, day, origin, dest) # pick 
```

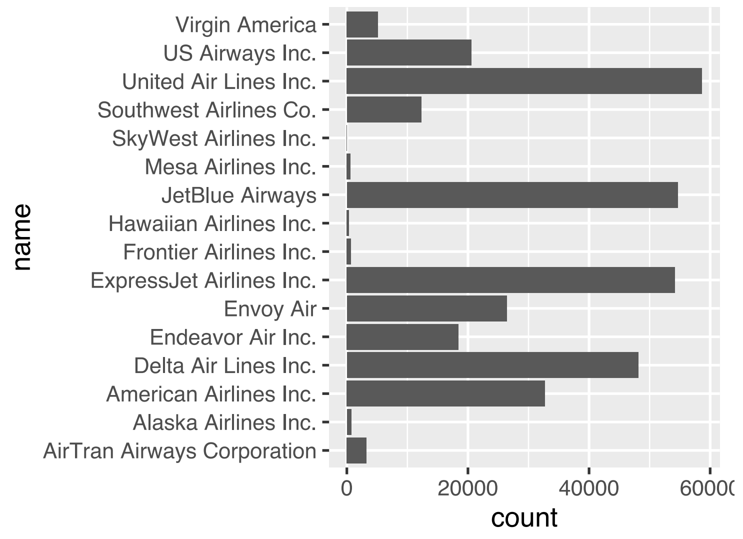
```
Joining, by = "carrier"
```

```
flight_data
```

```
# A tibble: 336,776 x 8
   name                     carrier flight  year month   day origin dest 
   <chr>                    <chr>    <int> <int> <int> <int> <chr>  <chr>
 1 United Air Lines Inc.    UA        1545  2013     1     1 EWR    IAH  
 2 United Air Lines Inc.    UA        1714  2013     1     1 LGA    IAH  
 3 American Airlines Inc.   AA        1141  2013     1     1 JFK    MIA  
 4 JetBlue Airways          B6         725  2013     1     1 JFK    BQN  
 5 Delta Air Lines Inc.     DL         461  2013     1     1 LGA    ATL  
```

Flights out of New York City in 2013

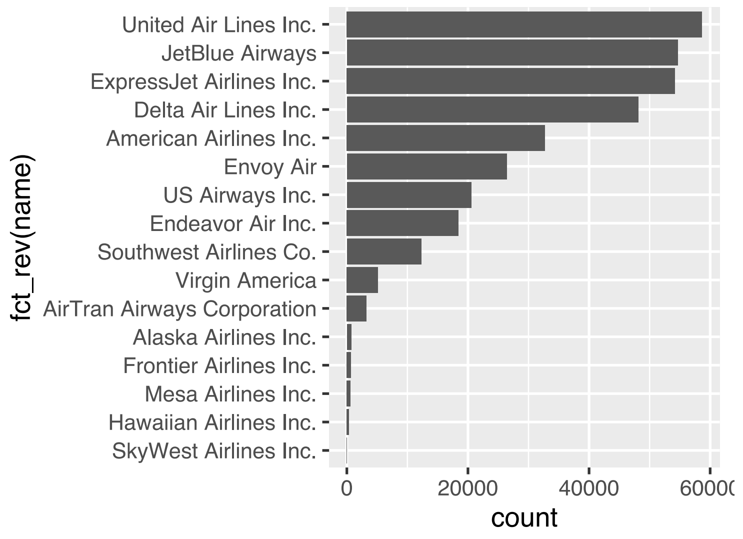
```
flight_data %>%
  ggplot(aes(y = name)) +
  geom_bar()
```



As (almost) always, the default alphabetic ordering is terrible

Flights out of New York City in 2013

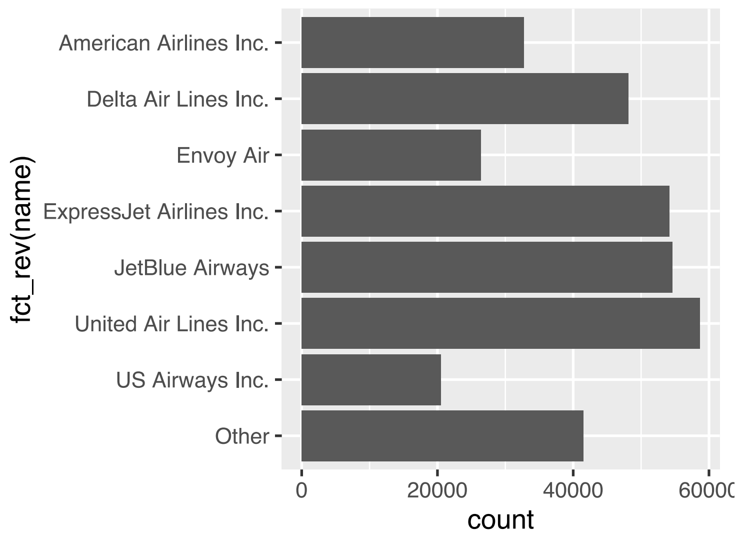
```
flight_data %>%
  mutate(
    name = fct_infreq(name)
  ) %>%
  ggplot(aes(y = fct_rev(name)))
  geom_bar()
```



Ordering by frequency is better, but do we need to show all airlines?

Flights out of New York City in 2013, with lumping

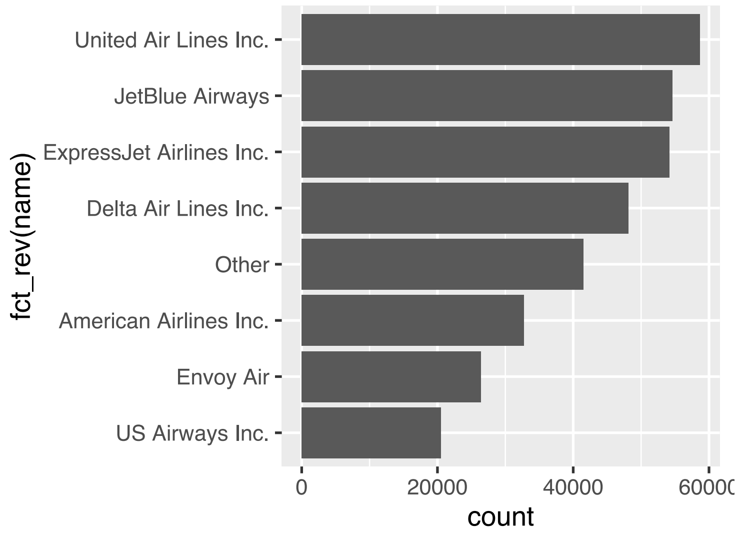
```
flight_data %>%
  mutate(
    # keep only the 7 most common
    name = fct_lump(name, 7)
  ) %>%
  ggplot(aes(y = fct_rev(name)))
  geom_bar()
```



Now the ordering is again alphabetic...

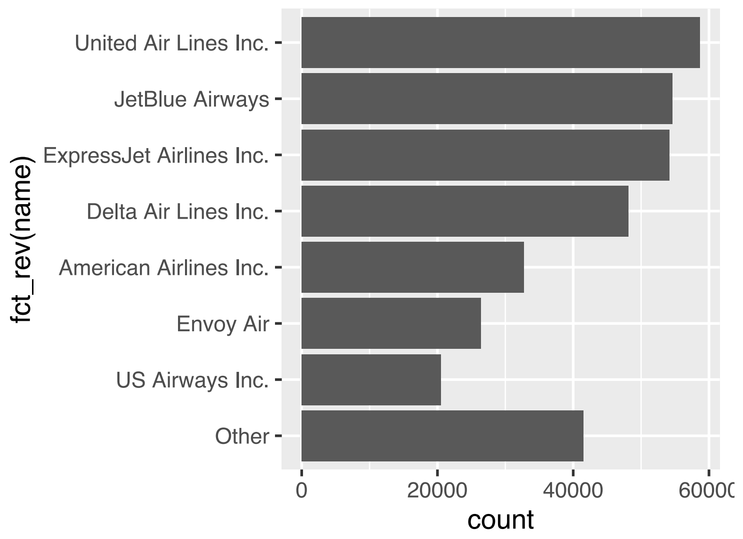
Flights out of New York City in 2013, with lumping

```
flight_data %>%
  mutate(
    # order after lumping
    name = fct_infreq(fct_lump(na
  ) %>%
  ggplot(aes(y = fct_rev(name)))
  geom_bar()
```



Flights out of New York City in 2013, with lumping

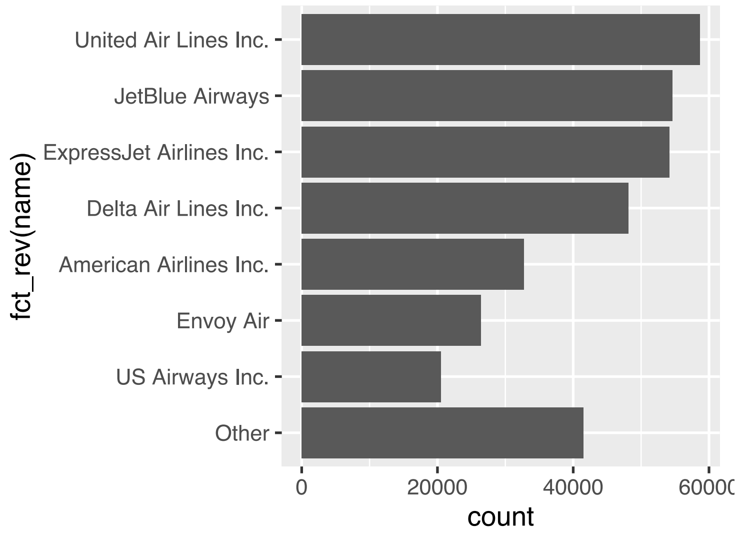
```
flight_data %>%
  mutate(
    # order before lumping
    name = fct_lump(fct_infreq(na
  ) %>%
  ggplot(aes(y = fct_rev(name)))
  geom_bar()
```



In most cases, you will want to order before lumping

Flights out of New York City in 2013, with lumping

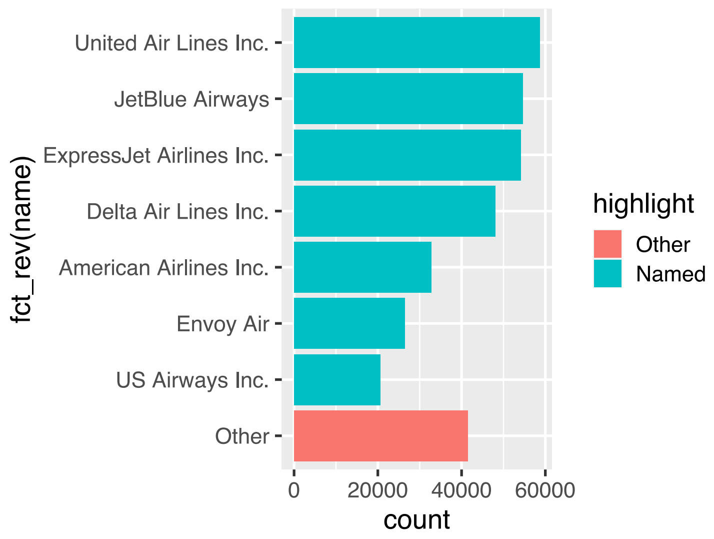
```r
flight_data %>%
  mutate(
    # order before lumping
    name = fct_lump(fct_infreq(na
  ) %>%
  ggplot(aes(y = fct_rev(name)))
  geom_bar()
```



Can we visually separate the "Other" category?

Flights out of New York City in 2013, with lumping

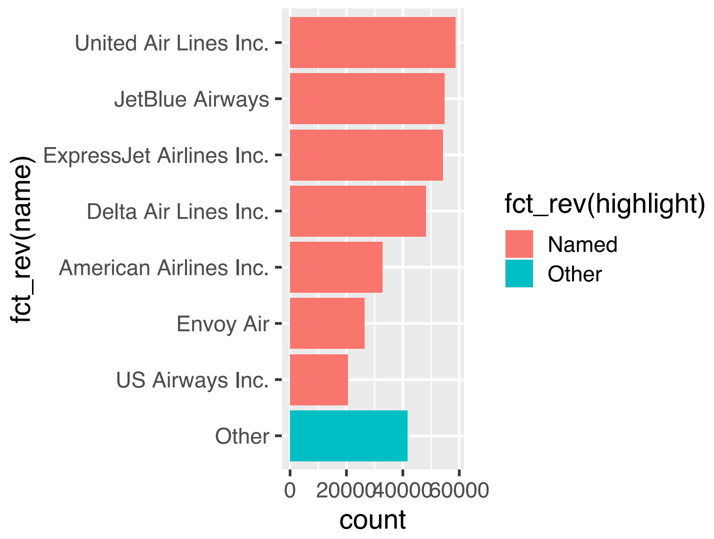
```r
flight_data %>%
  mutate(
    name = fct_lump(fct_infreq(na
    # Use `fct_other()` to manual
    # levels not called "Other" i
    highlight = fct_other(
      name,
      keep = "Other", other_level
    )
  ) %>%
  ggplot() +
  aes(
    y = fct_rev(name),
    fill = highlight
  ) +
  geom_bar()
```



One annoying issue: The legend is in the wrong order

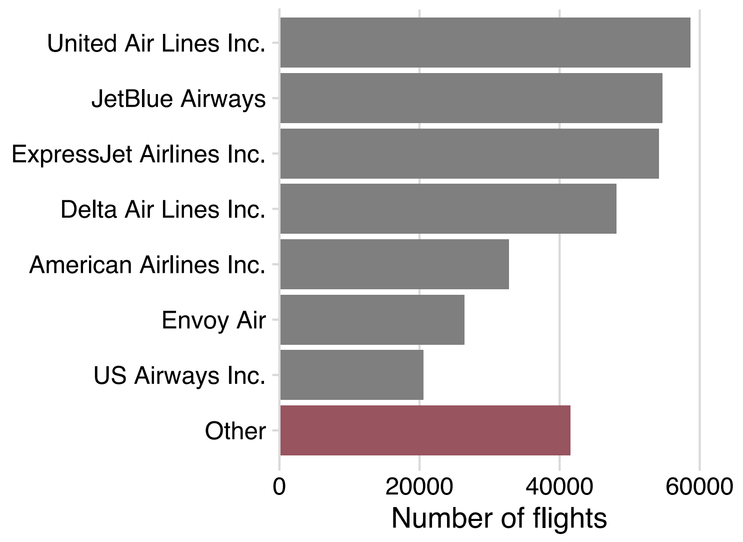
Flights out of New York City in 2013, with lumping

```
flight_data %>%
  mutate(
    name = fct_lump(fct_infreq(na
    # Use `fct_other()` to manual
    # levels not called "Other" i
    highlight = fct_other(
      name,
      keep = "Other", other_level
    )
  ) %>%
  ggplot() +
  aes(
    y = fct_rev(name),
    # reverse fill aesthetic
    fill = fct_rev(highlight)
  ) +
  geom_bar()
```



Flights out of New York City in 2013, final tweaks

```
flight_data %>%
  mutate(
    name = fct_lump(fct_infreq(name
    highlight = fct_other(
      name, keep = "Other", other_l
    )
  ) %>%
  ggplot() +
  aes(y = fct_rev(name), fill = hig
  geom_bar() +
  scale_x_continuous(
    name = "Number of flights",
    expand = expansion(mult = c(0, 
  ) +
  scale_y_discrete(name = NULL) +
  scale_fill_manual(
    values = c(
      Named = "gray50", Other = "#9
    ),
    guide = "none"
  ) +
```



Summary of key factor manipulation functions

Function	Use case	Documentation
`fct_relevel()`	Change order of factor levels manually	click here
`fct_infreq()`	Put levels in descending order of how frequently each level occurs in the data	click here
`fct_rev()`	Reverse the order of factor levels	click here
`fct_reorder()`	Put levels in ascending order determined by a numeric variable or summary function	click here
`fct_lump_n()`	Retain the *n* most frequent levels and lump all others into `"Other"`	click here
`fct_other()`	Manually group some factor levels into `"Other"`	click here

Further reading

- Fundamentals of Data Visualization: Chapter 6: Visualizing amounts
- **forcats** documentation: Introduction to forcats
- **forcats** reference documentation: Change order of levels