Assignment 7

Adrian Bracher (Matr. Nr. 01637180)

12.05.2021

Contents

1 Introduction to Factor Variables

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.1.0     v dplyr   1.0.5
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

1.1 Recognizing factor variables

We start this chapter by simply printing out the data frame and then checking if a column of the data frame contains data of type factor.

```
# Print out the dataset
print(multiple_choice_responses)

# Check if CurrentJobTitleSelect is a factor
is.factor(multiple_choice_responses$CurrentJobTitleSelect)
```

1.2 Getting number of levels

The tidyverse functions mutate_if(), summarise_all() and gather() are introduced in this exercise. The comments describe how they are applied in the code below.

```
# Change all the character columns to factors
responses_as_factors <- multiple_choice_responses %>%
    mutate_if(is.character, as.factor)

number_of_levels <- responses_as_factors %>%
    # apply the function nlevels to each column
    summarise_all(nlevels) %>%
    # change the dataset from wide to long
    gather(variable, num_levels)
```

1.3 Examining number of levels

In the code below we use top_n to select the 3 rows with the highest number of levels. pull() is used to extract a column and removing the name, which leaves the values from the column.

```
# Select the 3 rows with the highest number of levels
number_of_levels %>%
    top_n(3, num_levels)
```

```
##                 variable num_levels
## 1  CurrentJobTitleSelect         16
## 2       EmployerIndustry         16
## 3 MLMethodNextYearSelect         25
```

```
number_of_levels %>%
    # filter for where the column called variable equals CurrentJobTitleSelect
    filter(variable == "CurrentJobTitleSelect") %>%
    # pull num_levels
    pull(num_levels)
```

```
## [1] 16
```

1.4 Examining levels

In this exercise we call the function levels() to get all (distinct) levels of the input column.

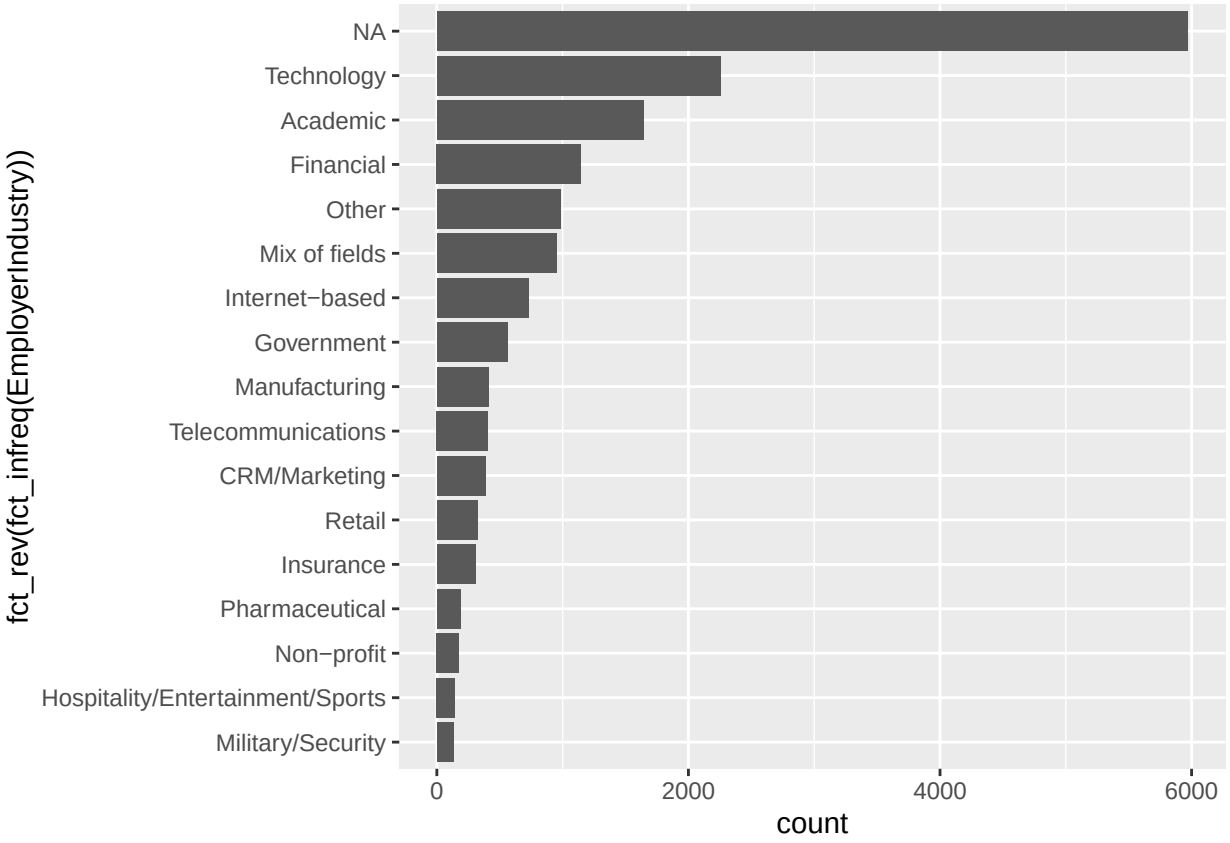
```
responses_as_factors %>%
    # pull CurrentJobTitleSelect
    pull(CurrentJobTitleSelect) %>%
    # get the values of the levels
    levels()
```

```
##  [1] "Business Analyst"
##  [2] "Computer Scientist"
##  [3] "Data Analyst"
##  [4] "Data Miner"
##  [5] "Data Scientist"
##  [6] "DBA/Database Engineer"
##  [7] "Engineer"
##  [8] "Machine Learning Engineer"
##  [9] "Operations Research Practitioner"
## [10] "Other"
## [11] "Predictive Modeler"
## [12] "Programmer"
## [13] "Researcher"
## [14] "Scientist/Researcher"
## [15] "Software Developer/Software Engineer"
## [16] "Statistician"
```

1.5 Reordering a variable by its frequency

The package "forcats" allows us to modify ggplots in various ways. Here, we first flip the coordinates with "+ coord_flip()" and call fct_infreq() to order the data and then fct_rev() to reverse the order.

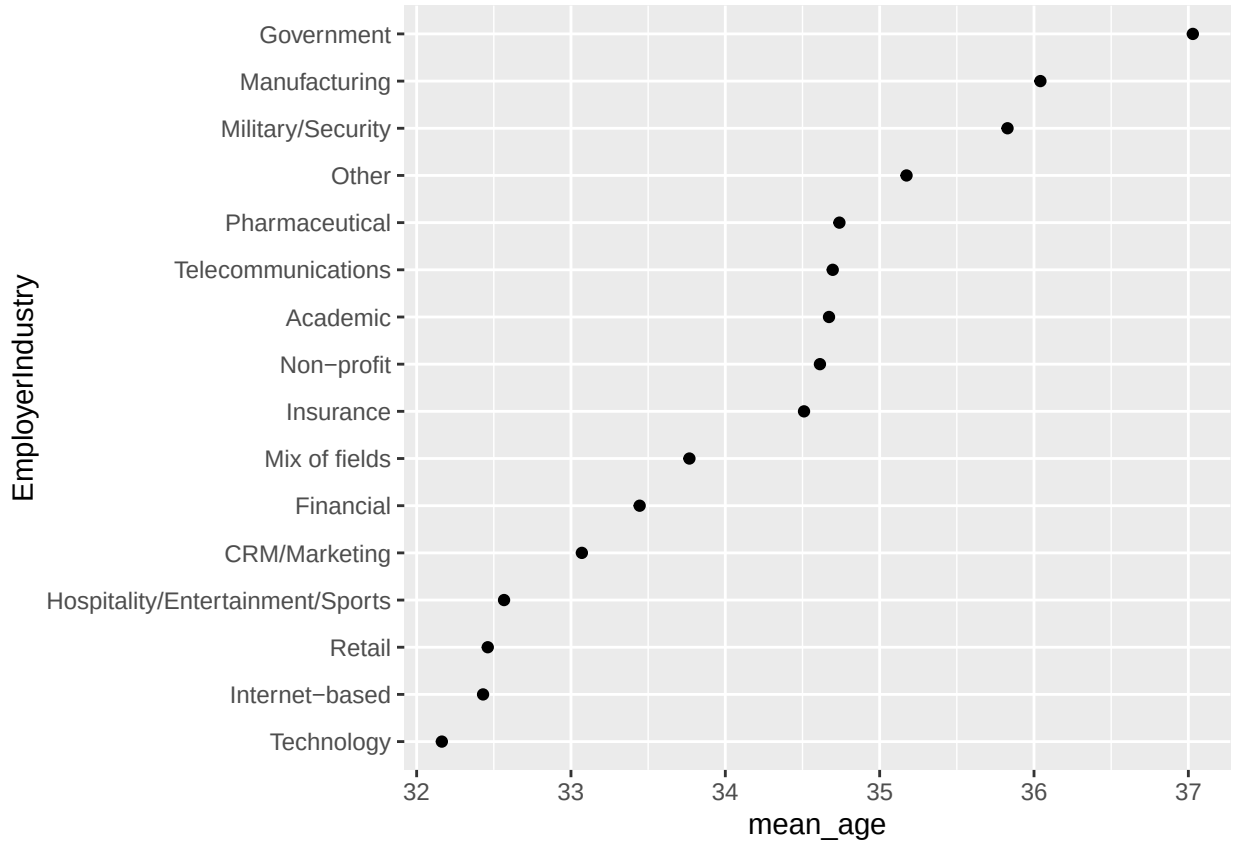
```
# Make a bar plot
ggplot(multiple_choice_responses, aes(x=fct_rev(fct_infreq(EmployerIndustry)))) +
    geom_bar() +
    # flip the coordinates
    coord_flip()
```



1.6 Ordering one variable by another

As the final exercise in this chapter we create a plot. We use mutate(), in combination with fct_reorder() to reorder EmployerIndustry and add a column mean_age with summarise().

```
multiple_choice_responses %>%
  # remove NAs
  filter(!is.na(EmployerIndustry) & !is.na(Age)) %>%
  # get mean_age by EmployerIndustry
  group_by(EmployerIndustry) %>%
  summarise(mean_age = mean(Age)) %>%
  # reorder EmployerIndustry by mean_age
  mutate(EmployerIndustry = fct_reorder(EmployerIndustry, mean_age)) %>%
  # make a scatterplot of EmployerIndustry by mean_age
  ggplot(aes(x = EmployerIndustry, y = mean_age)) +
    geom_point() +
    coord_flip()
```



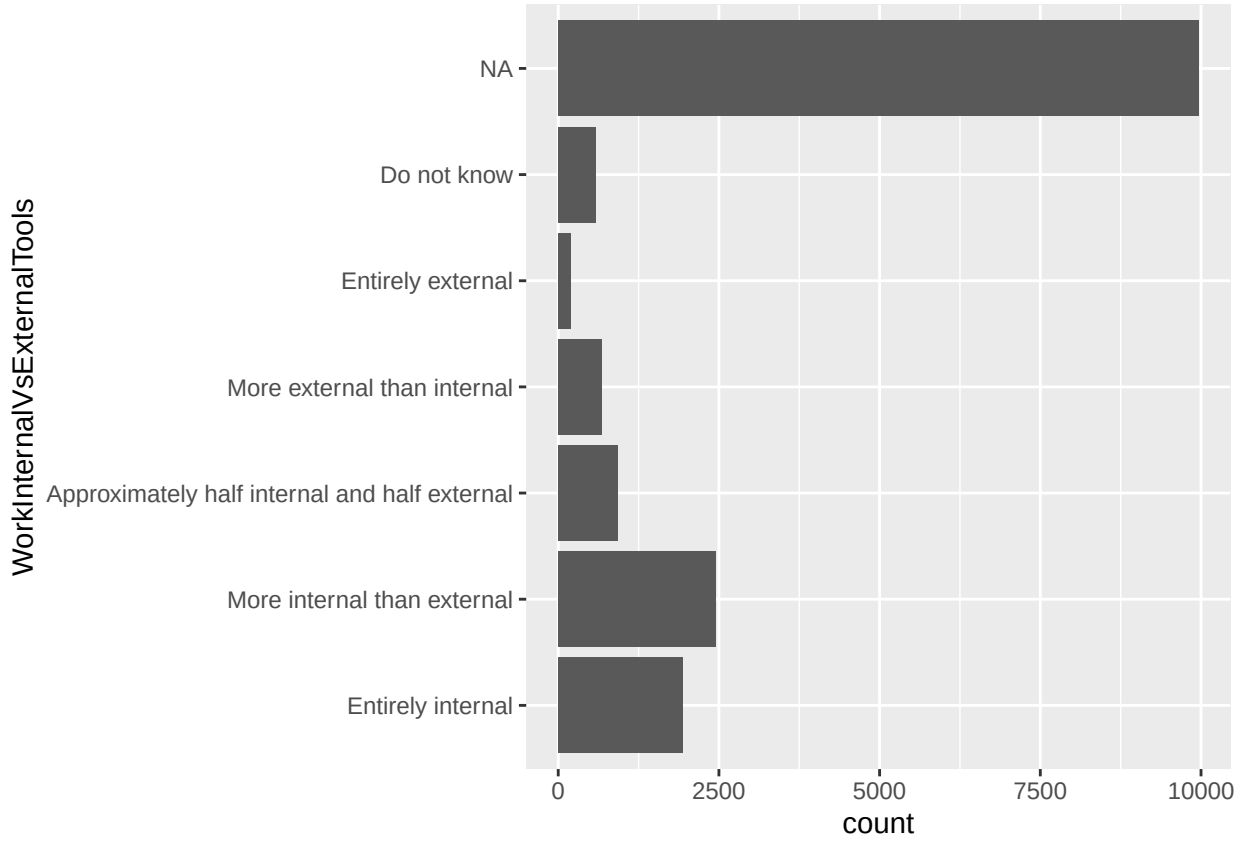
Manipulating Factor Variables ## Changing the order of factor levels Here we learn how to use fct_relevel to manually change the order of factor levels. Then, we make a bar plot of the results.

```
# Get the levels of WorkInternalVsExternalTools
levels(multiple_choice_responses$WorkInternalVsExternalTools)
```

```
## [1] "Approximately half internal and half external"
## [2] "Do not know"
## [3] "Entirely external"
## [4] "Entirely internal"
## [5] "More external than internal"
```

```
## [6] "More internal than external"
```

```
# Reorder the levels from internal to external
mc_responses_reordered <- multiple_choice_responses %>%
    mutate(WorkInternalVsExternalTools = fct_relevel(WorkInternalVsExternalTools,"Entirely internal", "
# Make a bar plot of the responses
ggplot(mc_responses_reordered, aes(x=WorkInternalVsExternalTools)) +
    geom_bar() +
    coord_flip()
```



Tricks of fct_relevel() The reordering of levels with fct_relevel() can take very advanced forms with multiple parameters, for example last = Inf puts the specified level at the very last.

```
multiple_choice_responses %>%
    # Move "I did not complete any formal education past high school" and "Some college/university stud
    mutate(FormalEducation = fct_relevel(FormalEducation, "I did not complete any formal education past
    # Move "I prefer not to answer" to be the last level.
    mutate(FormalEducation = fct_relevel(FormalEducation, "I prefer not to answer", after = Inf)) %>%
    # Move "Doctoral degree" to be after the 5th level
    mutate(FormalEducation = fct_relevel(FormalEducation, "Doctoral degree", after=5)) %>%
    # Examine the new level order
    pull(FormalEducation) %>%
    levels()
```

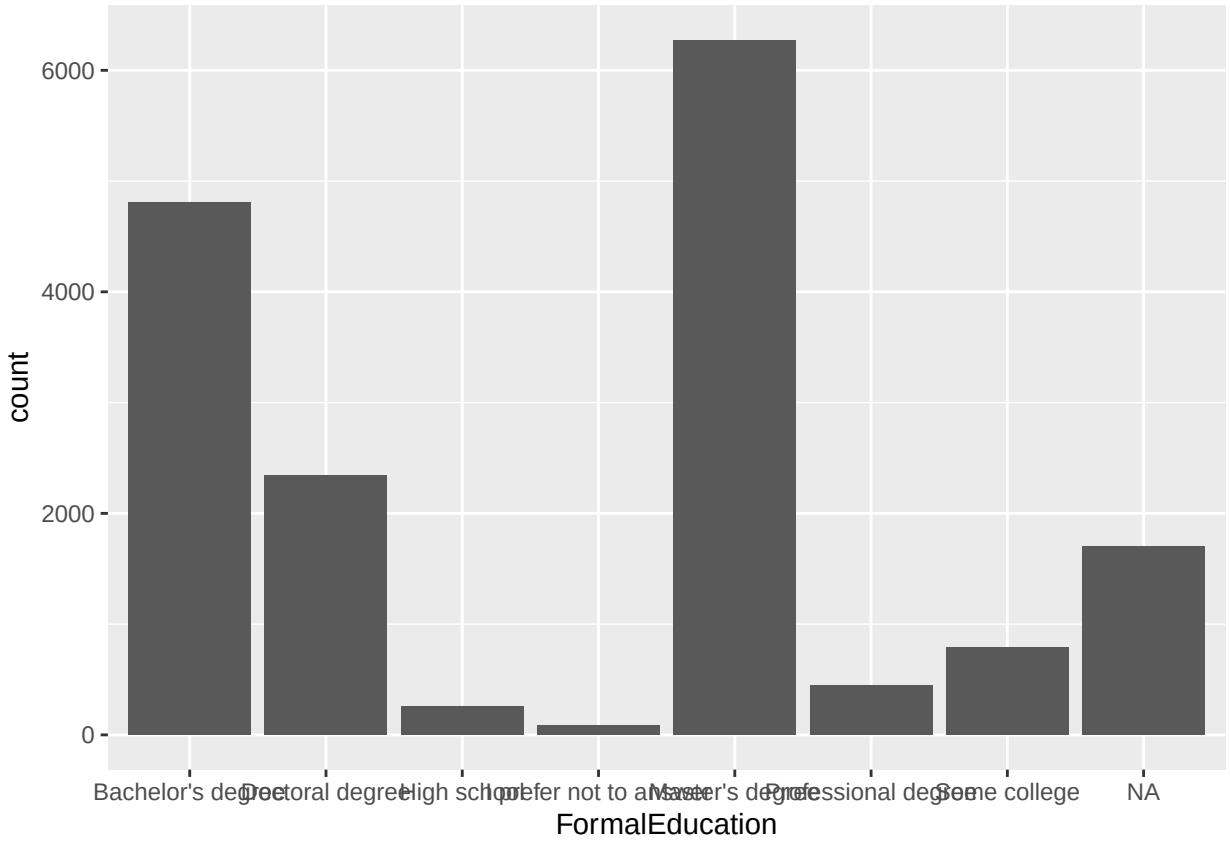
```
## [1] "I did not complete any formal education past high school"
## [2] "Some college/university study without earning a bachelor's degree"
## [3] "Bachelor's degree"
## [4] "Master's degree"
```

```
## [5] "Professional degree"
## [6] "Doctoral degree"
## [7] "I prefer not to answer"
```

1.7 Renaming a few levels

In the following we decide to shorten the value of some levels with fct_recode().

```
multiple_choice_responses %>%
    # rename the appropriate levels to "High school" and "Some college"
    mutate(FormalEducation = fct_recode(FormalEducation,
    "High school" = "I did not complete any formal education past high school",
    "Some college" = "Some college/university study without earning a bachelor's degree")) %>%
    # make a bar plot of FormalEducation
    ggplot(aes(x = FormalEducation)) +
    geom_bar()
```



Manually collapsing levels In this exercise we learn how to collapse multiple levels into one with fct_collapse() and then also collapse all the others, unspecified, into one level "Other" with fct_other().

```
multiple_choice_responses %>%
    # Create new variable, grouped_titles, by collapsing levels in CurrentJobTitleSelect
    mutate(grouped_titles = fct_collapse(CurrentJobTitleSelect,
        "Computer Scientist" = c("Programmer", "Software Developer/Software Engineer"),
        "Researcher" = "Scientist/Researcher",
        "Data Analyst/Scientist/Engineer" = c("DBA/Database Engineer", "Data Scientist",
                                              "Business Analyst", "Data Analyst",
                                              "Data Miner", "Predictive Modeler"))) %>%
```

```
    # Keep all the new titles and turn every other title into "Other"
    mutate(grouped_titles = fct_other(grouped_titles, keep = c("Computer Scientist", "Researcher", "Data
    # Get a count of the grouped titles
    count(grouped_titles)
```

```
##                    grouped_titles    n
## 1 Data Analyst/Scientist/Engineer 4928
## 2              Computer Scientist 2556
## 3                      Researcher 1597
## 4                           Other 2749
## 5                            <NA> 4886
```

1.8 Lumping variables by proportion

The function fct_lump can be used for collapsing the least common levels into "other".

```
multiple_choice_responses %>%
  # remove NAs of MLMethodNextYearSelect
  filter(!is.na(MLMethodNextYearSelect)) %>%
  # create ml_method, which lumps all those with less than 5% of people into "Other"
  mutate(ml_method = fct_lump(MLMethodNextYearSelect, prop=0.05)) %>%
  # count the frequency of your new variable, sorted in descending order
  count(ml_method, sort=TRUE)
```

```
##              ml_method    n
## 1                Other 4405
## 2        Deep learning 4362
## 3          Neural Nets 1386
## 4 Time Series Analysis  680
```

1.9 Preserving the most common levels

This time we don't collapse a fixed proportion, but all but a fixed amount of levels (in this case 5).

```
multiple_choice_responses %>%
  # remove NAs
  filter(!is.na(MLMethodNextYearSelect)) %>%
  # create ml_method, retaining the 5 most common methods and renaming others "other method"
  mutate(ml_method = fct_lump(MLMethodNextYearSelect, n=5, other_level = "other method")) %>%
  # count the frequency of your new variable, sorted in descending order
  count(ml_method, sort=TRUE)
```

```
##              ml_method    n
## 1        Deep learning 4362
## 2         other method 3401
## 3          Neural Nets 1386
## 4 Time Series Analysis  680
## 5     Bayesian Methods  511
## 6          Text Mining  493
```

2 Creating Factor Variables

2.1 Grouping and reshaping similar columns

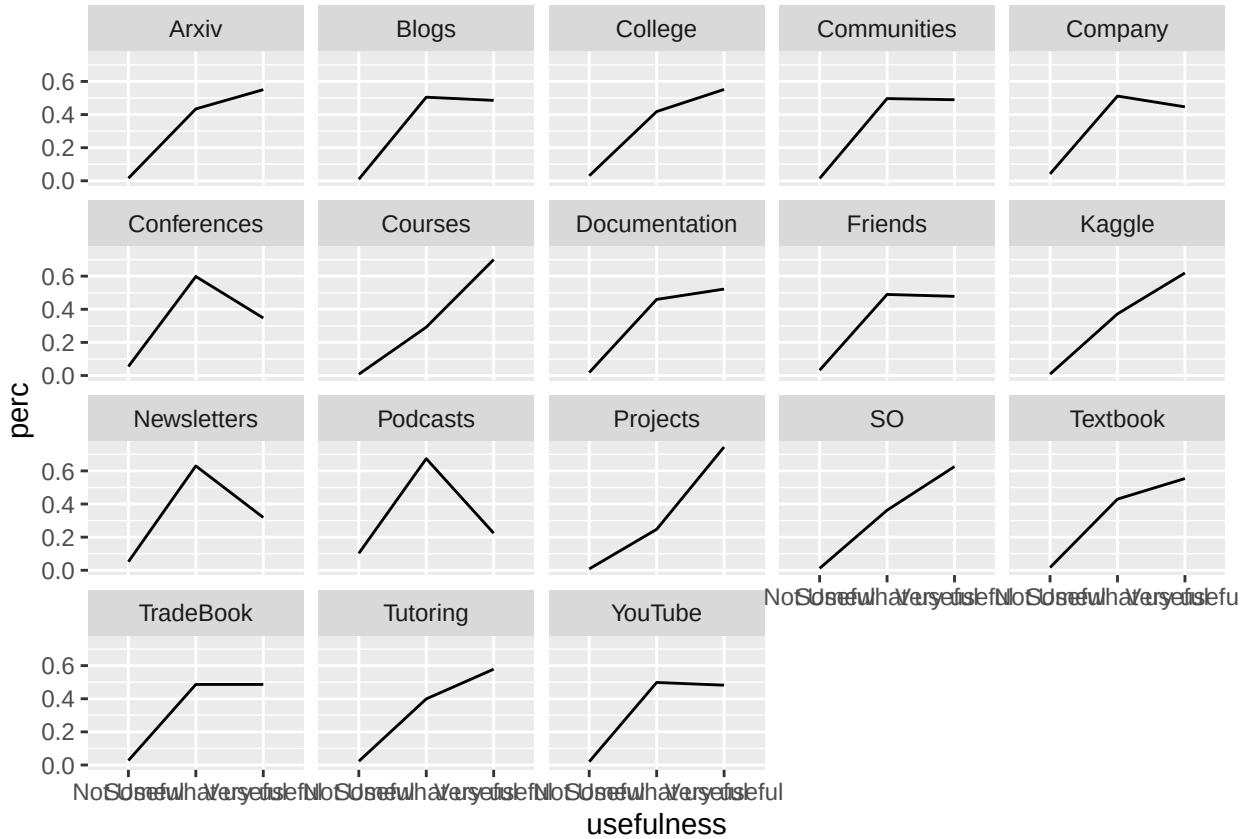
In this exercise, we learn how to use str_remove in combination with mutate.

```
learning_platform_usefulness <- multiple_choice_responses %>%
  # select columns with LearningPlatformUsefulness in title
  select(contains("LearningPlatformUsefulness")) %>%
  # change data from wide to long
  gather(learning_platform, usefulness) %>%
  # remove rows where usefulness is NA
  filter(!is.na(usefulness)) %>%
  # remove "LearningPlatformUsefulness" from each string in learning_platform
  mutate(learning_platform = str_remove(learning_platform, "LearningPlatformUsefulness"))
```

2.2 Summarizing data

In the code below we create a faceted plot with the dplyr function add_count().

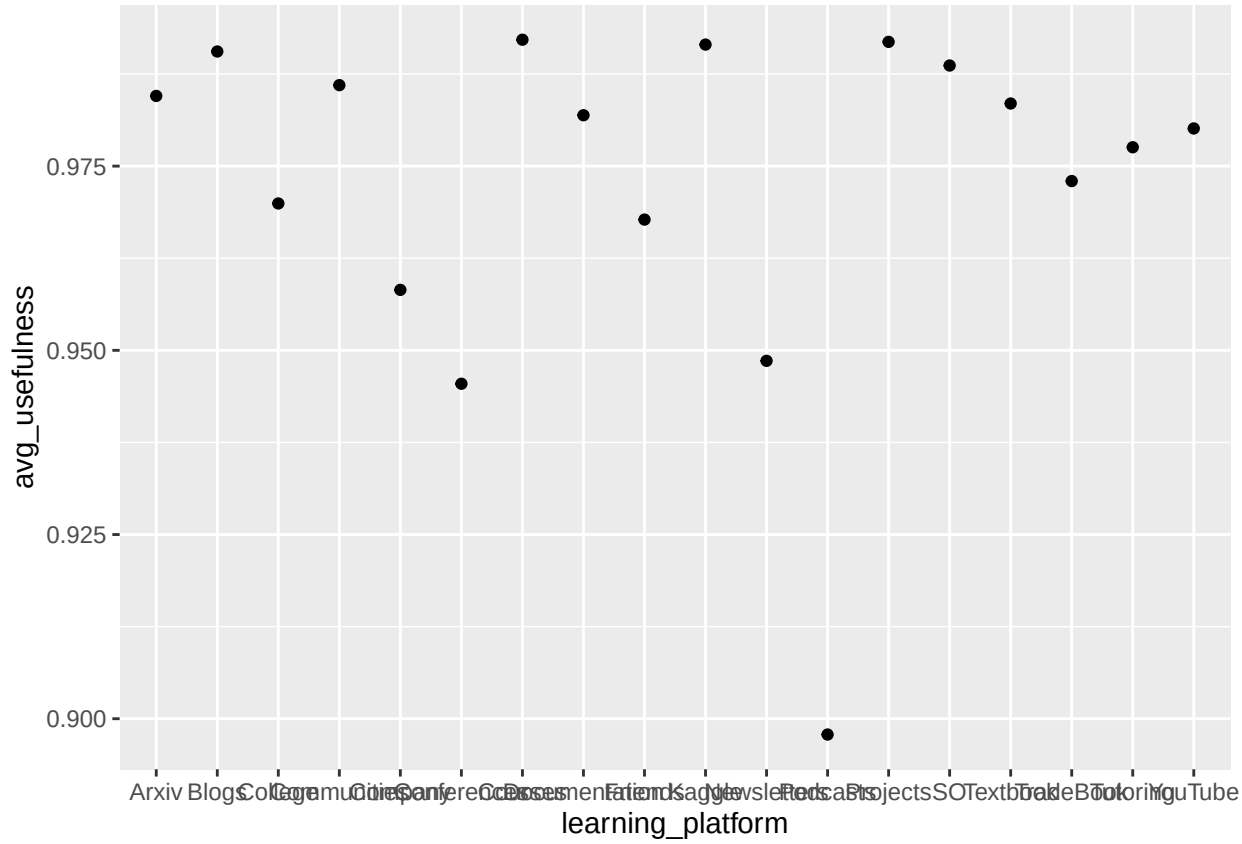
```
perc_useful_platform <- learning_platform_usefulness %>%
  # change dataset to one row per learning_platform usefulness pair with number of entries for each
  count(learning_platform, usefulness) %>%
  # use add_count to create column with total number of answers for that learning_platform
  add_count(learning_platform, wt = n, name='nn') %>%
  # create a new column, perc, that is the percentage of people giving that response for that learning_
  mutate(perc = n / nn)

# create a line graph for each question with usefulness on x-axis and percentage of responses on y
ggplot(perc_useful_platform, aes(x = usefulness, y = perc, group = learning_platform)) +
  geom_line() +
  facet_wrap(~ learning_platform)
```



2.3 Creating an initial plot

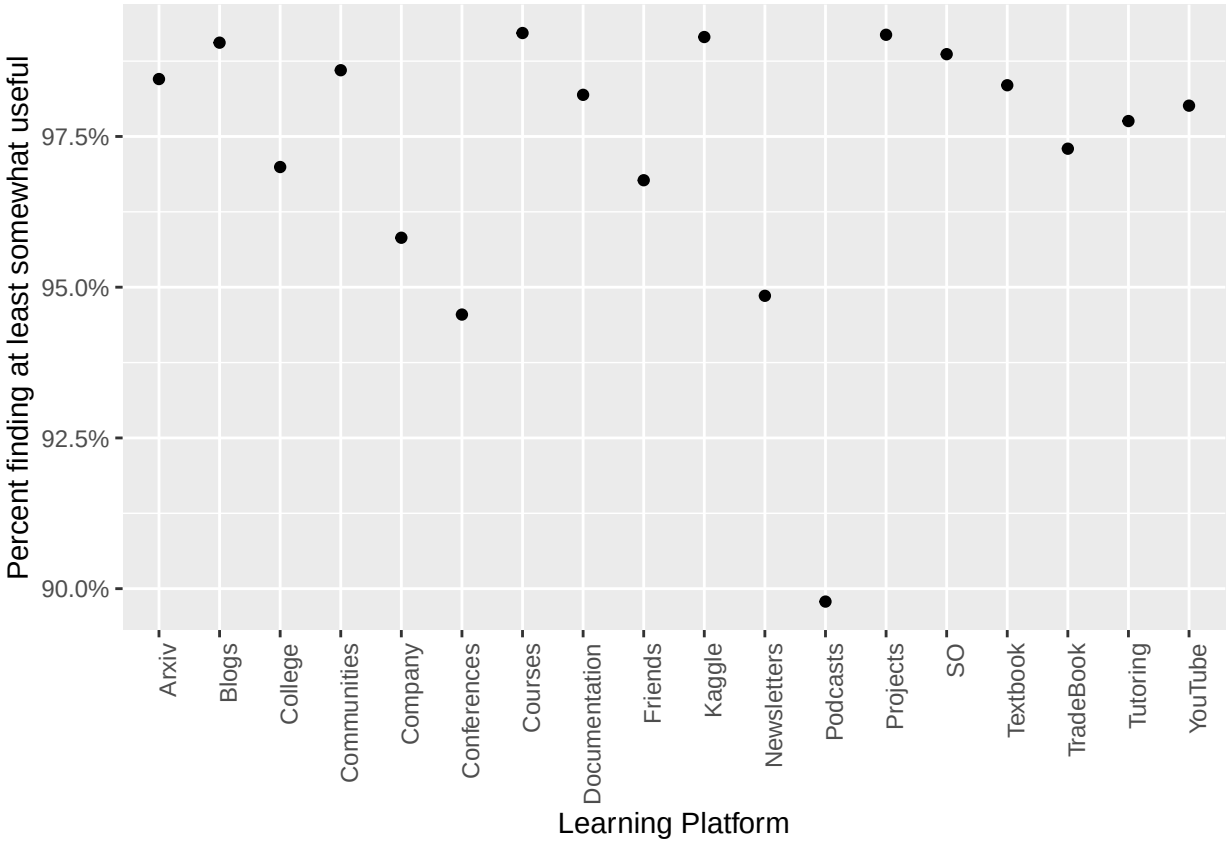
In this exercise we create a plot using various dplyr functions, and the function if_else() is introduced.

```
usefulness_by_platform <- learning_platform_usefulness %>%
    # If usefulness is "Not Useful", make 0, else 1
    mutate(usefulness = if_else(usefulness == "Not Useful", 0, 1)) %>%
    # Group by learning platform
    group_by(learning_platform) %>%
    # Summarize the mean usefulness for each platform
    summarise(avg_usefulness = mean(usefulness))
# Make a scatter plot of average usefulness by learning platform
ggplot(usefulness_by_platform, aes(x=learning_platform, y=avg_usefulness)) +
       geom_point()
```



Editing plot text In this exercise we learn how to rotate x-axis test, rename the axis labels and change the axis scale.

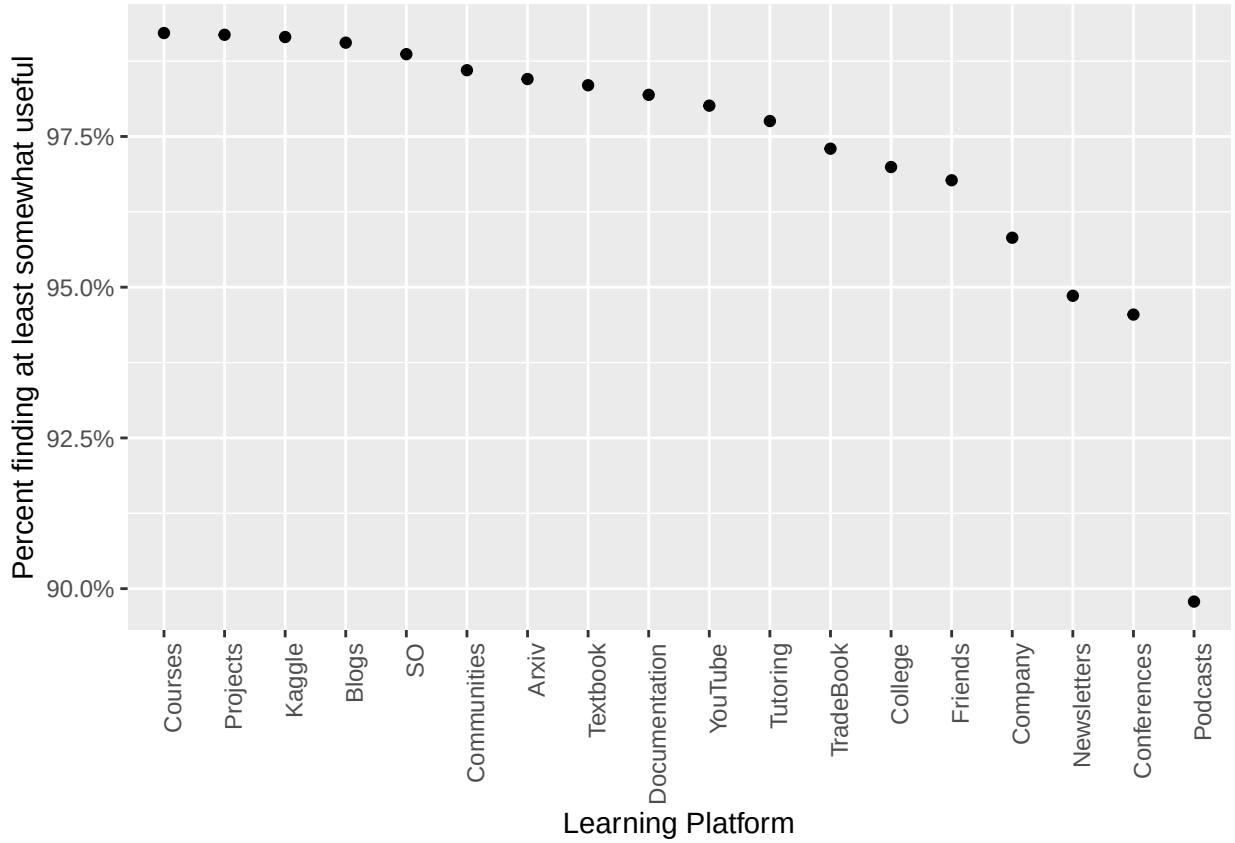
```
ggplot(usefulness_by_platform, aes(x = learning_platform, y = avg_usefulness)) +
    geom_point() +
    # rotate x-axis text by 90 degrees
    theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
    # rename y and x axis labels
    labs(x = "Learning Platform", y = "Percent finding at least somewhat useful") +
    # change y axis scale to percentage
    scale_y_continuous(labels = scales::percent)
```



2.4 Reordering graphs

Here we learn how to apply fct_reorder and fct_rev inside a mutate call.

```
usefulness_by_platform %>%
    # reorder learning_platform by avg_usefulness
    mutate(learning_platform = fct_reorder(learning_platform, avg_usefulness)) %>%
    # reverse the order of learning_platform
    mutate(learning_platform = fct_rev(learning_platform)) %>%
    ggplot(aes(x = learning_platform, y = avg_usefulness)) +
    geom_point() +
    theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
    labs(x = "Learning Platform", y = "Percent finding at least somewhat useful") +
    scale_y_continuous(labels = scales::percent)
```



case_when() with single variable In this exercise we learn how to use the function case_when().

```
# Check the min age
min(multiple_choice_responses$Age, na.rm=TRUE)
```

```
## [1] 0
```

```
# Check the max age
max(multiple_choice_responses$Age, na.rm=TRUE)
```

```
## [1] 100
```

```
multiple_choice_responses %>%
    # Filter for rows where Age is between 10 and 90
    filter(between(Age, 10, 90)) %>%
    # Create the generation variable based on age
    mutate(generation = case_when(
    between(Age, 10, 22) ~ "Gen Z",
    between(Age, 23, 37) ~ "Gen Y",
    between(Age, 38, 52) ~ "Gen X",
    between(Age, 53, 71) ~ "Baby Boomer",
    between(Age, 72, 90) ~ "Silent"
    )) %>%
    # Get a count of how many answers in each generation
    count(generation)
```

```
##    generation     n
## 1 Baby Boomer   832
## 2       Gen X  3162
```

```
## 3       Gen Y 10281
## 4       Gen Z  2037
## 5      Silent    37
```

2.5 case_when() from multiple columns

We extend our knowledge from the previous exercise using case_when to multiple columns.

```
multiple_choice_responses %>%
    # Filter out people who selected Data Scientist as their Job Title
    filter(CurrentJobTitleSelect != "Data Scientist") %>%
    # Create a new variable, job_identity
    mutate(job_identity = case_when(
        CurrentJobTitleSelect == "Data Analyst" &
        DataScienceIdentitySelect == "Yes" ~ "DS analysts",
        CurrentJobTitleSelect == "Data Analyst" &
        DataScienceIdentitySelect %in% c("No", "Sort of (Explain more)") ~ "NDS analyst",
        CurrentJobTitleSelect != "Data Analyst" &
        DataScienceIdentitySelect == "Yes" ~ "DS non-analysts",
        TRUE ~ "NDS non analysts")) %>%
    # Get the average job satisfaction by job_identity, removing NAs
    group_by(job_identity) %>%
    summarise(avg_js = mean(JobSatisfaction, na.rm=TRUE))
```

```
## # A tibble: 4 x 2
##   job_identity     avg_js
##   <chr>             <dbl>
## 1 DS analysts        6.44
## 2 DS non-analysts    6.93
## 3 NDS analyst        6.14
## 4 NDS non analysts   6.43
```

3 Case Study on Flight Etiquette

3.1 Changing characters to factors

In the code below we use mutate_if to convert all characters to factors

```
flying_etiquette %>%
    # Change characters to factors
    mutate_if(is.character, as.factor) %>%
    # Filter out those who have never flown on a plane
    filter('How often do you travel by plane?' != "Never")
```

3.2 Tidying data

We apply select, contains, gather and filter to the flying_etiquette data frame in order to tidy them.

```
gathered_data <- flying_etiquette %>%
    mutate_if(is.character, as.factor) %>%
    filter('How often do you travel by plane?' != "Never") %>%
    # Select columns containing "rude"
    select(contains("rude")) %>%
    # Change format from wide to long
    gather(response_var, value)
```

```
## Warning: attributes are not identical across measure variables;
## they will be dropped
```

3.3 Cleaning up strings

The function str_remove can also be used with regex expressions like ".*" as you can see in the following:

```
gathered_data %>%
    # Remove everything before and including "rude to " (with that space at the end!)
    mutate(response_var = str_remove(response_var, ".*rude to ")) %>%
    # Remove "on a plane"
    mutate(response_var = str_remove(response_var, "on a plane"))
```

3.4 Dichotomizing variables

Here we use if_else, str_replace and mutate to make a new column called rude, which contains a binary value 1 or 0 to distinguish rudeness.

```
dichotimized_data <- gathered_data %>%
    mutate(response_var = str_replace(response_var, '.*rude to ', '')) %>%
    mutate(response_var = str_replace(response_var, 'on a plane', '')) %>%
    # Remove rows that are NA in the value column
    filter(!is.na(value)) %>%
    # Dichotomize the value variable to make a new variable, rude
    mutate(rude = if_else(value %in% c('No, not rude at all', 'No, not at all rude'), 0, 1))
```

3.5 Summarizing data

Again, we apply summarise to add a new column perc_rude that aggregates another column.

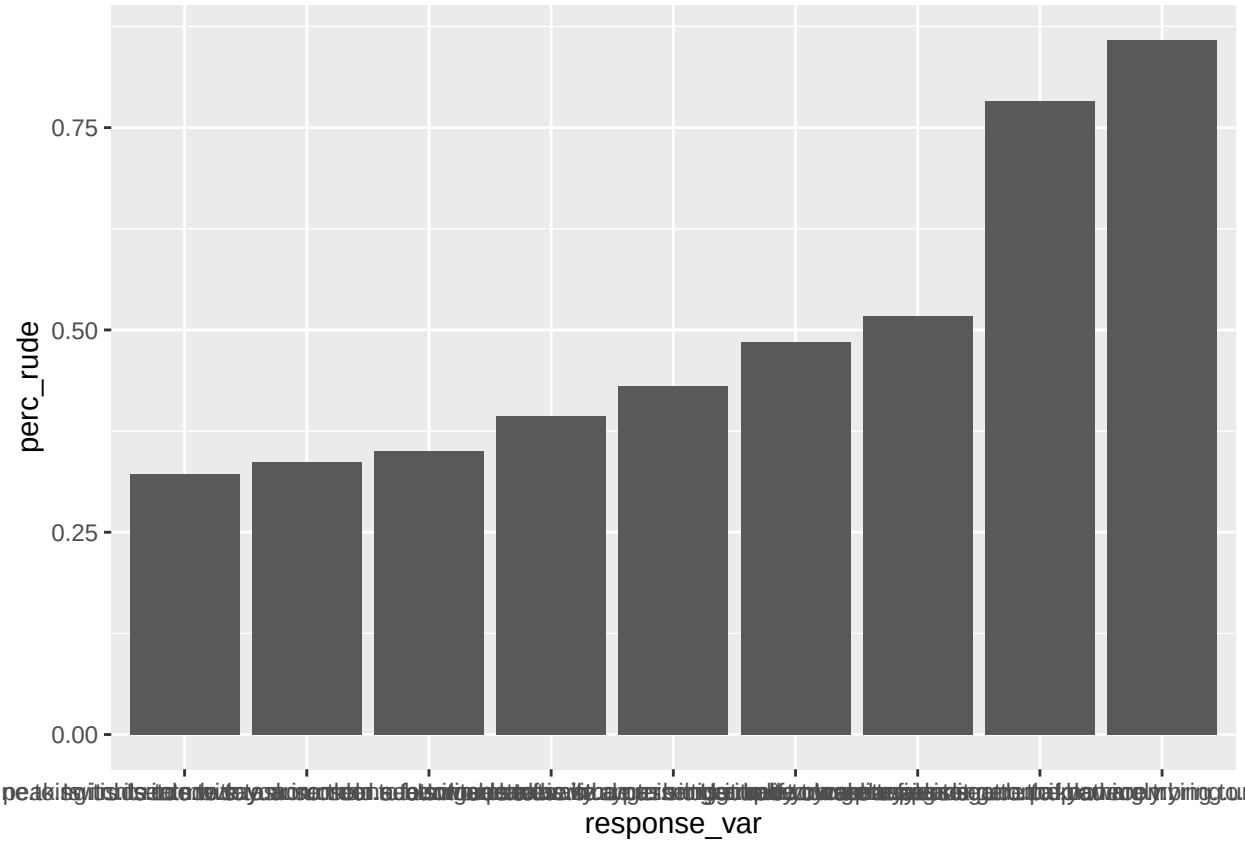
```
rude_behaviors <- gathered_data %>%
    mutate(response_var = str_replace(response_var, '.*rude to ', '')) %>%
    mutate(response_var = str_replace(response_var, 'on a plane', '')) %>%
    # Remove rows that are NA in the value column
    filter(!is.na(value)) %>%
    mutate(rude = if_else(value %in% c("No, not rude at all", "No, not at all rude"), 0, 1)) %>%
    # Group by response_var
    group_by(response_var) %>%
    # Create perc_rude, the percent considering each behavior rude
    summarise(perc_rude = mean(rude))

rude_behaviors
```

```
## # A tibble: 9 x 2
##   response_var                                                  perc_rude
##   <chr>                                                             <dbl>
## 1 Generally.speaking..is.it.rude.to.say.more.than.a.few.words.tothe.s~     0.351
## 2 In.general..is.it.rude.to.knowingly.bring.unruly.children.on.a.plan~     0.859
## 3 In.general..is.itrude.to.bring.a.baby.on.a.plane.                        0.431
## 4 Is.it.rude.to.ask.someone.to.switch.seats.with.you.in.order.to.be.c~     0.393
## 5 Is.it.rude.to.wake.a.passenger.up.if.you.are.trying.to.go.to.the.ba~     0.486
## 6 Is.itrude.to.ask.someone.to.switch.seats.with.you.in.order.to.be.cl~     0.322
## 7 Is.itrude.to.move.to.an.unsold.seat.on.a.plane.                          0.337
## 8 Is.itrude.to.recline.your.seat.on.a.plane.                               0.517
## 9 Is.itrude.to.wake.a.passenger.up.if.you.are.trying.to.walk.around.       0.783
```

3.6 Creating an initial plot

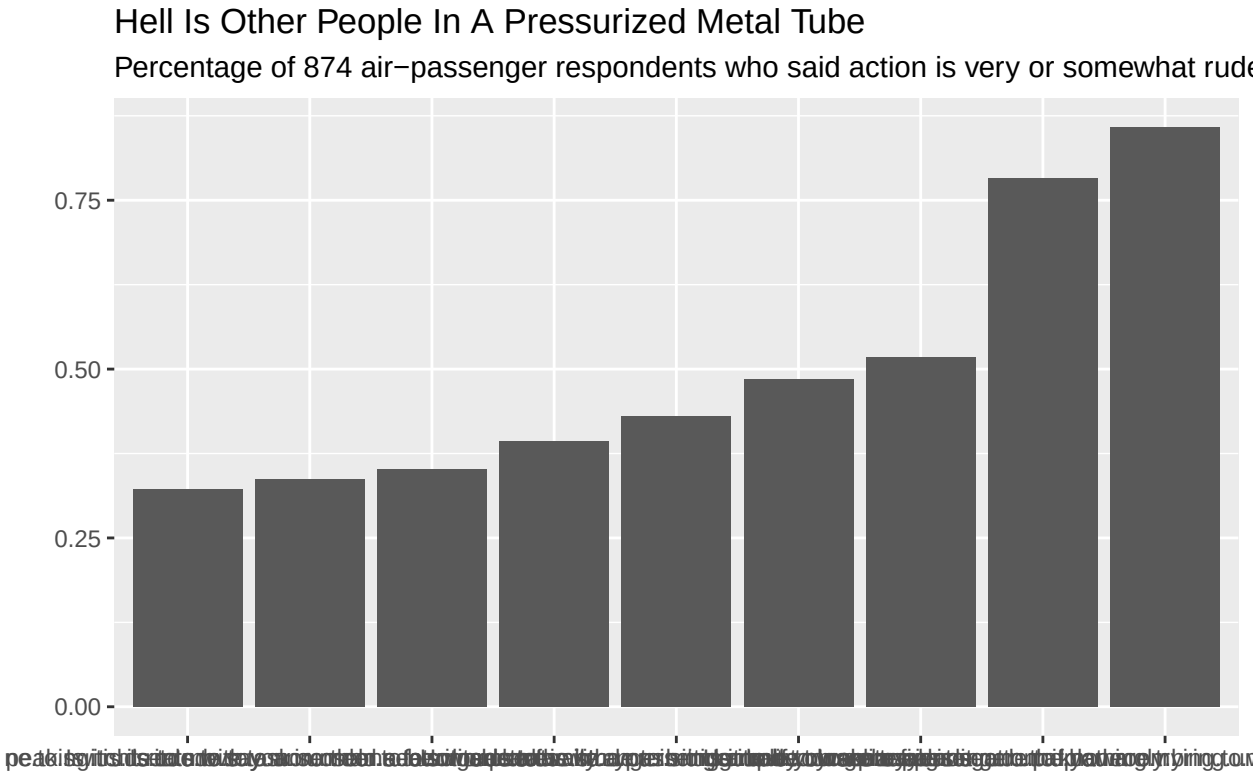
We create a simple plot using functions such as fct_reorder.

```
initial_plot <- rude_behaviors %>%
    # reorder response_var by perc_rude
    mutate(response_var = fct_reorder(response_var, perc_rude)) %>%
    # make a bar plot of perc_rude by response_var
    ggplot(aes(x = response_var, y = perc_rude)) +
    geom_col()


# View your plot
initial_plot
```



Fixing labels We use the labs function to change title, subtitle, caption, x -axis and y-axis labels.

```
titled_plot <- initial_plot +
    # Add the title, subtitle, and caption
    labs(title = "Hell Is Other People In A Pressurized Metal Tube",
         subtitle = "Percentage of 874 air-passenger respondents who said action is very or somewhat ru
         caption = "Source: SurveyMonkey Audience",
         # Remove the x- and y-axis labels
         x = "",
         y = "")

titled_plot
```



Flipping things around In this exercise we use theme with axis.text as well as axis.ticks to change the plot to our wishes.

```
flipped_plot <- titled_plot +
    # Flip the axes
    coord_flip() +
    # Remove the x-axis ticks and labels
    theme(axis.text.x = element_blank(),
        axis.ticks.x = element_blank())
```

3.7 Finalizing the chart

We set the label parameter to percent(perc_rude) to label each bar with the percent rude value.

```
library(memisc)
flipped_plot +
    # Apply percent() to perc_rude to label above the bar with the perc value
    geom_text(aes(label = percent(perc_rude),
                  y = perc_rude + .03),
              position = position_dodge(0.9),
              vjust = 1)
```