Day 5 - Data manipulation with the tidyverse

Introduction

The purpose of today's activity is to build some confidence using the tools in the `tidyverse` R package to manipulate data. To guide our exploration today, we will use a dataset[1] containing all movies and TV shows available on Netflix (as of 2022).

At the end of today's notes, you should be familiar with the following functions:

- `filter`
- `select`
- `mutate`/`case_when`
- `group_by`
- `summarize`

---

[1] Link to the data set on Kaggle.

The architecture of the tidyverse

The `tidyverse` R package is a conglomerate of R packages all developed to improve the manipulation of data in R. Librarying the `tidyverse` R package will allow the use of functions from nine different packages.

```
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.2.1     v readr     2.2.0
v forcats   1.0.1     v stringr   1.6.0
v ggplot2   4.0.3     v tibble    3.3.1
v lubridate 1.9.5     v tidyr     1.3.2
v purrr     1.2.2
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becom
```

> **!** STOP
>
> Many code chunks in this document will be set to not evaluate (`eval = F`). This is done so that you can fill in the code as you work through this document. **Remember to set `eval = T` after you fix the code in a chunk.**

Use the `read_csv` function in the `readr` library to load the `netflix.csv` data. Call the data `netflix`, then print the first six rows using `head`.

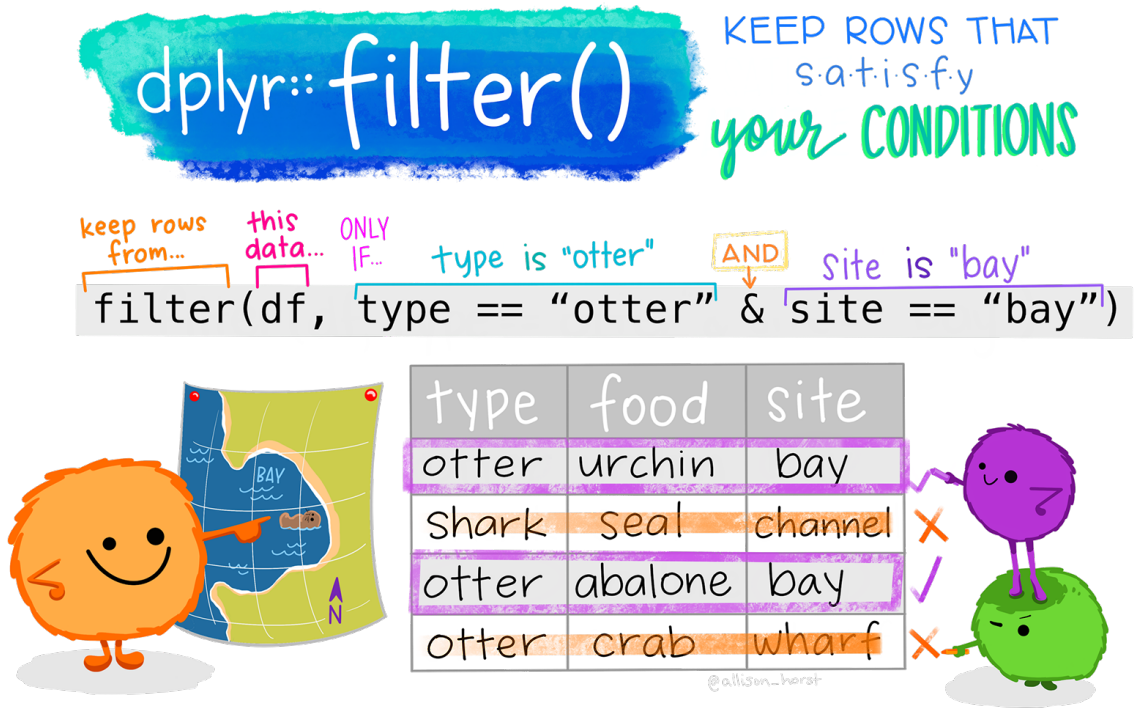
```
# read the data into R
## not sure how to use read_csv? run ?read_csv in the console
netflix <- read_csv("netflix_titles.csv")
head(netflix)
```

```
# A tibble: 6 x 12
  show_id type    title    director cast  country date_added release_year rating
  <chr>   <chr>   <chr>    <chr>    <chr> <chr>   <chr>              <dbl> <chr>
1 s1      Movie   Dick Jo~ Kirsten~ <NA>  United~ September~          2020 PG-13
2 s2      TV Show Blood &~ <NA>     Ama ~ South ~ September~          2021 TV-MA
3 s3      TV Show Ganglan~ Julien ~ Sami~ <NA>    September~          2021 TV-MA
4 s4      TV Show Jailbir~ <NA>     <NA>  <NA>    September~          2021 TV-MA
5 s5      TV Show Kota Fa~ <NA>     Mayu~ India   September~          2021 TV-MA
6 s6      TV Show Midnigh~ Mike Fl~ Kate~ <NA>    September~          2021 TV-MA
# i 3 more variables: duration <chr>, listed_in <chr>, description <chr>
```

`dplyr` verbs

> **Tidyverse verbs**
>
> Much of the data manipulation through the `tidyverse` will be handled by the `dplyr` package, which uses a number of key *verbs* to accomplish tasks in R.

The `filter` verb



Often, we wish to filter to rows that satisfy a number of conditions; this task is accomplished via the `filter` verb. Work through the examples below to familiarize yourself with filtering.

```
# filter to only movies
netflix %>%
  filter()

# filter to only movies that are rated R
## hint: a "," may be used to separate multiple conditions
netflix %>%
  filter()

# filter to movies or shows that are rated R or TV-MA
```

```
## hint: "|" may be used as "or"
netflix %>%
  filter()

## alternatively, %in% can be used to identify when a string is contained in a vector
netflix %>%
  filter()

# filter to movies with Nicolas Cage in them (greatest actor of our generation)
## hint: grepl("Nicolas Cage", cast) returns TRUE if "Nicolas Cage" is in the cast
netflix %>%
  filter()

# filter to movies that are listed under categories including crime
netflix %>%
  filter()
```

The `select` verb

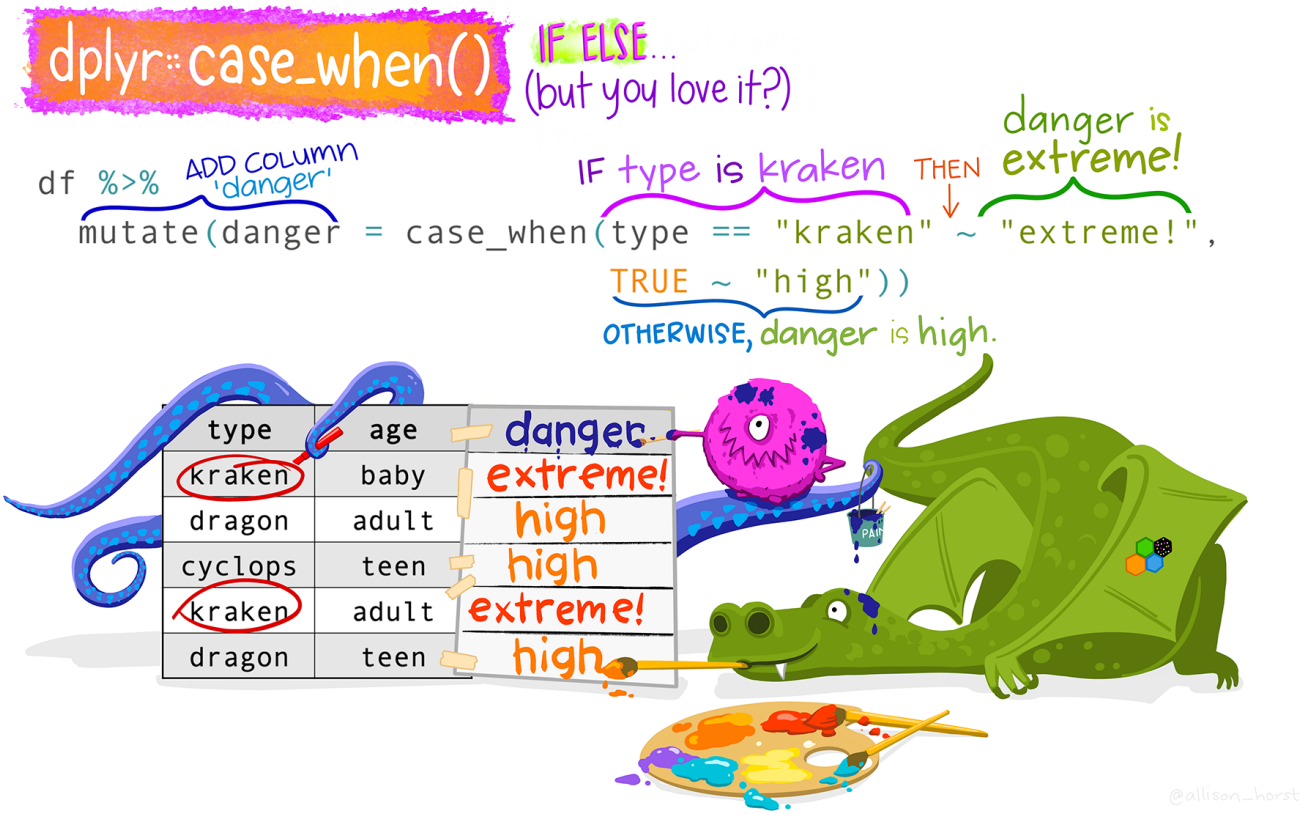
The `filter` verb allows us to filter to specific *rows*. The `select` verb allows us to select specific *columns*.

```
# select the title and description columns
netflix %>%
  dplyr::select()

# select columns that start with the letter d
# hint: ?starts_with
netflix %>%
  dplyr::select()
```

The `mutate` verb

The `mutate` verb allows us to add another column to a data set. It is often combined with `case_when` to create categorical variables.




```
# add a column called source that indicates the rows of these data are from netflix
netflix_updated <- netflix %>%
  mutate() %>%
  dplyr::select(source, everything())
netflix_updated

# add a column called post2000s that denotes whether the movie/show released after 1999
netflix_updated <- netflix %>%
  mutate(
    post2000s = case_when(

    )
  ) %>%
  dplyr::select(post2000s, everything())
netflix_updated

# create a new column that denotes whether the movie/show is available in US
netflix_updated <- netflix %>%
  mutate() %>%
  dplyr::select(, everything())
netflix_updated
```

```r
# challenge: add a column containing the first member of the cast
## there are a couple of different ways to do this - ChatGPT could help!
netflix_updated <- netflix %>%
  mutate() %>%
  dplyr::select(, everything())
netflix_updated
```

The `summarize` and `group_by` verbs

Statistics is often about making comparisons. When we want to compute summary statistics by a grouping variable, we often use `summarize` in conjunction with `group_by`.

```
# calculate the number of times each rating appears in the data
## hint: the function n() can be used to count the number of rows
netflix %>%
  group_by() %>%
  summarize()

# calculate the mean and standard deviation of year by type
## (it doesn't make a ton of sense to do this)
netflix %>%
  group_by() %>%
  summarize()
```