

Exploratory data analysis

Week 4, Lecture 7

<https://live.ds306.org>

September 22, 2026

Recap: reviewer behavior in MovieLens

- We built pipelines to summarize ratings by movie and by reviewer.
- We compared reviewers' rating distributions.
- We considered how reviewers' habits affect movie averages.

Today we will explore a new dataset from [IMDb](#).

Exploratory data analysis

When we inspect and summarize data, look for patterns, and use what we find to decide what to investigate next, we are performing **exploratory data analysis**.

Types of EDA questions we'll examine:

- What **variation** occurs within a variable?
- What **covariation** occurs between variables?
- What **unusual values** or **outliers** exist within the data?

Today's dataset

```
glimpse(imdb)
```

- 7,072 movies on IMDb that have at least 25,000 votes. Each row is a movie.
- `imdb_rating` is the score that IMDb publishes.
- The data are in wide form.

Warmup 1: Find a movie you know

Find your favorite recent film. Submit its title, year, total votes, and IMDb rating. Use backticks for column names with spaces, such as `Total ratings`.

```
imdb |>
```

```
  filter(Title == "_____")
```

Report your film's title, release year, total votes, and IMDb rating.

Warmup 2: Which movies attracted the most votes?

Find the **five most-voted** movies released in 2023. Show title and total votes.

```
imdb |>  
  
  filter(_____) |>  
  
  arrange(_____) |>  
  
  slice_head(n = _____) |>  
  select(Title, `Total ratings`)
```

Which five movies released in 2023 have the most votes? Report their titles and vote counts.

Warmup 3: Percentage of one-star votes

For movies released in 2023, calculate the **percentage of votes that are one star**. Find the five highest percentages.

Write your pipeline a step at a time. Run it to inspect the results; click **Run Code** to request checklist feedback.

```
imdb |> head()
```

- Keep only movies released in 2023.
- Calculate each movie's percentage of one-star ratings.
- Order movies from highest to lowest one-star percentage.

Questions for today

- Where do most movie scores fall?
- Which movies have the largest gap between their raw average and IMDb score?
- Do older movies have higher ratings?

What are we plotting?

```
imdb |>
  select(Title, Year, imdb_rating, `Total ratings`) |>
  head()
```

- One row is a movie. This extract has 7,072 movies, each with over 25,000 votes.
- `imdb_rating` is IMDb's published score.

Marginal and joint distributions

- A **marginal distribution** describes one variable: e.g., the distribution of IMDb scores.
- A **joint distribution** describes two variables together: e.g., IMDb scores vs decade of release.
- A **conditional distribution** restricts the comparison to a group: e.g., IMDb scores among movies from the 1990s.

One continuous variable

Where do most movies fall?

A histogram visualizes the variation within a single continuous variable.

```
hist(imdb$imdb_rating)
```

Does this plot look good to you? Is it informative? Describe a way you could make it better.

How much detail do we want?

```
hist(imdb$imdb_rating, breaks = 3)
```

Vary the number of bins in the histogram – what changes? Which setting would you choose to use and why?

Improving the plot

All plots should have:

- A descriptive title and labeled axes, with clear units.
- Clear representation of the data (e.g., appropriate bin width for histograms).
- Sourcing/attribution.

```
hist(imdb$imdb_rating)
```

Improve this histogram using the guidelines above. (If you don't know how to do it, use the ? button to tell AI how you want the plot to look, and paste the resulting code.) Run your code, then submit your plot. We will look at some of the plots together.

One discrete variable

How many movies came from each decade?

Visualizing the decades

- A bar plot visualizes the variation within a single discrete variable.
- Let's see how many movies came from each decade.
- We first need to turn a year such as 1997 into its decade, 1990s.

```
breaks <- seq(1910, 2030, by = 10)
labels <- paste0(seq(1910, 2020, by = 10), "s")
cut(
  c(1989, 1990, 1997, 2000),
  labels = labels,
  breaks = breaks,
  right = FALSE
)
```

Count the films in each decade

Use `cut()` inside `mutate()` to group Year into decades, then count the movies in each decade.

```
decade_counts <- imdb |>  
  mutate(decade = cut(_____, breaks = _____, labels = _____, right = FALSE)) |>  
  count(_____)
```

Bar chart of films by decade

```
barplot(decade_counts$n, names.arg = decade_counts$decade)
```

Improve this bar chart with better labels, titles, sourcing, etc. You can use the ? button to ask AI for help. Run your code, then submit your plot so we can compare designs.

Two continuous variables

IMDb publishes a score and vote counts for each movie. Does the average computed from those counts agree with IMDb's score?

Estimating the average rating

Using what we have learned so far, we want to be able to write:

```
1 imdb |> group_by(Title) |> summarize(mean_rating = mean(Rating))
```

Explain why this won't work. How do we need to transform our data to compute the mean rating correctly?

Plan the table

Here is the structure we want:

```
# A tibble: 4 × 3
  Title rating    votes
  <chr> <chr>    <dbl>
1 71    1 ratings    516
2 71    2 ratings    261
3 71    3 ratings    415
4 71    4 ratings    866
```

Describe in words how we can make the `imdb` table look like this – what is the name of the operation we should perform? If there are 7,072 movies and ten rating levels, how many rows should the new table have?

Gathering columns with `pivot_longer()`

To reshape data from wide to long, we have to specify:

- which columns to gather,
- where their names go,
- and where their values go.

```
ratings_long <- imdb |>
  pivot_longer(
    cols = `1 ratings`:`10 ratings`,
    names_to = _____,
    values_to = _____
  )
ratings_long |> head()
```

First try: average the rating column

We have a rating column now. Let's try averaging it:

```
ratings_long |> summarize(mean_rating = mean(rating))
```

What went wrong? Why does `mean(rating)` return NA instead of an average?

Text vs numeric

`pivot_longer()` copied the old **column names** into `rating`. Column names are text, so `rating` is a **character** column.

```
ratings_long |> select(Title, rating, votes) |> glimpse()
```

Look for `<chr>` beside `rating` and `<dbl>` beside `votes`.

Character: labels

"1 ratings"

"10 ratings"

"2 ratings"

Numeric: quantities

1 < 2 < 10

We can average these numbers.

What does a data type tell R?

The type determines how R stores values and which operations make sense.

- **Character:** text such as a movie title or "10 ratings".
- **Numeric:** quantities such as a vote count or mean rating.
- **Logical:** TRUE or FALSE, such as the result of `Year == 2023`.

Compare a number with its text representation:

```
10 + 2  
"10" + 2
```

Turn rating labels into numbers

- `parse_number(txt)` tries to extract a number from a piece of text.
- `as.numeric()` handles plain numeric text such as "10", but not a label such as "10 ratings".

```
ratings_numeric <- ratings_long |>  
  mutate(rating = parse_number(rating))  
ratings_numeric |> select(Title, rating, votes) |> head()
```

Which average do we want?

Suppose three people gave 2 stars and one person gave 10 stars.

```
small_votes <- tibble(rating = c(2, 10), votes = c(3, 1))  
small_votes
```

What would `small_votes |> summarize(mean(rating))` produce? What is the actual average across the four people? Explain why the answers differ.

Computing a weighted average

`weighted.mean(rating, votes)` multiplies each rating by its vote count, adds the products, and divides by total votes.

```
movie_means <- ratings_numeric |>
  group_by(`IMDb code`, Title, Year, imdb_rating) |>
  summarize(
    mean_rating = weighted.mean(rating, votes),
    .groups = "drop"
  )
movie_means |> head()
```

We now have one row per movie and both scores needed for the scatterplot.

How closely do the two scores agree?

A *scatterplot* visualizes the covariation between two continuous variables. Each point represents a movie with both a raw average and an IMDb score.

```
plot(movie_means$mean_rating, movie_means$imdb_rating)
```

Improve this scatterplot: add a descriptive title, labeled axes, and a source. Make overlapping points easier to see. You can use the ? button to ask AI for help. Run your code, then submit your plot so we can compare designs.

Show where the scores agree

The line $y = x$ marks movies with the same score under both calculations.
`alpha` makes points transparent, helping us see where movies overlap.

```
movie_means |>
  ggplot(aes(x = mean_rating, y = imdb_rating)) +
  geom_abline(slope = 1, intercept = 0, color = "gray50") +
  geom_point(alpha = 0.2) +

  labs(x = "Raw average rating (1-10)", y = "IMDb rating (1-10)")
```

Read the scatterplot

What does a point above or below the 45-degree line tell us about its two scores? Why might such discrepancies occur?

Which movies stand out?

- An **outlier** is an observation that differs substantially from the pattern in the rest of the data.
- Here, most movies lie near the agreement line.
- Movies far from that line have unusually large differences between their scores.

Find the biggest discrepancy

```
movie_means |> head()
```

- Start with all movies in `movie_means` and calculate gap as `mean_rating` minus `imdb_rating`.
- Order movies by the absolute value of gap, largest first.
- Keep the movie with the largest absolute gap.

The largest discrepancy: Martin (2024)



Martin (2024)

The mean computed from votes is **8.42**; IMDb's score is **2.2**.

One discrete and one continuous variable

Do older movies have higher ratings?

Average IMDb rating by decade

```
imdb_decade |> head()
```

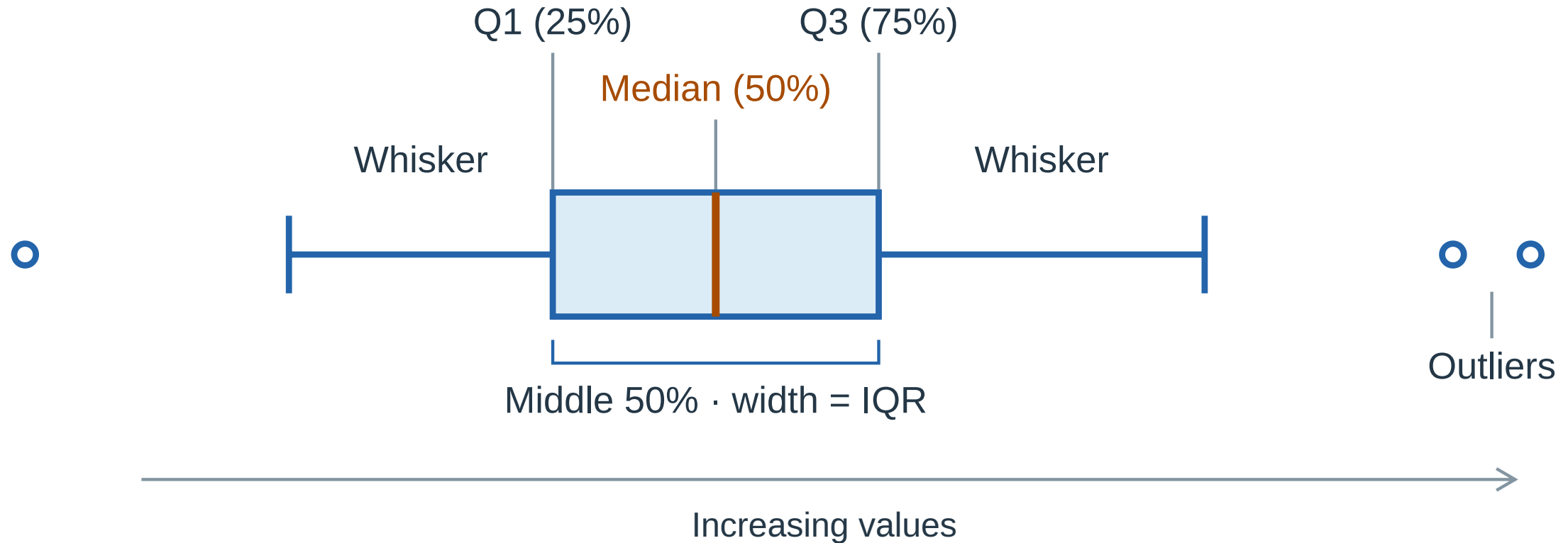
- Group the movies by decade.
- Summarize into one row per decade, including the number of movies as `n_movies`.
- Calculate `avg_rating` as the mean of `imdb_rating` within each decade, giving every movie

What does the average leave out?

- Movies from the same decade can have very different ratings.
- Two decades could have the same average but different spreads.
- We want to compare the distributions of ratings within decades.

A **boxplot** summarizes a numerical variable within each category.

Interpreting a boxplot



Whiskers end at the most extreme observations within **1.5 IQR** of the box. Points beyond them are shown individually; they are not necessarily errors.

IMDb ratings by decade

`imdb_decade` contains one row per movie, with its decade already added.

`decade_counts` contains one row per decade and its movie count.

In `boxplot()`, `imdb_rating ~ decade` means “show the distribution of IMDb ratings separately for each decade.” `data` supplies the table.

```
boxplot(imdb_rating ~ decade, data = imdb_decade)
```

Improve this boxplot: add a descriptive title, labeled axes, readable decade labels, and a source. Keep the decades in chronological order. You can use the ? button to ask AI for help. Run your code, then submit your plot so we can compare designs.

Plotting data over time

The boxplots suggest that ratings differ across release decades. Does the pattern change gradually, or are there particular years that stand out?

We will calculate the mean IMDb rating for each year and plot it over time.

Build a table with one row per year

Build your pipeline and click **Run Code** to check your progress.

```
imdb |> head()
```

- Group all movies in `imdb` by release Year.
- Summarize into one row per year, with `avg_rating` equal to the mean of `imdb_rating`, giving each movie equal weight.
- Include `n_movies`, the number of

Plot the mean IMDb rating by release year

```
yearly_ratings |> ggplot(aes(x = Year, y = avg_rating)) +  
  
  geom_point(aes(size = n_movies), color = "#2364aa") + # add means  
  geom_line() + # add line connecting yearly means  
  
  labs(x = "Release year", y = "Mean IMDb rating across movies", size = "Movies")
```

Add confidence whiskers for the mean

A mean based on fewer movies is (usually) less precise. Its **standard error** is $\text{sd}(\text{imdb_rating}) / \sqrt{n()}$.

```
yearly_intervals <- imdb |>
  group_by(Year) |>
  summarize(avg_rating = mean(imdb_rating), n_movies = n(),
            se = sd(imdb_rating) / sqrt(n()), .groups = "drop") |>
  filter(n_movies >= 2) |>
  mutate(margin = qt(0.975, df = n_movies - 1) * se)

yearly_intervals |>
  ggplot(aes(x = Year, y = avg_rating)) +
  geom_point(color = "#2364aa") +
  labs(x = "Release year", y = "Mean IMDb rating (95% CI)") +

  geom_errorbar(aes(ymin = avg_rating - margin,
                    ymax = avg_rating + margin), width = 0)
```

Explaining the trend

This file includes only movies with at least 25,000 votes. Which older movies are likely to reach that threshold today? Which might be missing? How does this selection process affect the trend in average ratings over time?

Which plot should I try?

Variables	Starting plot	Our example
One numerical	Histogram	Movie-average ratings
One categorical	Bar chart	Release decades
Two numerical	Scatterplot	Raw average and IMDb score
Numerical and categorical	Boxplots	Ratings within decades
Numerical summary over time	Line chart	Mean IMDb rating by release year

Week 4, Lecture 7: Exploratory data analysis

Closed September 22, 2026 at 3:45 PM

18 class questions · 1129 anonymous responses · 0 student questions

The following slides preserve what students submitted during class. No names or login information are included.

Report your film's title, release year, total votes, and IMDb rating.

71 anonymous responses

Plain answer with film details <small>Directly gives a film title and release year without the other requested numeric metadata.</small>	31
Partial numeric metadata included <small>Provides a film title with some numeric information (e.g., ratings count) but omits one or more requested fields or leaves the response incomplete.</small>	14
Title with implausible future year <small>Provides a film title paired with a year that appears to be a future or unlikely release year, suggesting an error or speculation.</small>	8
Nonresponse or misspelling <small>Responses that do not provide the requested film metadata and appear incomplete or contain typos rather than an answer.</small>	7

Response themes for question 1

Code-style data-selection expression**3**

Uses a code/filter expression intended to extract title, year, total ratings, and IMDb rating from a dataset rather than supplying the values directly.

Title only, ambiguous punctuation**3**

Gives only a film title (possibly with trailing punctuation) without year or numeric metadata.

Vague placeholder response**3**

Noninformative reply that neither names a film nor supplies requested metadata.

Other responses**2**

Student responses

1. 10 Things I hate about you (1999)
2. Avatar, released in 2009
3. Baby Driver, 2017. The other two categories are not loading properly
4. Barbie 2023
5. Barbie tt1517268 2023
6. Barbie, 2023, 45 total ratings
7. Barbie, 2025
8. batman
9. Braveheart, 1995, I cant see the other 2

Student responses

1. cars, 2006 7.3
2. cars, 2006,
3. Dune, 1984
4. fast and furious, 2009
5. Gensis, 2021
6. Good Will Hunting 1997
7. I chose The Lego Batman Movie and the year is 2017, total votes is 2000
8. Inception
9. Moneyball, 2011

Student responses

1. Moneyball, 2011, 1 ratings of 2169 2ratings of 1044
2. Mortal Engines; 2018; can't yet see the rest
3. Oppenheimer, 2023, 10767
4. Shrek, 2001
5. Spiderman, 2002, 200
6. The film's title is Cars, release year is 2006, total ratings is some amount, and the imdb_rating is also some amount.
7. The Prestige, 2006, 8.5 imdb
8. The title is "The Arrival", it came out in 1996 and its rating is 4000 5 stars on IMDB
9. The title of the film is Bottoms, it was released in 2023 and has over 5,000 ratings

Student responses

1. Titanic, 1997,
2. Titanic, 1997.
3. whiplash, 2014
4. Zootopia is released in 2016

Student responses

1. [REC] 3: Genesis, released year 2012, total ratings
2. 2014, 18570
3. 7.8
4. Crazy rich asians was release in 2018 and has
5. Film Title is Cars, Year 2006, total votes 10251, imdb rating is dbl
6. I chose 10 things i hate about you. it couldn't gine
7. Interstellar tt0816692 2014 18570 7504 10036 16165
8. Interstellar, 2014, 10 ratings,
9. Interstellar, 2014, 100000

Student responses

1. Shrek 2001 10 ratings
2. The Fast and the Furious: Tokyo Drift, 2006, 326445, 6.1
3. The Martian tt3659388 2015 4350 2217 3315 6331 # i 8 more variables: `5 ratings` <dbl>, `6 ratings` <dbl>, `7 ratings` <dbl>, # `8 ratings` <dbl>, `9 ratings` <dbl>, `10 ratings` <dbl>, # `Total ratings` <dbl>, imdb_rating <dbl>
4. The Room, 2003, 5369 ratings
5. Title `IMDb code` Year `1 ratings` `2 ratings` `3 ratings` `4 ratings` <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> 1 [REC] 3: Ge... tt1649444 2012 2868 2435 3908 5788

Student responses

1. Cars,2026
2. Michael (2026)
3. obsession 2026
4. obsession 2026,
5. Obsession, 2026, 30,540 total votes, 4.3 IMDb rating
6. Odyssey (2026)
7. Spider-Man: Brand New Day, 2026
8. Spiderman: Brand New Day, 2026

Student responses

1. did not have neough reviews
2. Interstellar 2014
3. Jurassic Park was released in 1993, can't find the number of votes due to it just saying dbl
4. Looking for obsession but can tfind it
5. Minions, 2015, 30,217, 5
6. obsseive
7. The Batman, 2022, ?

Student responses

1. `filter(title = "White Chicks"), released in 2004`
2. `imdb |> # fold: [Filter by title] filter(Title == "28 years later:bone temple")`
3. `imdb |> filter(title == "Oppenheimer") |> select(title, year, `Total ratings`, `IMDb rating`)`

Student responses

1. Alice in Wonderland, forgrt the time but the Disney rdition with real characters
2. Baby driver has 9000 1 stars and
3. Interstellar,

Student responses

1. a
2. I couldn't find mine, but ill try another
3. something

Which five movies released in 2023 have the most votes? Report their titles and vote counts.

64 anonymous responses

Minimal non-informative reply <small>Very short, non-descriptive responses that do not convey a procedure or answer.</small>	12
Opaque single-character reply <small>Extremely brief, non-informative responses consisting of a single character or similarly opaque content.</small>	11
Bare list of movie title guesses <small>Provides an unordered list of likely 2023 movie titles without vote counts or methodology.</small>	9
R/dplyr data-filtering approach <small>Provides an R pipeline using dplyr-like verbs to filter for 2023, sort by total ratings descending, take the top five, and select title and ratings.</small>	9

Response themes for question 2

Incorrect numeric movie-votes list

7

Gives a list of movie titles paired with vote counts that appear inconsistent or unlikely for 2023 releases, suggesting incorrect data or formatting.

Malformed code/data-manipulation attempt

6

Attempts to express a filtering and sorting pipeline but with syntax errors or mixed function names; intent is to filter for 2023 and take top five by votes.

Commented R pipeline with minor errors

5

An R/dplyr-style pipeline written with inline comments and slight logical mistakes (e.g., using `>=` for year or misusing count), aiming to filter for 2023, sort by votes, and select top five.

Response themes for question 2

Irrelevant single-word or junk reply

5

One-word or short nonspecific responses that do not relate to the requested data or procedure.

Student responses

1. .
2. a
3. b
4. I cound' tfind it
5. I did not quite get there, but
6. I wasnt able to find it
7. idk
8. n
9. Not enough time

Student responses

1. Not sure
2. something
3. x

Student responses

1. 12th fail
2. 12th Fail, 20 Days in Marioupol 65, A good Person, and a haunting in Venice
3. 12th Fail, 20 Days in..., 65, a good person, and a haunting... are the most rated movies, each with over 25,000 vote counts.
4. A huanting in Venice, 65, 12th fail
5. Air 191293
6. Paradise (25011), Gadar 2 (25235), The Vaccine War (25360), Taylor Swift: The Eras Tour (25471), Hi Nanna (25480)
7. Q
8. spiderman
9. the movies in 2023

Student responses

1. unable to arrange by Total ratings due to there being a space in the name
2. waht is a vote

Student responses

1. Barbie
2. im getting an error here: arrange(desc(Total_ratings))
3. Oppenheimer Barbie Spider-Man John wick and guardians of the gakaxy
4. oppenheimer barbie Spiderman Guardians John Wick
5. Oppenheimer, BARbie, Spider man across the spider verse, guardians od the galaxy 3, john wick 4
6. Oppenheimer, Barbie, Spider-Man: Across the Spider-Verse, Guardians of the Galaxy Vol 3, John Wick
7. Oppenheimer, barbie, spiderman, guardians, john wick
8. Oppenheimer, Spider-man, Barbie, Gotg vol 3, Spider-man.
9. Spder-Man and a few other movies

Student responses

1. 12th fail
2. 12th fall
3. barbie, oppenheimer
4. i'm not exactly sure
5. `imdb |> # fold: [Keep movies released in 2023] filter(year = 2023) |> # endfold # fold: [Sort from most votes to fewest] arrange(desc(votes) |> # endfold # fold: [Keep the first five movies and the requested columns] slice_head(n = 5) |> select(Title, `Total ratings`)`
6. `imdb |> filter(Year == 2023) |> arrange(desc(`Total ratings`)) |> slice_head(n = 5) |> select(Title, `Total ratings`)`
7. Oppenheimer, 990010; Barbie, 654761; Spider-Man: Across the Spider-Verse, 502557; Guardians of the Galaxy Vol. 3, 459179; John Wick: Chapter 4, 402472
8. spiderman
9. `unable to: # fold: [Keep movies released in 2023] filter(Year == 2023) |> # endfold # fold: [Sort from most votes to fewest] arrange(desc(Total_ratings)) |> # endfold # fold: [Keep the first five movies and the requested columns] slice_head(n = 5) |> select(Title, `Total ratings`) # endfold`

Student responses

1. 1 12th Fail 165920 2 20 Days in Mariupol 42811 3 65 104506 4 A Good Person 28382 5 A Haunting in Venice 138786
2. 12th Fail (165920) 20 days in Mariupol (42811) 65 (104506) A good person (28382) A haunting in Venice (138786)
3. 12th fail (165920), 20 Days in Mariupol (42811). 65 (104506).A good person (28382), haunting in Venice(138786)
4. 12th Fail 160000
5. title `Total ratings` <chr> <dbl> 1 Adipurush 136863 2 Winnie-the-Pooh: Blood and Honey 36335 3 Kisi Ka Bhai Kisi Ki Jaan 41900 4 The Archies 53481 5 Peter Pan & Wendy 30169
6. Oppenheimer 990010, Barbie 654761, Spider-Man across the Spider-Verse 502557, Guardians of the Galaxy Vol 3 459179, John Wick Chapter Four 402472
7. The five most-voted movies released in 2023 were Oppenheimer (990,010 votes), Barbie (654,761), Spider-Man: Across the Spider-Verse (502,557), Guardians of the Galaxy Vol. 3 (459,179), and John Wick: Chapter 4 (402,472).

Student responses

1. 1 Oppenheimer 990010 2 Barbie 654761 3 Spider-Man: Across the Spider-Verse 502557 4 Guardians of the Galaxy Vol. 3 459179 5 John Wick: Chapter 4 402472
2. `filter(year = 2023) arrange(desc(votes)) splice.head(n=5)`
3. `imdb |> # fold: [Keep movies released in 2023] filter("Year" == "2023") |> # endfold # fold: [Sort from most votes to fewest] arrange(Total ratings) |> # endfold # fold: [Keep the first five movies and the requested columns] slice_head(n = 5) |> select(Title, `Total ratings`)`
4. is this right? 12th Fail 165920 2 20 Days in Mariupol 42811 3 65 104506 4 A Good Person 28382 5 A Haunting in Venice 138786
5. Oppenheimer, Barbie, Spiderman, Guardians of the galaxy 3, John Wick ch4; 990010, 654761, 502557, 459179, 402472
6. Paradise Gadsden 2 vaccine war Taylor swift and hi nanna

Student responses

1. 12th fail, 20 days in Mariupol, 65, a good person, a haunting in Venice but I think it's not properly sorting by votes
2. 12th fail, 20 days, 65, adipurush, air
3. 12th Fail: 165920, 20 Days in Mariupol:
4. `imdb |> # fold: [Keep movies released in 2023] filter(year == 2023) |> # endfold # fold: [Sort from most votes to fewest] arrange(desc(votes)) |> # endfold # fold: [Keep the first five movies and the requested columns] slice_head(n = 5) |> select(Title, `Total ratings`)`
5. `imdb |> # fold: [Keep movies released in 2023] filter(year >= 2023) |> # endfold # fold: [Sort from most votes to fewest] arrange(count(votes)) |> # endfold # fold: [Keep the first five movies and the requested columns] slice_head(n = 5) |> select(Title, `Total ratings`)`

Student responses

1. 5 movies released in 2023 total votes
2. 5 movies released in are
3. Gensis
4. Paradise (25011), Gadar 2 (25235), The Vaccine War (25360), Taylor Swift: The Eras Tour (25471), Hi Nanna (25480)
5. revenges

- Keep only movies released in 2023. - Calculate each movie's percentage of one-star ratings. - Order movies from highest to lowest one-star percentage. - Print info for the five highest one-star percentage movies.

88 anonymous responses

Other responses

88

Student responses

1. `colnames(imdb) |> filter(Year == 2023) |> mutate(one_star_pct = `1 ratings` / `Total ratings` * 100) |> arrange(desc(one_star_pct)) |> slice_head(n = 5) |> select(Title, `Total ratings`, one_star_pct)`
2. `glimpse(imdb) |> filter(Year == 2023) |> mutate(one_star_pct = `1 ratings` / `Total ratings` * 100) |> arrange(desc(one_star_pct)) |> select(Title, Year, `1 ratings`, `Total ratings`, one_star_pct) |> slice_head(n = 5)`
3. `highest_percentage <- imdb |> filter(Year == 2023) |> mutate(one_star_per = `1 ratings` / `Total ratings`) |> arrange(desc(one_star_per)) |> slice_head(n = 5) |> select(Title, one_star_per) highest_percentage`
4. `highest_percentage <- imdb |> filter(Year == 2023) |> mutate(one_star_pct = `1 ratings` / `Total ratings`) |> arrange(desc(one_star_pct)) |> slice_head(n = 5) highest_percentage`
5. `imdb |> filter(Year == 2023) |> mutate(one_star_pct = `1 ratings` / `Total ratings`) |> arrange(desc(one_star_pct)) |> slice_head(n = 5) |> select(Title, one_star_pct)`

Student responses

1. `imdb |> # fold: [Keep movies released in 2023] filter(Year == "2023") |> # endfold # fold: [Sort from most votes to fewest] mutate(one_star_pct = '1 ratings' / 'Total ratings') |> arrange(desc(one_star_pct)) |> # endfold # fold: [Keep the first five movies and the requested columns] slice_head(n = 5) |> select(Title, one_star_pct) # endfold`
2. `imdb |> # fold: [Keep movies released in 2023] filter(Year == 2023) |> mutate(one_star_pct = `1 ratings` / `Total ratings`) arrange(desc("one_star_pct")) |> slice_head(n = 5) |> select(Title, one_star_pct)`
3. `imdb |> # Step 1: Filter for 2023 releases with at least one vote filter(year == 2023, total_votes > 0) |> # Step 2: Calculate the one-star percentage mutate(one_star_pct = (one_star_votes / total_votes) * 100) |> # Step 3: Sort by percentage descending arrange(desc(one_star_pct)) |> # Step 4: Take the top 5 highest slice_head(n = 5)`

Student responses

1. `imdb |> filter(Year == 2023) |> mutate(one_star_pct = '1 ratings' / 'Total ratings') |> arrange(desc('one_star_pct')) |> slice_head(n = 5) |> select(Title, one_star_pct)`
2. `imdb |> filter(Year == 2023) |> mutate(one_star_pct = '1 ratings' / 'Total ratings') |> arrange(desc(one_star_pct)) |> slice_head(n = 5)`
3. `imdb |> filter(Year == 2023) |> mutate(one_star_pct = '1 ratings' / 'Total ratings') |> arrange(desc(one_star_pct)) |> slice_head(n = 5) |> select(Title, one_star_pct)`
4. `imdb |> filter(Year == 2023) |> mutate(one_star_pct = `1 ratings` / `Total ratings` * 100) |> arrange(desc(one_star_pct)) |> slice_head(n = 5) |> select(Title, one_star_pct)`
5. `imdb |> filter(Year == 2023) |> mutate(one_star_pct = `1 ratings` / `Total ratings`) |> arrange(desc(one_star_pct)) |> slice_head(n = 5)`
6. `imdb |> filter(Year == 2023) |> mutate(one_star_pct = 100 * `1 star` / `Total ratings`) |> arrange(desc(one_star_pct)) |> slice_head(n = 5) |> select(Title, one_star_pct)`

Student responses

1. `imdb |> filter(Year == 2023) |> mutate(one_star_pct = 100 * ("1 ratings" / 'Total ratings')) |> arrange(desc(one_star_pct)) |> slice_head(n = 5) |> select(Title, one_star_pct)`
2. `imdb |> filter(Year == 2023) |> mutate(pct_one_star = `1 ratings` / votes * 100) |> arrange(desc(pct_one_star)) |> slice_head(n = 5) |> select(Title, pct_one_star)`
3. `imdb |> filter(Year == 2023) |> mutate(Percent_1_star = '1 ratings' / 'total ratings') |> arrange(desc(Percent_1_star)) |> slice_head(n = 5) |> select(Title, Percent_1_star)`
4. `imdb |> filter(Year == 2023) |> mutate(one_star_pct = '1 ratings' / 'Total ratings') |> arrange(desc(one_star_pct)) |> slice_head( n = 5 ) |> select(Title, one_star_pct)`
5. `imdb |> filter(Year == "2023") |> mutate(on_s = 100 * `1 ratings` / `Total ratings`) |> arrange(desc(on_s)) |> slice_head(n= 5)|> select(Title, one_s)`
6. `imdb |> filter(Year == "2023") |> mutate(one_star_pct = `1 ratings` / `Total ratings`) |> arrange(desc(one_star_pct)) |> slice_head(n = 5) |> select(Title, `one_star_pct`)`

Student responses

1. `imdb |> filter(Year == 2023) |> mutate("one_star_prop" = `1 ratings` / `Total ratings`) |> select(Title, one_star_prop) |> arrange(desc(one_star_prop)) |> slice_head(n = 5)`
2. `imdb |> filter(Year == 2023) |> mutate(one_pct = "1 ratings" / "Total ratings")`
3. `imdb |> filter(Year == 2023) |> mutate(percentage = `1 ratings` / `Total ratings` * 100) |> arrange(desc(percentage)) |> head(5) |> select(Title, percentage)`
4. `imdb |> filter(Year == 2023) |> mutate(one_star_prop = `1 ratings` / `Total ratings`) |> arrange(desc(one_star_prop)) |> slice_head(n=5) |> select(Title, one_star_prop)`
5. `imdb |> filter(Year == 2023) |> mutate( one_star_prop = `1 ratings` / `Total ratings` ) |> arrange(desc(one_star_prop)) |> select(Title, one_star_prop, imdb_rating) |> head(5)`
6. `imdb |> mutate(bad_reviews = `1 ratings` / `Total ratings` * 100) |> arrange(desc(bad_reviews)) |> select(Title, Year, bad_reviews) |> head(5)`
7. `imdb |> filter(Year == 2023) |> mutate(one_star_pct = "1 ratings" / "Total ratings") |> arrange(desc(one_star_pct)) |> slice_head(n = 5)`

Student responses

1. `imdb |> head(filter(Year == 2023) |> mutate(one_star_pct = `1` / `Total ratings` * 100) |> arrange(desc(one_star_pct))) |> slice_head(n = 5))`
2. `imdb |> filter(Year == 2023) |> mutate(one_star_pct = '1 ratings' / 'Total ratings') |> arrange(desc(one_star_pct)) |> slice_head(n = 5)`
3. `imdb |> filter (Years == 2023) |> mutate(one_star_prop = `1 ratings` / `Total ratings`) |> arrange(desc(one_star_prop)) |> slice_head(n = 5) |> select(Title, on_star_prop)`
4. `imdb |> filter(Year == '2023') |> mutate(one_star_pct = '1 ratings' / 'Total ratings') |> arrange(desc('one_star_pct')) |> slice_head(n = 5) |>`
5. `imdb |> filter(Year == 2023) |> mutate (one_star_pct = `1 ratings` / `Total ratings`) |> arrange(desc(one_star_pct)) |> select(Title, one_star_pct) |> print()`
6. `imdb |> filter(Year == 2023) |> mutate( onepercentage = `1 ratings` / `Total ratings`) |> arrange(desc(onepercentage))|> print(n = 5)`
7. `imdb |> filter(Year == 2023) |> mutate(one_star = '1 ratings' / "Total ratings") |> arrange(desc(one_star)) |> slice_head(n = 5)`

Student responses

1. `imdb |> filter(Year == 2023) |> mutate(one_star_pct = '1 ratings' / 'Total ratings') arrange(desc(one_star_pct)) |> slice_head(n=5) |> select(Title, one_star_pct)`
2. `imdb |> filter(Year == 2023) |> mutate(one_star_pct = '1 ratings' / 'Total ratings') |> arrange(desc(one_star_pct)) |> slice_head(n = 5) |> select(Title, one_star_pct)`
3. `imdb |> filter(Year == 2023) |> mutate(one_star_pct = `1 ratings` / `Total ratings` ) arrange(desc(one_star_pct)) |> slice_head(5)`
4. `imdb |> filter(Year == 2023) |> mutate(one_star_pct = `1 ratings` / `Total ratings` * 100) |> arrange(desc(one_star_pct)) |> select(Title, Year, one_star_pct) head(5)`
5. `imdb |> filter(Year == 2023) |> mutate(one_star_pct = `1 ratings` / `Total ratings`) |> arrange(desc(one_star_pct)) |> slice_head(n = 5) |> select(Title, one_star_pct)`
6. `imdb |> filter(Year == 2023) |> mutate(one_star_pct = `1 ratings` / `Total ratings`) |> arrange(desc(one_star_pct)) |> slice_head(n=5)`
7. `imdb |> filter(Year == 2023) |> mutate(one_star_pct = `1 ratings` / `Total ratings`) |> arrange(desc(one_star_pct)) |> select(`Title`, one_star_pct) |> head(5)`

Student responses

1. `imdb |> filter(Year == 2023) |> mutate(one_star_percent = `1 ratings` / `Total ratings`) |> arrange(desc(one_star_percent)) |> slice_head(n = 5) |> select(Title, one_star_percent)`
2. `imdb |> filter(Year == 2023) |> mutate(one_star_percentage = `1 ratings` / `Total ratings`) |> arrange(desc(one_star_percentage)) |> select(Title, one_star_percentage) |> head(n = 5)`
3. `imdb |> filter(Year == 2023) |> mutate(one_stars = ('1 ratings' / total_votes) * 100) |> arrange(desc(one_stars)) |> head(5)`
4. `imdb |> filter(Year == 2023) |> select(Year, Title) |> mutate(one_star_percentage = `1 ratings` / `Total ratings`) |> arrange(desc(one_star_percentage)) |> slice_head(n=5)`
5. `imdb |> filter(Year == 2023) |> mutate(one_star_pct = `1 ratings` / `Total ratings`) |> arrange(desc(one_star_pct)) |> slice_head(n = 5) |> select(Title, one_star_pct)`
6. `imdb |> filter(Year == 2023)|> mutate(oneproportion = '1 ratings' / 'Total ratings') |> arrange(desc(oneproportion)) |> head(n =5)`

Student responses

1. `imdb |> filter(`Year` == 2023) mutate(pct_1_star = `1 ratings` / `Total ratings` * 100) |> arrange(desc(pct_1_star)) |> slice_head(n = 5)`
2. `imdb |> filter(Year == "2023") |> mutate(pct_one_star = `1 ratings` / `Total ratings`) |> arrange(desc(pct_one_star)) |> slice_head(n = 5) |> select(Title,pct_one_star)`
3. `imdb |> filter(Year == "2023") |> group_by(Title) |> summarize(onepercent = ('1 ratings') / 'Total ratings') |> arrange(desc(onepercent)) |> slice_head(n = 5) |> print()`
4. `imdb |> filter(Year == 2023) mutate(one_star_pct = `1 ratings` / `Total ratings`) arrange(desc(one_star_pct)) |> slice_head(n = 5) |> select(Title, one_star_pct)`
5. `imdb |> filter(Year == 2023) imdb |> mutate(onestar = '1 ratings' / 'Total ratings') imdb |> arrange(onestar) |> slice_head(n=5) |> select(Title,onestar)`
6. `imdb |> filter(Year == 2023) mutate(percent_1star = '1 ratings' / 'Total ratings') |> arrange(desc(percent_1star)) |> slice_head(n = 5) |> select(Title, percent_1star)`

Student responses

1. `imdb |> filter(Year == 2023) |> mutate(one_star_pct = '1 ratings' / 'Total ratings') |> arrange(desc(one_star_pct)) |> slice_head(n = 5) |> select(Title, one_star_pct)`
2. `imdb |> filter(Year == 2023) |> mutate(one_star_pct = '1 ratings' / 'Total ratings') arrange(desc(one_star_pct)) |> slice_head(n = 5) |> select(Title, one_star_pct)`
3. `imdb |> filter(Year == 2023) |> mutate(one_star_pct = '1 ratings' / 'Total ratings') |> arrange(desc(one_star_pct)) |> slice_head(n = 5)`
4. `imdb |> filter(Year == 2023) |> mutate(one_star_pct = `1 ratings` / `Total ratings`) arrange(desc(one_star_pct)) |> slice_head(n = 5) |> select(Title, one_star_pct)`
5. `imdb |> filter(Year == 2023) |> mutate(one_star_pct = `1 ratings` / `Total ratings`) |> arrange(desc(one_star_pct)) |> slice_head(n = 5) |> select(Title, one_star_pct)`
6. `imdb |> filter(Year == 2023) |> mutate(one_star_pct = 1 ratings / Total ratings) arrange(desc(one_star_pct)) |> slice_head(n = 5) |> select(Title, one_star_pct)`
7. `imdb |> filter(Year == 2023) |> mutate(one_star_perc = '1 ratings' / 'Total ratings')`

Student responses

1. `imdb |> filter(Year == 2023) |> mutate(one_stars = "1 ratings" / "Total ratings") |> arrange(desc(one_stars)) |> slice_head(n=5)`
2. `imdb |> filter(Year == 2023) |> mutate(one_star_pct = `1 ratings` / `Total ratings`) |> arrange(desc(one_star_pct)) |> slice_head(n=5) |> select(Title, one_star_pct)`
3. `imdb |> filter(Year == 2023) |> mutate(one_star_pct = '1 ratings' / 'Total ratings')) |> arrange(desc(one_star_pct)) |> slice_head(n=5) |> select(Title, one_star_pct)`
4. `imdb |> filter(Year == 2023) |> mutate(one_star_pct = '1 ratings' / 'Total ratings')`
5. `imdb |> filter(Year == 2023) |> mutate(one_star_pct = '1 ratings' / 'Total ratings' arrange(desc(one_star_pct))`
6. `imdb |> filter(Year == 2023) |> mutate(one_star_pct = '1 ratings' / 'Total ratings')`
7. `imdb |> filter(Year == 2023) |> mutate(one_star_pct = '1 ratings' / 'Total ratings') |> arrange(desc(one_star_pct)) |> slice_head(n=5)`
8. `imdb |> filter(Year == 2023) |> mutate(one_star_pct = `1 ratings` / `Total rating`) |> arrange`

Student responses

1. `imdb |> filter(Year == 2023) |> mutate(one_star_pct = `1 ratings` / `Total ratings`) |> arrange(desc(one_star_pct)) |> slice_head(n=5) |> select(Title, one_star_pct)`
2. `imdb |> filter(Year == 2023) |> mutate(one_star_pct = '1 ratings' / 'Total rating' |> arrange(desc('one_star_pct')) |> slice_head(n = 5)`
3. `imdb |> filter(Year == 2023) |> mutate(onepercent = `1 ratings` / `Total ratings`) |> arrange(desc(`onepercent`)) |> head(5)`
4. `imdb |> filter(Year == 2023) |> mutate (one_star_prop = "1 ratings" / "Total ratings")`
5. `imdb |> filter(Year == 2023) |> mutate(one_star_prop = '1 ratings' / "Total ratings") |> arrange(desc(one_star_prop)) |> slice_head(n = 5)`
6. `imdb |> filter(Year == 2023) |> mutate(onestarpct = `1 ratings` / `Total ratings`) |> arrange(desc(onestarpct)) |> slice_head(n = 5)`
7. `imdb |> filter(Year == 2023) |> mutate(percentOne = '1 ratings' / 'Total ratings') |> arrange(desc(percentOne)) |> head(n=5)`
8. `imdb |> filter(Year == 2023) |> mutate(one_perc = `1 ratings` / `Total ratings`) |> arrange(desc(one_perc)) |> slice_head(n=5) |> select(Title, one_perc)`

Student responses

1. `imdb |> filter(Year == 2023)|> mutate(one_star_pct = `1 ratings`/`Total ratings`)|> arrange(desc(one_star_pct))`
2. `imdb |> filter(Year == 2023)|>`
3. `imdb |> filter(Year==2023) |> mutate(percentage=`1 ratings`/`Total ratings`)|> arrange(desc(percentage)) |> slice_head(n = 5) |> select(Title, percentage)`
4. `imdb |> head()`
5. `imdb |> head()`
6. `imdb |> head() imdb |> filter(Year == 2023) |> mutate(one_star_pct = (`1 ratings` / `Total ratings`) * 100) |> arrange(desc(one_star_pct)) |> slice_head(n = 5) |> select(Title, one_star_pct)`
7. `imdb |> head() filter(Year == 2023) |> mutate(one_star_pct = '1 ratings' / 'Total ratings' * 100) arrange(desc('one_star_pct')) |> slice_head(n = 5)`
8. `imdb |> head() |> filter(Year == 2023) |> mutate(one_star_pct = '1 ratings' / 'Total ratings') |> arrange(desc(one_star_pct)) |> slice(head(5)) |> select(Title, one_star_pct)`
9. `imdb |> head() |> filter(Year == 2023) |> mutate(one_star_pct = `1 ratings`/`Total ratings`)|> arrange(desc(one_star_pct))|> slice_head(n=5)`

Student responses

1. `imdb |> head(n=5)`
2. `imdb_2023 <- imdb |> filter(Year == 2023, !is.na(`Total ratings`)) |> mutate( percentage = `1 ratings` / `Total ratings` ) |> arrange(desc(percentage)) |> select(Title, Year, percentage) |> slice_head(n = 5) imdb_2023`
3. `movies_2023 <- movies %>% filter(year == 2023) movies_pct_1star <- movies_2023 %>% mutate(pct_1star = 100 * votes_1 / votes) top_5_1star <- movies_pct_1star %>% arrange(desc(pct_1star)) %>% slice_head(n = 5) top_5_1star` If you want it in one pipeline: `movies %>% filter(year == 2023) %>% mutate(pct_1star = 100 * votes_1 / votes) %>% arrange(desc(pct_1star)) %>% slice_head(n = 5)`

Does this plot look good to you? Is it informative? Describe a way you could make it better.

86 anonymous responses

Simple positive impression

22

Student expresses general approval of the plot without further detail.

Want subgrouping or temporal breakdown

17

Student feels the overall histogram hides important structure and suggests grouping by genre, year, director, or time periods to be more informative.

Recommend clearer axes and title

12

Student suggests improving axis limits, labels, or the chart title to increase clarity.

Informative but visually unappealing

11

Student finds the plot conveys information yet criticizes its visual design or attractiveness.

Response themes for question 4

Title should be simpler or clearer**10**

Student specifically notes the plot title is confusing or too cryptic and recommends making it simpler or clearer.

Suggest expand x-axis to show full range**6**

Student recommends widening or rescaling the x-axis so the plot fills the row and displays the complete range of IMDb ratings.

Add mean/median or summary stats**4**

Student suggests including statistical summaries such as the median and mean on the plot to convey central tendency.

Adjust bins or show distribution details**4**

Student recommends changing histogram bin widths, scales, normalization, or bar widths to reveal distribution features or lower-rating detail.

Student responses

1. I feel like its informative as a baseline as you are initially diving into the data set. It doesn't give a ton of information outside of the fact that the average rating is around 7 and not many people give out super low scores.
2. i like this plot
3. I think it is good simple graph. To make it better, we could make the histogram units into smaller bars. like labeling by .5
4. I think this plot looks good because it shows the distribution of ratings, but I also don't know what movie this is referring to, so I'm not sure how useful it is.
5. it is good and infromative, it shows the distribution pretty well
6. It is informative
7. It looks alright, I'm just not quite sure of its usefulness. It tells us that the average rating for a movie is about 6.5 when it ideally should be 5. We could make it better by perhaps considering other variables to center the movie ratings around a more centered point of measurement.
8. It looks good and somewhat descriptive, but I would probably make the bin smaller to get a more accurate visual.
9. it looks good to me, but if the title and the x name can be edited it will look better.

Student responses

1. It looks good, maybe more ticks on the x-axis
2. it looks good. i think the x axis should be labeled more where there is data, and also have a more descriptive label than the code for what we are looking at
3. looks good, not super infomrtiave
4. Looks pretty good to me , the data are distributed as a mountain. Maybe give the lower data a little more space.
5. sounds good to me, how to improve
6. This graph looks fair to me, as its left skewed a bit, means the most users gave fairly high rates.
7. This plot is pretty informative since it's unimodal and we can see a clear center of IMDB ratings here.
8. This plot looks good to me. However, it loses some information. It could be better if we can specify the width of each bar so that we can tell what scores are contained in each bar.
9. To me it doesn't look too bad. I mean you got the rating on bottom and frequency on y. I think you could potentially have better labels like x is rating and y is number of votes. You could maybe also add color, or change the title. I would say maybe make a note that this is skewed right

Student responses

1. yes
2. Yes I do like the plot it gives a good idea of what ratings are more common than others
3. yes, informative
4. yes, pretty good, the bottom axis is formatted weird. Make it 1-10

Student responses

1. I just dont think this plot has many useful information outside the fact that the mean imbd score is around 6.5. i think maybe a more genre specific plot could be intrsting
2. I think the plot is informative. People tend to give a 6-8 rating to movies. I wonder if it could be more informative by breaking it down by movie genre.
3. It doesn't split up by time frame so it doesn't answer our question. We would need some sort of time frame and to see how the frequencies change.
4. It doesnt look the best because its hard to tell what rating is the best evn thoughwe know the rating os in between 6 and 8.
5. It is a left-skew curve. We may need more data to find out the information we need
6. It is fine if all you want to see is which imdb ratings are most common, but this type of graph tells us nothing about the movies themselves - release date, genre, etc - which could tell us more about why they were rated that way.
7. It looks good and could be informative but it would be better if you made different plots for different genres or with some other filter.

Student responses

1. It's informative but it doesn't show how ratings have changed over time
2. it's not good to me because i can only get total rating distribution for all movies in this plot, and really not much information for specific movies
3. No. This histogram covers every movie in the dataframe, which is not that informative
4. The bell shape is informative, but the data is left-skewed.
5. The plot is informative in a way that it shows the frequency of imdb ratings, so if someone just wanted to know the spread of them, this would be useful. However, it doesn't show the spread across years or anything else like that.
6. This plot is informative, but it could be better if you further sorted by genre, director, or year
7. This plot shows the imdb rating by frequency across all films, but is not very informative about how it has changed over time. we could instead do a barchart separate by year to see how it changes over the years
8. While this plot shows the variation of ratings, if we want to answer the question of whether current ratings are lower than past ratings, we should group the data by periods in the histogram

Student responses

1. Yes it is not very skewed and looks very normally distributed
2. yes, shows the current maxima and bell curve trend, also indicates its skewed

Student responses

1. I like how the data looks on the plot but I would change the axis titles to make it look better and be more descriptive
2. I like the data in the plot, but it would be much more informative with better axis labels and a title that tells you exactly what the histogram is measuring
3. I think it looks fine. What could be helpful would be a title and axis label for x axis, and maybe the scale could have a few more numbers, or the bins for the histogram could be a bit smaller.
4. I would like more specific labels on the x axis
5. It doesn't look good and could be more informative. The first thing I would do is change the x-axis to imdb-ratings and the y-axis to number of movies or something similar.
6. It is a decent plot, but one way it could be improved would be if the axes were renamed and the title was more descriptive of the dataset we are pulling from.
7. It is informative, but we should change the titles to be more readable and we should change the bins to reflect the scores users can give. also, the x axis should be labeled 1 -10

Student responses

1. It is somewhat informative because we can see that most movies have IMDb ratings around 6–8, with fewer very low ratings. However, the labels are not very clear. I would improve it by using a more descriptive title and changing the x-axis label from `imdb$imdb_rating` to “IMDb Rating.”
2. Make the title and x-axis a little cleaner
3. plot dosent look bad, ,aybe adjust axis and chart title?
4. The plot is somewhat informative, but we could label the x-axis and title so that the plot is more informative
5. this is an informative plot, but the upper and lower limits should be displayed to be more clear

Student responses

1. I don't like the x axis label, I need it to be better worded. I also don't like how the x axis ticks are set up, I want to see ticks from maybe 0 to some value and that entire line to be longer.
2. It is informative, but can be better with more labeling.
3. It is informative, but doesn't look super good. Better titles would be good and maybe showing ratings all the way from 1-10 on the x axis
4. it seems informative but is not very visually pleasing
5. It shows the basic distribution and how the ratings are centered at around 7. Could be improved with some stats
6. It successfully shows an overall distribution of ratings but doesn't give any detailed information
7. no, the labels aren't informative and there is no units
8. The plot looks alright. It doesn't communicate the variance as intended, though; perhaps a better title, along with a better scale would help.
9. The x-axis could be more detailed instead of just the even numbers (which don't even go up to the max rating). Same with the y-axis. The title and axes titles aren't descriptive either.

Student responses

1. yeah that plot is pretty. maybe some colors could make it look prettier
2. Yes this plot looks good and I would say it is informative but its hard to see what the histogram is informing. So by changing the variables or th x and y axis would be better

Student responses

1. I think this plot looks good to me it tells me the distribution of imdb ratings for all films is around normal I think it could be better if it categorises or tells us more about the rating of individual films
2. It could be better by decreasing the bin length to show the data better
3. It could be better by decreasing the bin size to show a more general trend.
4. It gives an overview of the dataset, but we want more specific number or central tendency.
5. Its not a bad plot and it displays the data well, however just displaying the ratings of all the movies is not incredibly informative
6. The frequency could be turned into percentage each bin
7. The plot is ok, it lets you know how frequent the various numeric ratings are, it would be more informative if it was focused on a specific year, movie, or user
8. The title is too cryptic, I would make a more simple title
9. This plot is ok if you just wanted to see the distribution of all the ratings. However, since there is so much data it is hard to differentiate between the ratings and doesn't tell anything about how they differ over years

Student responses

1. This plot looks good to me as it shows the distribution of imdb ratings, but to make it better, we can add titles for each axis.

Student responses

1. Expand the x-axis to fill the entire row so we can see the full range of IMDb ratings.
2. I like the plot because it shows the frequency of each movie rating, but a way it can be better is if you added what movie the rating corresponded to. The labels and titles could be improved
3. it is skewed
4. The plot is informative but could be constructed in a better way. The first thing is to give actual labels to the x axis and the title of the graph. I would also alter the scale that is represented on the x-axis because it only goes to 8 but we have much more data that goes beyond 8. Bin size may need to be changed as well
5. This plot tells me only about what people vote overall for movies listed in imdb. It could be labeled better
6. We can make this plot better by defining the ranges that each bar covers. And we could also make the x axis wider so that it properly covers all of the data points

Student responses

1. could include median and mean
2. It could be divided into fewer groups, since you can only rate whole and half numbers.
3. It is not infomrative because we don't know which movie or even what genre, it is just a distribution of scores for all.
4. This plot is informative, but we can need to show what movies have those high/low ratings, a dot plot would be best here

Student responses

1. not rly, dont know what for
2. The plot is somewhat informational as it shows that the ratings are left skewed and clustyered between 6-8, however the labels are not that great/confusing
3. This plot shows the frequency of different imdb ratings. The histogram is not normalized, so the data is off-center
4. You could change the bin sizes to better capture the lower ratings

Vary the number of bins in the histogram -- what changes? Which setting would you choose to use and why?

0 anonymous responses

No responses were submitted.

Improve this histogram using the guidelines above. (If you don't know how to do it, use the ? button to tell AI how you want the plot to look, and paste the resulting code.) Run your code, then submit your plot. We will look at some of the plots together.

65 anonymous responses

Other responses

65

Student responses

- 1.
- 2.
- 3.
- 4.
- 5.
- 6.
- 7.
- 8.
- 9.

Student responses

- 1.
- 2.
- 3.
- 4.
- 5.
- 6.
- 7.
- 8.
- 9.

Student responses

- 1.
2. .
3. Added Title, changed breaks
4. Adding more labvels on the x access and in the tile, and add bins.
5. Changes the title and the label of the axis and picks the right number of bins
6. `ggplot(movies, aes(x = imdb_rating)) + geom_histogram(binwidth = 0.5, boundary = 0, color = "white", fill = "steelblue") + labs( title = "Distribution of IMDb ratings for 7,072 movies", x = "IMDb rating (0 to 10)", y = "Number of movies", caption = "Source: movies dataset" )`
7. `ggplot(movies, aes(x = rating)) + geom_histogram(binwidth = 0.5, color = "black", fill = "skyblue") + labs( title = "Histogram of Movies vs. their Ratings", x = "Rating", y = "Number of Movies" )`
8. `ggplot(your_data, aes(x = imdb_rating)) + geom_histogram(binwidth = 5, fill = "steelblue", color = "white") + labs( title = "Distribution of imdb_rating" + theme_minimal()`
9. `ggplot(your_data, aes(x = your_variable)) + geom_histogram(binwidth = 2, color = "black", fill = "skyblue") + labs( title = "Distribution of your_variable", x = "your_variable (units)", y = "Count" )`

Student responses

1. help
2. `hist( imdb$imdb_rating, breaks = 15, main = "Distribution of IMDb Ratings", xlab = "IMDb Rating", ylab = "Number of Movies", xlim = c(0, 10) )`
3. `hist( imdb$imdb_rating, breaks = 8, main = "Distribution of IMDb Movie Ratings", xlab = "IMDb Rating", ylab = "Number of Movies")`
4. `hist(imdb ratings)`
5. `hist(imdb$imdb_rating, breaks = 10, main = 'Movie Ratings', xlab = "IMDb rating (0 to 10)", ylab = "Number of movies", col = "lightblue", border = "white")`
6. `hist(imdb$imdb_rating, breaks = 10, main = "Distribution of IMDb ratings (Source: IMDb)", xlab = "IMDb rating (points)", ylab = "Number of films")`
7. `hist(imdb$imdb_rating, breaks = 100)`
8. `hist(imdb$imdb_rating, breaks = 50, main = "Distribution of IMDb ratings for movies in the dataset", xlab = "IMDb rating (0 to 10)", ylab = "Number of movies", col = "lightblue", border = "white") mtext("Source: IMDb dataset used in class", side = 1, line = 3, cex = 0.8)`
9. `hist(imdb$imdb_rating, breaks = seq(0, 10, by = .1), main = "Distribution of IMDb Ratings", xlab = "IMDb rating (0 to 10)", ylab = "Number of movies")`

Student responses

1. `hist(imdb$imdb_rating, breaks = seq(0, 10, by = 0.5), main = "Distribution of IMDb Ratings for Movies in the Dataset", xlab = "IMDb rating (0 to 10)", ylab = "Number of movies", col = "lightblue", border = "white") mtext("Source: IMDb movie dataset used in class", side = 1, line = 3, cex = 0.8)`
2. `hist(imdb$imdb_rating, breaks = seq(0, 10, by = 0.5), main = "Distribution of IMDb ratings for movies", xlab = "IMDb rating (0 to 10)", ylab = "Number of movies", col = "lightblue", border = "white") mtext("Source: IMDb movie dataset used in class", side = 1, line = 3, cex = 0.8)`
3. `hist(imdb$imdb_rating, breaks = seq(0, 10, by = 0.5), main = "Distribution of IMDb ratings for movies", xlab = "IMDb rating (0 to 10)", ylab = "Number of movies", col = "lightblue", border = "white") mtext("Source: IMDb movie dataset used in class", side = 1, line = 3, cex = 0.8)`
4. `hist(imdb$imdb_rating, main = 'IMDB Ratings by Frequency', xlab = 'IMDB Rating', bin = 50)`

Student responses

1. `hist(imdb$imdb_rating, main = "Distribution of IMDb movie ratings", xlab = "IMDb rating (0 to 10)", ylab = "Number of movies", breaks = 150, col = "lightblue", border = "white")`
2. `hist(imdb$imdb_rating, main = "Distribution of IMDB Ratings", xlab = "IMDB rating (0 to 10)", ylab = "Number of movies", breaks = 20, col = "lightblue", border = "white")`
3. `hist(imdb$imdb_rating, main = "Distribution of IMDb ratings", xlab = "IMDb rating (0-10)", ylab = "Number of movies", breaks = 20, col = "lightblue", border = "white")`
4. `hist(imdb$imdb_rating, main = "Distribution of IMDB Ratings", xlab = "IMDB rating", ylab = "Frequency", breaks = 20)`
5. `hist(imdb$imdb_rating, main = "Distribution of IMDb Ratings", xlab = "IMDb rating", ylab = "Number of movies", col = "skyblue", border = "white", breaks = 10)`
6. `hist(imdb$imdb_rating, main = "Distribution of IMDb Ratings", xlab = "IMDb rating", ylab = "Number of movies")`
7. `hist(imdb$imdb_rating, main = "Histogram of IMDB Ratings, From IMDB Database", xlab = "IMDB Rating", ylab = "Frequency", col = "green")` I added labels

Student responses

1. `hist(imdb$imdb_rating, main = "Histogram of IMDb ratings", xlab = "IMDb rating", ylab = "Number of movies", col = "skyblue", border = "white")`
2. `hist(imdb$imdb_rating, main = "Imdb Rating distribution")`
3. `hist(imdb$imdb_rating,main = "Distribution of IMDb ratings", xlab = "Avg Rating", ylab = "Number of Ratings", breaks = 10)`
4. `hist(imdb$imdb_rating)`
5. `hist(imdb$imdb_rating) main = "Histogram of Total IMDB Ratings" xlab = "IMBD Rating (stars)" ylab = "Frequency (ratings)"`
6. i add axis and labels
7. I add Descriptive title , Labeled axes + units , Appropriate bin width
8. I added x and y axis labels a title changed color of the plot
9. I changed nothign!

Student responses

1. I changed the axes.
2. I changed the color, and the title and the axes names
3. I think this could be improved with a source attribution
4. `library(ggplot2) ggplot(imdb, aes(x = imdb_rating)) + geom_histogram(binwidth = 0.5, fill = "steelblue", color = "white") + labs( title = "Distribution of IMDb Ratings", x = "IMDb rating (0 to 10)", y = "Number of films", caption = "Source: imdb dataset" )`
5. sorry man
6. still working on it
7. title = total ratings distribution, x lab = rating from 0 to 10
8. Title has been added and x-axis expanded
9. `x <- imdb$imdb_rating hist(x, breaks = seq(1, 10, by = 0.5), main = "Distribution of IMDb ratings", xlab = "IMDb rating (0 to 10)", ylab = "Number of movies", col = "lightblue", border = "white")`

Improve this bar chart with better labels, titles, sourcing, etc. You can use the ? button to ask AI for help. Run your code, then submit your plot so we can compare designs.

67 anonymous responses

Other responses

67

Student responses

- 1.
- 2.
- 3.
- 4.
- 5.
- 6.
- 7.
- 8.
- 9.

Student responses

- 1.
- 2.
- 3.
- 4.
- 5.
- 6.
- 7.
- 8.
- 9.

Student responses

- 1.
- 2.
- 3.
- 4.
- 5.
- 6.
- 7.
8. Add a title
9. add lablin and making each popt clearer

Student responses

1. Add title and labels
2. added all labels
3. added title, and axis labels
4. all the labels are there, but maybe more bars
5. `barplot( decade_counts$n, names.arg = decade_counts$decade, col = "steelblue", border = "white", main = "Number of Films by Decade", xlab = "Decade", ylab = "Number of films" ) sharp`
6. `barplot( decade_counts$n, names.arg = decade_counts$decade, col = "steelblue", border = "white", main = "Number of Films by Decade", xlab = "Decade", ylab = "Number of films", las = 2 )` If you want, you can also add a source note with `mtext()`: `mtext("Source: film dataset", side = 1, line = 4, cex = 0.8)` `las = 2` rotates the decade labels so they are easier to read.
7. `barplot( decade_counts$n, names.arg = decade_counts$decade, col = "steelblue", border = NA, main = "Number of films by decade", xlab = "Decade", ylab = "Number of films", las = 2 )`
8. `barplot( decade_counts$n, names.arg = decade_counts$decade, main = "Number of films by decade", xlab = "Decade", ylab = "Number of films", col = "skyblue", border = "white", las = 2 )`

Student responses

1. `barplot( decade_counts$n, names.arg = decade_counts$decade, main = "Number of films by decade", xlab = "Decade", ylab = "Number of films", col = "steelblue" )`
2. `barplot( decade_counts$n, names.arg = decade_counts$decade, main = "Number of films by decade", xlab = "Decade", ylab = "Number of films", col = "steelblue" )`
3. `barplot( decade_counts$n, names.arg = decade_counts$decade, main = "Number of films by decade", xlab = "Decade", ylab = "Number of films", col = "steelblue", border = "white", las = 2 )`
4. `barplot( decade_counts$n, names.arg = decade_counts$decade, main = "Number of films by decade", xlab = "Decade", ylab = "Number of films", col = "yellow", border = "white" ) mtext("Source: film dataset used in class", side = 1, line = 3, cex = 0.8)`
5. `barplot(decade_counts$n, names.arg = decade_counts$decade, main = 'Number of Movies by Decade', xlab = 'Decade', ylab = 'Number of Movies')`
6. `barplot(decade_counts$n, names.arg = decade_counts$decade, main = "Films by decade", xlab = "Year", ylab = "Number of movies released")`

Student responses

1. `barplot(decade_counts$n, names.arg = decade_counts$decade, main = "Number of films by decade", xlab = "Decade", ylab = "Number of films")`
2. `barplot(decade_counts$n, names.arg = decade_counts$decade, main = "Number of Movies Released per Decade, From IMDB Database", xlab = "Decade", ylab = "Number", col = "pink")`
3. `barplot(decade_counts$n, names.arg = decade_counts$decade, main = "Rating Distribution", xlab = "Decades", ylab = "Freq")`
4. `barplot(decade_counts$n, names.arg = decade_counts$decade)`
5. `barplot(decade_counts$n, names.arg = decade_counts$decade)`
6. `barplot(decade_counts$n, names.arg = decade_counts$decade)`
7. better
8. change labels
9. change to blue

Student responses

1. each decade to have one bar
2. getting htjere
3. I added a title, axis labels for both x and y to make everyrthing more clear
4. I changed the title and made the bins smaller.
5. I made this more understandable
6. I made x and y axis change names
7. i would add axes and title names main = "Number of Films by Decade", xlab = "Decade", ylab = "Number of films"
8. I would change decade title and number of films on y axis
9. `imdb |> mutate(decade = floor(Year / 10) * 10) |> count(decade) -> decade_counts barplot(decade_counts$n, names.arg = paste0(decade_counts$decade, "s"), main = "Number of Movies Released by Decade", xlab = "Release Decade", ylab = "Number of Movies", las = 2) mtext("Source: IMDb dataset", side = 1, line = 5, adj = 1, cex = 0.8)`

Student responses

1. include axes titles
2. label all horizontal bars
3. `par(mar = c(7, 4, 4, 2) + 0.1) barplot( decade_counts$n, names.arg = decade_counts$decade, las = 2, col = "steelblue", border = "white", main = "Number of films by decade", xlab = "Decade", ylab = "Number of films" )`
4. The title should be "Distribution of Films throughout Time". The x-axis should be labeled as "Decade". The y-axis should be labeled as "Frequency". Sourcing should be attributed to IMDb.
5. This plot needs a title `ggplot(films, aes(x = decade)) + geom_bar(fill = "steelblue") + labs( title = "Number of Films Released by Decade", x = "Decade", y = "Number of Films", caption = "Source: films dataset" ) + theme_minimal()`
6. To improve this plot we could add labels as to what the axes actually are
7. use ggplot
8. You need to add axis labels, a better tick system for the x axis, a title, source of data, make it a histogram might be better too, rather than a bar chart?

Explain why this won't work. How do we need to transform our data to compute the mean rating correctly?

82 anonymous responses

Data in wide format needs reshaping

38

Students note the dataset has one row per movie with multiple rating columns and must be converted to a long/tidy format (gather/pivot_longer) so all ratings are in a single column before averaging.

Multiple rating columns present

15

Students point out there are two separate rating measures/columns that must be handled before computing a mean.

Titles already aggregated/grouped

8

Student notes that titles are already grouped, suggesting aggregation may be preventing correct mean calculation.

Invalid or mismatched variable name

7

Students indicate the current variable name used for rating is incorrect or not found, implying a naming mismatch must be fixed.

Response themes for question 8

Other responses

5

Responses the model could not place reliably.

Missing or incomplete ratings across rows

4

Students observe that not all ratings are present for every item and that all individual ratings are needed to calculate the correct mean.

Need to create a column for mean

3

Students suggest computing a new column (e.g., via mutate) that contains the mean rating after transforming the data appropriately.

Other responses

2

Student responses

1. All of the data is in wide form and needs to be translated to long form.
2. Because the dataset is now wide form. need to reshape the data from wide form to long form.
3. Because we are not claeer about Title. We need to transform the data to numbers
4. current data has multiple rating columns
5. Data is already in a "wide" format, with one row per title. We want multiple rows per title, one row for each numerical rating.
6. Data needs reshaping
7. Each movie only has 1 row. it's in wide format and needs to be turned to long.
8. It won't work because each Title can appear many times, and the Rating values are often stored as text ranges or separate pieces rather than one clean numeric value per movie. Taking mean(Rating) directly would either fail or give the wrong result.
9. mutate a new column

Student responses

1. Need to transform the data from long form
2. no, currently the rating is shown as number of votes but we need it as a proportion or weighted average to correctly calculate mean rating
3. Put all the ratings into one column
4. Since the one movie contains one row of `imdb_rating` only, so the mean function makes no sense here.
5. The data has one row per movie, with vote counts in separate columns for each star rating, rather than individual votes in a Rating column.
6. The data is in the wide format meaning a column exists for each rating category so mean rating can't be computed
7. The data is not in the correct form, the data needs to be pivoted longer
8. the data is not in the right form. need to make the data long
9. The data isn't in the right form because we are calculating the mean frequency for each rating instead of the actual mean of the stars being rated.

Student responses

1. the ratings are in wide form so they are in 10 different columns and not under the name Rating. I think if we transform it to long in 1 column called rating, then we could do something like this
2. There is no column called rating. Our data is wide, meaning the ratings are split up between columns. We need our data to be long to summarize it like this.
3. This assumes that we have long data (a row for each movie + rating combo) but we have wide data, a column for each rating value
4. This code won't work because each movie has one row with separate columns for each rating (1–10), rather than individual ratings. We need to reshape the data into long format and use the number of votes for each rating to calculate a weighted average.
5. This is not going to work because we do not have the data in tidy data. Instead, we need to make the data longer by making it so that values of our data are no longer column names
6. This will not work because "Rating" consists of different frequencies of 1, 2, 3, etc. It needs to be counted then averaged for each film.

Student responses

1. This will not work because we do not have a column called rating, so we cannot take the mean of that, there are many rating columns present
2. This will not work, because we have the rows as each movie and this means we cannot group by title because title and ratings are both columns. We would basically need to invert the data, Have each row be a rating, and then group by title and average out the particular votes per rating
3. This won't work because each movie is one row
4. This won't work because of the wide format of the data. It needs to be reshaped. The ratings need to be converted to long format
5. This won't work because Rating is the number of ratings of that movie not the actual rating. We need to get the mean rating by summing all of each rating and dividing by the total counts of rating.
6. This won't work because all rows are one film with each rating counted
7. We need the data to be in long format.
8. we need to calculate rating
9. We need to transform the data from ratings by their groups of stars using wide or long function.

Student responses

1. We need to transform the data from wide to long
2. We need to transform the data to have a column for overall Rating, weighing the stars and making an overall total
3. We need to use pivot_long to get the each rating accounted for numerically, instead of categorically as a column by each title
4. You can't yet group by title as the data is in the wrong format for doing so.
5. you need all the ratings

Student responses

1. .
2. Because Rating isn't a column name yet, meaning there is nothing for the code to find the mean of. We would need to get a Rating column.
3. For the mean rating to be computed correctly, we need to transform the data so that each row represents a movie
4. imdb is formatted so each row is a different review, not that each of the rows is the entirety of one variable.
5. its isn the wrong form as rating is rn serperate by 1s atr to start up to five
6. Rating is not a variable, there are a collection of rating coumns (1 ratings, 2 ratings, etc.) that need to be accounted for before we compute mean from all those individual ratings
7. Rating is not stored in the correct way for this to work, you have to manually create a function to compute the rating to do it like this
8. The data is currently long, both title and ratings are column names. We need to rearrange the dataframe
9. The ratings are currently stored int the same data, we what to store them in different period

Student responses

1. there are two rating measures. Transform by either using mean between the two or picking one
2. There is no Rating column that has the a persons exact rating. You can't take the mean of this ratings column.
3. This won't work because the rating column is split into 5 different categories and the code doesn't know that could we create a mean rating using all 5 columns?
4. we currently have the number of ratings on each of the 1-5 starts so we don't have one whole ratings column where we can just find the mean
5. we need to transform the data. It is in long form right now, so each individual rating is its own column, so you can't just mean out the ratings. you need to use pivot_wide
6. won't work because its missing

Student responses

1. Grouping by title averages each movie's own rating column with itself. We would need to summarize(`mean_rating = mean(rating)`) on the whole dataset
2. I think we need to join the data somehow and then categorize it.
3. Multiple instances of a title?
4. The data has multiple rows per title for reasons other than separate ratings, so taking the function after grouping by Title can give a misleading result or just repeat the same rating information across duplicated rows.
5. The data isn't in the right form and we will need to transfer the data so that it is grouped correctly and we have to use pivot
6. the titles are already grouped, the rating # is how many raters not the score. we would need to use the rating # as weight for each score
7. title can appear multiple times with multiple ratings causing duplicating ratings skewing the mean
8. titles are not unique

Student responses

1. Maybe because there isn't a category called "Rating" in the initial imdb data sheet? It needs to be transformed first
2. Rating is not one of our variables, and it should be weighted
3. rating isn't a column option, we need to use total rating, data is wide format and needs reshaping
4. Rating isn't a valid name here. Each row is a bijective with the title, so we need to convert each column (1 ratings, 2 ratings, etc) and mean over those to get the mean rating.
5. the data is not in the right format right now, there is no Rating column as of now
6. There is no rating column,
7. there is no variable named "Rating" yet

Student responses

1. have to shift enter and do it one by one also check the spell
2. The data needs to be transformed so that we use the right mean and the table shows exactly what we need
3. The part with gettting the mean is wrong, you need to calculate it a different way, need to filter
4. There are many different movies for the same rating so taking the mean rating will give us a lot of duplicating values
5. Use the wieghts

Student responses

1. because the data is stored right now as the number of 1 ratings, 2 ratings, etc so this will not work
2. Each Title is different, so summarize actually wont do any thing.
3. There would be missing or incomplete values, the data is also not in the correct shape
4. We need to get the raw votes to use this method.

Student responses

1. This will compute one mean for all movies.
2. You need to use mutate in order to make a new column for the mean rating
3. You need to use the mutate command first

Describe in words how we can make the `imdb` table look like this -- what is the name of the operation we should perform? If there are 7,072 movies and ten rating levels, how many rows should the new table have?

0 anonymous responses

No responses were submitted.

What went wrong? Why does `mean(rating)` return `NA` instead of an average?

81 anonymous responses

Missing values present (NAs)

31

Responses identifying that NA/missing values in the data cause the mean to return NA.

Ratings stored as strings, not numeric

15

Responses pointing out that rating values are text (e.g., '1 rating') or contain titles, so they cannot be averaged.

Wrong column name or variable used

13

Responses saying the code is referencing the incorrect column or object (wrong name like rating vs ratings or imdb_rating) or the rating variable wasn't created.

Need to group before aggregating

9

Responses saying aggregation should be done within groups (by movie title) rather than overall.

Response themes for question 10

Mention of is.na misuse or not using is.na**5**

Responses noting that the is.na function should be used (or warning against not using it) to handle missing values.

Code worked for that student**3**

Responses stating the code ran successfully for the responder (no issue encountered).

Other: weighting or not weighted**3**

Responses mentioning weighting or that the mean didn't weight counts (distinct viewpoint about aggregation/weighting).

Other responses**2**

Student responses

1. Because we haven't actually weighted the ratings yet we just have the number of the different ratings that were given
2. For some reason i dont get the error that everyone else is getting but I assume the reason for the error is that there are some ratings where people dont rate?
3. if there is even one na within the data or if there is any non numeric values, the mean cannot be computed and it prints out na
4. It doesn't drop missing numbers, so NA becomes the average
5. It returns NA instead of the average because each column has different number of ratings
6. maybe some movies don't have every rating, so they have NA in that column, and so it returns NA
7. mean(rating) returns NA because if there is any missing value, the mean will be calculated as NA
8. mean(rating)returns NA because there exists an entry NA
9. missing data

Student responses

1. one of the values is missing
2. some of the ratings are na
3. The data type isn't numeric
4. The mean_rating turns out to be NA because we failed to filter out NA values in rating before.
5. The NAs were not removed.
6. The rating columns contains atleast one N/A value and by default mean adds this NA to the whole result rather than ignoring it
7. The rating is text, not numbers, so it cannot be used with mean().
8. There are missing values, leading to the value of NA.
9. there are na in the data

Student responses

1. There are NA ratings still in the data set. $1 + \text{NA} / 4 = \text{NA}$
2. There are some missing values, so the mean returns NA
3. There is a NA value where a movie is not rated
4. There is an issue with NAs somewhere in the column
5. there may be some missing data
6. There might be an NA value in 'rating'
7. There must have been a missing value for one of the data points, which does not let us calculate the mean, the ratings are in text form and need to be converted to numbers so that they can be added and divided
8. There was possibly one missing NA so that causes the entire thing to be NA
9. There's missing data and we can't add missing values to the total to average out

Student responses

1. theres some NA values
2. they use Total Ratings, not ratings
3. This did not work because it is dealing with NA values and cannot pass these through arithmetic functions so the entire thing will evaluate to NA. To fix this, we can have it ignore NA values
4. We have NA's somewhere in these columns, or that these original data are strings and not doubles.

Student responses

1. Can't mean rating because they're not a numeric value. They're categories for the ratings not an actual numerical value that can be summed.
2. It is currently a string, 1 rating 2 rating no the no . of starts ur giving it ratings_long |> mutate(rating = readr::parse_number(rating)) |> summarize(mean_rating = mean(rating, na.rm = TRUE))
3. It went because they aren't numeric values they're continuous
4. mean(rating) returns NA because rating contains text such as "1 ratings" rather than numbers. mean() cannot average text. Extract the numeric values first:
5. might be stored in a different format like text or missing altogether, meaning mean can't compute the average since it only works on numerics and logicals
6. rating column doesn't have any values in it need to use n column
7. rating column is text and not numbers, mean only works on numbers so it returns NA
8. rating is not a number. It is a string
9. ratings are now "1 rating", "2 ratings", etc, they are strings, which couldn't be averaged

Student responses

1. the current rating column is an a string from 1 rating to 10 rating, not a numerical value. As a result, ratings_long prints out a NA vallue.
2. The rating column contains text (like "1 ratings") instead of numbers. Since mean() only works with numeric values, we need to convert the ratings into numbers before calculating the average.
3. The rating column contains the string name of the rating type, rather than the numerical value.
4. The rating column contains the titles of the former rating columns, not the vote counts themselves.
5. The warning means rating is not stored as a numeric variable. It is probably character or factor text, so mean() cannot average it and returns NA. Common causes: - values like "4 stars" instead of 4 - factor labels - mixed text and numbers - missing values written as text Check it with: class(rating) str(rating)
6. They are not numbers they are categories.

Student responses

1. Because it should be ratings, not rating, without an s it can be very different
2. Because rating isn't the correct name of the column, "ratings" is. And it probably needs to be in quotations.
3. Because the mean should be calculating "ratings" instead of "rating"
4. incorrect data type
5. It is NA because here "rating" is the count of the each starts while in this function we want to actually find the average of star values not how many ratings there are
6. It returns an NA instead of an average because the values aren't numbers.
7. It says that the argument is not numeric or logical. it doesn't know what "rating" is
8. the rating object wasn't created with proper values
9. they didn't use the right name

Student responses

1. we called it ratings not rating, so when I fixed it I got 14422.
2. We changed rating to star, so it cant find rating anymore?
3. We probably averaging a column that is not the actual star rating. A result like 14422 suggests rating contains something else, such as counts, IDs, or badly parsed text turned into numbers.
4. We used the wrong variable for ratings

Student responses

1. First the data is grouped by title so it is trying to give the mean rating for all movies and so it is possible that movie doesn't have data at a certain value causing it to be NA
2. It didn't take into account the value of the ratings ie 1 star, 2 star, etc. Because of this, it returned NA instead of the proper average
3. not grouped or calculated right
4. since there is no group?
5. The R didn't find the pip of rating inside it, which make it has a R Error
6. the ratings are now in each column and not a single one, you can't mean a string
7. The ratings aren't grouped by movie yet, so the average for a given movie can't be calculated, as they are also in text format
8. This went wrong because it was not weighting the scores correctly and then dividing by the total number of votes. It also needs to group by movie, so that it preserves ratings for each movie
9. We need to group by movie title, also the movies are text so we cannot calculate the mean

Student responses

1. do not use is na
2. I think rating is stored as factor instead of numeric variable, or mabe do na.rm = true
3. Maybe there was a 0 in the data somewhere? I'm not getting the same NA that is supposed to be happening so I'm not sure.
4. The rating columns are getting grouped together?
5. there are too many different rating columns it doesn't know which one we mean to group by

Student responses

1. i didnt get an error
2. it worked for me
3. The code actually works for me and I am given the value of 14422.

Student responses

1. answer is text not #
2. Because we need to weight the number of counts for each rating for each movie before finding a true mean.
3. this is because we do not have the variable name of rating, and we should use weighted average

What would ``small_votes |> summarize(mean(rating))`` produce? What is the actual average across the four people? Explain why the answers differ.

84 anonymous responses

Gives numeric difference and result 4

19

Student states '4' and mentions a difference of 1.5, implying an attempted calculation or comparison but unclear reasoning.

Unsure or no answer given

19

Student indicates they do not know or cannot determine the result.

Explains 6 vs 4 due to weighting

17

Student states the summarize result is 6 but that the true average across four people is 4, and explains the discrepancy by noting the mean calculation ignored frequencies or weights.

Says it prints 6 (no explanation)

7

Student asserts the output is 6 without additional reasoning or mention of the actual average.

Response themes for question 11

Says summarize returns 6

7

Student asserts the summarize call yields 6, implying they computed an unweighted mean or sum rather than the per-person average.

Restates the code without answer

6

Student simply repeats the code snippet and does not provide a numerical result or explanation.

Explicit weighted-average calculation noted

5

Student explicitly shows how to compute the true average using weights (sum of rating×votes divided by total votes) and identifies the code's error of averaging distinct rating labels instead of accounting for vote counts.

Gives a numeric average (4)

4

Student reports the average as 4 (interpreting result as four stars) without explanation of discrepancy.

Student responses

1. ``mean(rating)` <dbl> 1 6`, but actual is 4. The mean didn't take votes into account
2. 1.5, 5.5. the actual mean is 4
3. 4, it is was 1.5 less
4. 6 and 4, 6 because there is more equal weight, 4 because the weightings are different
5. 6, averages 2 numbers in rating column, ignoring votes column. differs bc mean
6. 6,4
7. 6; 4; unweighted vs weighted average
8. average two rows
9. It prints out 6, while it should have printed 4.

Student responses

1. It produced the average of the numbers "2" and "10" instead of the average including the counts for each rating
2. It would give a mean of the ratings. The actual average is 3, and they differ because of a bug in the code.
3. it would give the mean of 2 and 10 which is 6, but the actual mean is 4. These differ because it is not taking into account how many time each rating is made
4. `Mean(rating)` only takes the mean of the values in the rating column. instead, we want to weight 1, 2, ... 10 by the actual number of people who gave that rating of 1, 2, ..., 10
5. returns 6, actual 4, it is a weighted mean
6. `small_votes |> summarize(mean(rating))` would take the mean of the ratings in the table, not weighted by how many people gave each rating. So if the table has one row for 2 stars and one row for 10 stars, it would give $(2 + 10) / 2 = 6$. The actual average across the four people is: $(2 + 2 + 2 + 10) / 4 = 16 / 4 = 4$.
7. `summarize(mean(rating))` would give 6, as the rating col adds to 12, which divided by 2 is 6. The actual avg is 8.

Student responses

1. This would produce a mean rating of 6, as it only accounts for two numbers. The correct number should be 4.44, it's not weighted.
2. Weighted average
3. you would get 6, but the answer we are looking for is 2, because we are averaging the rating values not the most common rating

Student responses

1. 6
2. gives you the mean between all 4 ratings, averages ratings equally no weighting
3. idk
4. It returns 6, The actual should be $6 + 10/2$ which is 8. So for some reason it downgrades it but I think it takes straight average of 2 + 10 and doesn't consider the votes
5. It would give 6
6. It would just average all the numbers, add them up and divide, not give what
7. It would produce 6. However, the actual mean rating is 4. Simply using `small_votes |> summarize(mean(rating))` does not take the weight of frequency into account.
8. It would produce a mean of 4, whereas the actual average should be much closer to 2 because of the higher quantity of votes.
9. It would produce the mean of literally the set {2, 10}. The actual average would require finding the weighted average

Student responses

1. it would provide the mean per person rather than the mean overall
2. it would summarize the mean of ratings
3. Mean rating would give 6. The actual average is 4. They differ because mean(rating) just averages the rating categories that were given, while the actual average has to take into account the number of ratings in each category
4. not sure
5. summarize would return 6. the actual average is 4. they would differ because of weighting.
6. That code results in the number 6, while the actual average is 4. This is because you are only taking the mean of the ratings column, without taking into account how many people submitted that rating.
7. The average would be a 4/10?
8. the code produces a mean rating of 6, which is different from the actual average of 4. These differ because the code is doing $(2+10)/2$ instead of $(2+2+2+10)/4$
9. The rating mean should be 8

Student responses

1. You would get 6, not the actual average rating. You need to multiply each rating by n number of ratings before you take the average.

Student responses

1. actual average is 4, summarise only calculates means of rating but not of votes it got
2. average/mean is 4 but it would give 6. Code is adding up ratings then dividing. It is not calculating correctly
3. it doesn't take into account how many people voted
4. it would give 6 as it doesn't take into consideration that 3 people voted 2, making the real mean 4
5. It would give 6 as the actual value instead of the correct value of 4 since it does not account for the frequency of the votes
6. It would give 6. The actual mean should be 4, however we don't get that because the mean function only finds the mean between the two values given, it doesn't factor in the frequency of those values.
7. It would give an average of 6 because $10+2 / 2 = 6$. The actual average rating is 4, because the value 2 got 3x the number of votes as the value 10
8. it would produce 6 because it takes the average of the rating labels, as if there were only 1 person who voted for each. The real answer is 4
9. It would produce 6 because the average of 10 and 2 is 6. The actual average is 4

Student responses

1. it would produce 6, the average of $2 + 10 / 2$, but the actual answer is 4. The answers differ because the code does not take into account the number of times each rating was given.
2. It would take the mean rating of the four people, and it differs from the actual mean because of its calculation methods.
3. It wouldn't count all the 2s it would just give 6. instead of 4.
4. mean rating would give you a 6, but you have to take a weighted average, meaning for each row, rating * votes summed up divided by sum(votes) to get the actual average across the four people
5. The code would give us 6 because it is the average of 2 and 10 because it is not factoring in the number of votes. the actual mean should be 4.
6. The code would produce an average of 6, but the actual average across all four people is 4. This is because mean() treats both ratings equally instead of accounting for how many people gave each rating.
7. The command produces 6, but the actual average is 4. The difference is sourced from failure to take number of votes into consideration.

Student responses

1. You would get 6. The mean(rating) just finds the mean of the numerical labels "2" and "10" without looking at the actual number of votes.

Student responses

1. 6, but the actual should be 4
2. 6, the different sample gives different answer
3. it prints 6
4. it would produce 6
5. it would produce 6 because it is asking the mean of 10 and 2
6. it would produce 6. actual average would 6. it differs as the previous one did not count the numeric values
7. The code gives 6 as the average rating but the real average rating is 4

Student responses

1. 1
2. I got 6, but it should be 4.
3. It would give 6 but it would need to give 4
4. It would yield 6. It doesn't take the average weighted by the number of votes.
5. `mean(rating)` gives 6, but the average is actually 4. They are different because `mean(rating)` treats each rating level equally and does not account for the number of votes.
6. they differ because way r averages out each one
7. we would need to do a weight average since not all points count the same

Student responses

1. Actual avg is 4, but this will give 6 bc its taking each value as its own rating even the number of votes.
2. If you did `summarize(mean(rating))`, it would calculate 6 as the average. If you did `votes` instead, it would calculate 2 as the mean.
3. It would give 6 because it just averages 2 and 10, but the real answer would be $(2*3 + 10*1) / (3 + 1)$ which would be 4. It needs to be weighted.
4. It would give us the mean between 2 and 10 ratings which would be weighted and give 4
5. `small_votes |> summarize(mean(rating))`
6. you need to take a weighted average

Student responses

1. give the average value of rating
2. It is missing out on what the total actually is for the number of votes casts. It is missing out on having this figure change to reflect votes for only that specific movie.
3. It would produce 6 but we would expect 4 and this happens because it does 2 times 3 but we want 4. The votes have different weights
4. ratings of either 2 or 10, with 3 or 1 vote
5. The code would just give the mean between 2 and 10 which is 6. But the actual mean should be $2 + 2 + 2 + 10 / 4$, because we need to take in consideration of the number of people who voted each rating.

Student responses

1. It would return $16/4$ so an average of 4
2. mean rating be 4
3. small votes would probably say the mean is 6. the actual value should be 4
4. The actual average would be 4 stars but the small votes would produce

Improve this scatterplot: add a descriptive title, labeled axes, and a source. Make overlapping points easier to see. You can use the ? button to ask AI for help. Run your code, then submit your plot so we can compare designs.

72 anonymous responses

Other responses

72

Student responses

- 1.
- 2.
- 3.
- 4.
- 5.
- 6.
- 7.
- 8.
- 9.

Student responses

- 1.
- 2.
- 3.
- 4.
- 5.
- 6.
- 7.
- 8.
- 9.

Student responses

- 1.
- 2.
- 3.
- 4.
- 5.
- 6.
- 7.
- 8.
- 9.

Student responses

- 1.
- 2.
- 3.
- 4.
- 5.
- 6.
- 7.
8. add color
9. added axis labels

Student responses

1. added title, axis labels
2. Added titles to it and tried to make it easier to see overlapping
3. Added titles, grid layout, better domain and range spacing intervals
4. Added tittle
5. change the lable
6. Change the title to Movie Rating vs. Imdb rating, and the axis' titles to match that.
7. Change the title, etc.
8. change x axis to mean rating, y axis to imdb rating
9. Example in ggplot2: `ggplot(movies, aes(x = raw_avg_vote, y = imdb_score)) + geom_point(alpha = 0.5, color = "steelblue", size = 2) + labs(title = "Raw average votes and IMDb scores for movies", subtitle = "Each point represents one movie", x = "Raw average vote", y = "IMDb score", caption = "Source: movie ratings dataset used in class") + theme_minimal()`

Student responses

1. `ggplot(movies, aes(x = raw_avg, y = imdb_score)) + geom_point(color = "tomato", size = 3, alpha = 0.6) + labs( title = "Raw average vs IMDb score", x = "Raw average score", y = "IMDb score", caption = "Source: movies dataset" )`
2. Here is the updated plot.
3. i added the labels but had a hard time reducing the clutter and noise towards the upper right hand corner
4. I changed the title and named the axes
5. I have added axis and make it clearer
6. labels for all axis and title)messed up code)
7. `library(ggplot2) ggplot(movies, aes(x = audience_score, y = imdb_rating)) + geom_point(alpha = 0.5, color = "steelblue") + labs( title = "Audience score vs IMDb rating for movies", x = "Audience score", y = "IMDb rating", caption = "Source: movie ratings dataset" ) + theme_minimal()`
8. `movie_means |> ggplot(aes(x = mean_rating, y = imdb_rating)) + geom_abline(slope = 1, intercept = 0, color = "gray60", linewidth = 0.8) + geom_point(alpha = 0.15, size = 1.5, color = "steelblue") + labs(x = "Raw average rating (1-10)", y = "IMDb rating (1-10)", title = "Raw average rating vs IMDb rating") + theme_minimal()`

Student responses

1. `plot( movie_means$mean_rating, movie_means$imdb_rating, main = "Average movie rating vs IMDb rating", xlab = "Mean raw audience rating", ylab = "IMDb rating", pch = 16, col = rgb(0, 0.4, 0.8, 0.5) ) abline( lm(imdb_rating ~ mean_rating, data = movie_means), col = "red", lwd = 2 )`
2. `plot(movie_means$mean_rating, movie_means$imdb_rating, main = "Average movie rating vs IMDb rating", xlab = "Raw average rating", ylab = "IMDb rating", pch = 16, col = rgb(0, 0, 1, 0.4)) abline(lm(imdb_rating ~ mean_rating, data = movie_means), col = "red", lwd = 2) mtext("Source: movie_means data", side = 1, line = 4, cex = 0.8)`
3. `plot( movie_means$mean_rating, movie_means$imdb_rating, main = "Calculated Mean Rating vs IMDb Rating", xlab = "Calculated Mean Rating", ylab = "IMDb Rating", pch = 16, cex = 0.6, col = rgb(0, 0, 0, 0.25) ) mtext("Source: IMDb dataset", side = 1, line = 4, adj = 1, cex = 0.8)`
4. `plot( movie_means$mean_rating, movie_means$imdb_rating, main = "Computed Mean Rating vs. IMDb Rating", xlab = "Computed Mean Rating", ylab = "IMDb Rating" )`

Student responses

1. `plot( movie_means$mean_rating, movie_means$imdb_rating, main = "Raw average rating vs IMDb rating for movies", xlab = "Raw average rating", ylab = "IMDb rating", pch = 16, col = rgb(0, 0, 1, 0.4) )`
2. `plot( movie_means$mean_rating, movie_means$imdb_rating, main = "Raw average rating vs IMDb rating for movies", xlab = "Raw average rating", ylab = "IMDb rating", pch = 16, col = rgb(0, 0, 1, 0.4) ) mtext("Source: movie_means data", side = 1, line = 3, cex = 0.8)`
3. `plot( movie_means$mean_rating, movie_means$imdb_rating, main = "Raw average rating vs IMDb rating", xlab = "Raw average rating", ylab = "IMDb rating", pch = 16, col = rgb(0, 0, 1, 0.4) ) mtext("Source: movie_means data", side = 1, line = 4, cex = 0.`
4. `plot( movie_means$mean_rating, movie_means$imdb_rating, main = "Raw average vs IMDb score for movies", xlab = "Raw average rating", ylab = "IMDb rating", pch = 16, col = rgb(0, 0, 1, 0.4) )`
5. `plot(movie_means$mean_rating, movie_means$imdb_rating, main = 'IMDB Rating vs Average Score', xlab = 'Average Score', ylab = 'IMDB Rating', col = 'forestgreen')`

Student responses

1. `plot(movie_means$mean_rating, movie_means$imdb_rating, main = "Mean Movie Ratings on IMDB", xlab = "raw avg rating", ylab = "IBDB Rating")`
2. `plot(movie_means$mean_rating, movie_means$imdb_rating, main = "Mean Rating vs IMDB Rating, from the IMDB Database", xlab = "Mean Rating", ylab = "IMDB Rating", col = "blue")`
3. `plot(movie_means$mean_rating, movie_means$imdb_rating, main="Imdb rating vs Mean rating", xlab="mean rating", ylab='imdb rating')`
4. `plot(movie_means$mean_rating, movie_means$imdb_rating, xlab = 'Mean rating')`
5. `plot(movie_means$mean_rating, movie_means$imdb_rating, xlab = "Mean rating", ylab = "IMDb rating", main = "Mean Rating vs IMDb Rating")`
6. `plot(movie_means$mean_rating, movie_means$imdb_rating, xlab = "Raw average rating", ylab = "IMDb rating", main = "Raw average vs IMDb rating")`

Student responses

1. SLIDE 29 Ask about this slide Close Your question Ask for an explanation, R code, or help with this slide. Ask Your questions Slide 29
improveplot(movie_means\$mean_rating, movie_means\$imdb_rating) Try adding labels, a title, a source note, and transparency so overlapping points are easier to see. improveplot(movie_means\$mean_rating, movie_means\$imdb_rating) + labs(title = "Raw average rating vs IMDb rating for movies", x = "Raw average rating", y = "IMDb rating", caption = "
2. Spacing
3. Still needs a title
4. This graph could be improved with a title

What does a point above or below the 45-degree line tell us about its two scores? Why might such discrepancies occur?

0 anonymous responses

No responses were submitted.

- Start with all movies in `movie\_means` and calculate `gap` as `mean\_rating` minus `imdb\_rating`. - Order movies by the absolute value of `gap`, largest first. - Keep the movie with the largest absolute gap. - Show its title, release year, computed mean rating, IMDb rating, and `gap`.

79 anonymous responses

Other responses

79

Student responses

1. `movie_means |> mutate(gap = mean_rating - imdb_rating) arrange(desc(abs(gap)))`
2. `movie_means |> mutate(gap = mean_rating - imdb_rating) |> arrange(desc(abs(gap))) |> slice_head(n = 1) |> select(Title, Year, mean_rating, imdb_rating, gap)`
3. `movie_means |> mutate(gap = mean_rating - imdb_rating) |> arrange(desc(abs(gap))) |> slice_head(n = 1) |> select(Title, Year, mean_rating, imdb_rating, gap)`
4. `movie_means |> mutate(gap = mean_rating - imdb_rating) |> arrange(desc(abs(gap))) |> slice_head(n = 1) |> select(Title, Year, mean_rating, imdb_rating, gap)`
5. `movie_means |> mutate(gap = mean_rating - imdb_rating) |> arrange(desc(abs(gap))) |> slice_head(n = 1) |> select(Title, Year, mean_rating, imdb_rating, gap)`
6. `movie_means |> mutate(gap = mean_rating - imdb_rating) |> arrange(desc(abs(gap))) |> slice_head(n = 1) |> select(Title, Year, mean_rating, imdb_rating, gap)`
7. `movie_means |> mutate(gap = mean_rating - imdb_rating) |> arrange(desc(abs(gap))) |> slice_head(n = 1) |> select(Title, Year, mean_rating, imdb_rating, gap)`

Student responses

1. `movie_means |> mutate(gap = mean_rating - imdb_rating) |> arrange(desc(abs(gap))) |> slice_head(n = 1) |> select(Title, Year, mean_rating, imdb_rating, gap)`
2. `movie_means |> mutate(gap = mean_rating - imdb_rating) |> arrange(desc(abs(gap))) |> slice_head(n = 5) |> select(Title, Year, mean_rating, imdb_rating, gap)`
3. `movie_means |> mutate(gap = mean_rating - imdb_rating) |> arrange(desc(abs(gap))) |> slice(1) |> select(title, year, mean_rating, imdb_rating, gap) mutate()
abs(gap) arrange(desc(...)) slice(1) select()`
4. `movie_means |> mutate(gap= mean_rating - imdb_rating) |> arrange(desc(abs(gap))) |> slice_head(n=1) |> select(Title, Year, mean_rating, imdb_dating, gap)`
5. `movie_means |> mutate(rating_diff = imdb_rating - mean_rating) |> arrange(desc(abs(rating_diff))) |> slice_head(n = 1) |> select(Title, Year, mean_rating,
imdb_rating, rating_diff)`
6. `movie_means |> mutate(gap = mean_rating - imdb_rating) |> arrange(desc(abs(gap))) |> slice_head(n = 1)|> select(Title, Year, mean_rating, imdb_rating, gap)`

Student responses

1. `movie_means |> mutate(gap = (mean_rating - imdb_rating) |> arrange(desc(abs(gap))) |> slice_head(n = 1)`
2. `movie_means |> mutate(gap = mean_rating - imdb_rating)`
3. `movie_means |> mutate(gap = mean_rating - imdb_rating) |> arrange(desc(abs(gap))) |> slice_head(n = 1) |> select(Title, Year, mean_rating, imdb_rating, gap)`
4. `movie_means |> mutate(gap = mean_rating - imdb_rating) |> arrange(desc(abs(gap))) |> slice_head(n = 1) |> select(Title, Year, mean_rating, imdb_rating, gap)`
5. `movie_means |> mutate(gap = mean_rating - imdb_rating) |> arrange(desc(abs(gap))) |> slice_head(n=1) |> select(Title, Year, mean_rating, imdb_rating, gap)`
6. `movie_means |> mutate(gap = mean_rating - imdb_rating) |> arrange(desc(abs(gap))) |> slice_head(n=1) |> select(Title, Year, mean_rating, imdb_rating, gap)`
7. `movie_means |> mutate(gap = mean_rating - imdb_rating) |> arrange(desc(abs(gap))) |> slide_head(n=1)`
8. `movie_means |> mutate(gap = mean_rating - imdb_rating) |> arrange(desc(abs(gap))) |> slice_head(n=10) |> select(Title, gap, Year, mean_rating, imdb_rating)`

Student responses

1. `movie_means |> mutate(gap=mean_rating - imdb_rating) |> arrange(desc(abs(gap))) slice_head(n = 1) select(Title,year, mean_rating, imdb_rating, gap)`
2. `movie_means |> mutate(gap = abs(mean_rating - imdb_rating)) |> arrange(desc(gap)) |> slide_head(n = 1) |> select(Title, Year, mean_rating, imdb_rating, gap)`
3. `movie_means |> mutate(gap = mean_rating - imdb_rating) |> arrange(desc(abs(gap)))`
4. `movie_means |> mutate(gap = mean_rating - imdb_rating) |> arrange(desc(abs(gap))) |> select(Title, Year, mean_rating, imdb_rating, gap) |> head(1)`
5. `movie_means |> mutate(gap = mean_rating - imdb_rating) |> arrange(desc(abs(gap))) |> head(1) |> select(Title, Year, mean_rating, imdb_rating, gap)`
6. `movie_means |> mutate(gap=mean_rating - imdb_rating) head()`
7. `movie_means |> mutate(gap = mean_rating - imdb_rating) arrange(desc(abs(gap))) |> slice_head(n=1) |> select(Title,Year, mean_rating, imdb_rating, gap)`
8. `movie_means |> mutate(gap = mean_rating - imdb_rating) |> arrange(desc("gap"))`

Student responses

1. `movie_means |> mutate(gap = mean_rating - imdb_rating) |> arrange(desc(abs(gap))) |> slice_head(n = 1) select('Title', 'Year', 'mean_rating', 'imdb_rating', 'gap')`
2. `movie_means |> mutate(gap = mean_rating - imdb_rating) |> arrange(desc(abs(gap))) |> slice_head(n = 1) |> select(Title, Year, mean_rating, imdb_rating, gap)`
3. `movie_means |> mutate(gap = mean_rating - imdb_rating) |> arrange(desc(abs(gap))) |> slice_head(n = 1) |> select(Title, Year, mean_rating, imdb_rating, gap)`
4. `movie_means |> mutate(gap = mean_rating - imdb_rating) |> arrange(desc(abs(gap))) |> slice(head(n = 1) |> select(Title, Year, mean_rating, imdb_rating, gap)`
5. `movie_means |> mutate(gap = mean_rating - imdb_rating) |> arrange(desc(abs(gap))) |> slice_head(n=1) |> select(Title, Year, mean_rating, imdb_rating, gap)`
6. `movie_means |> mutate(gap = mean_rating - imdb_rating) |> arrange(desc(abs(gap)))|> slice_head(n = 1)|> select(Title, Year, mean_rating, imdb_rating)`

Student responses

1. `movie_means |> mutate (gap = `mean_rating` - `imdb_rating`) |> arrange(desc(abs(gap))) |> slice_head(n=1)|> select(Title, Year, mean_rating, gap, imdb_rating)`
2. `movie_means |> mutate (gap = mean_rating - imbd_rating) |> arrange(desc(abs(gap))) |> slice_head(n = 1) |> select(Title, Year, mean_rating, imdb_rating, gap)`
3. `movie_means |> mutate (gap = mean_rating - imdb_rating) |> arrange(desc(abs(gap))) |> head(n=1) |> select(Title, Year, mean_rating, imdb_rating, gap)`
4. `movie_means |> mutate( gap = mean_rating - imdb_rating) |> arrange(desc(abs(gap))) ) |> slice_head(n=1) |> select(Title, Year, mean_rating, imdb_rating, gap)`
5. `movie_means |> mutate( gap = mean_rating - imdb_rating) |> arrange(desc(abs(gap))) |> slice_head(n = 1) |> select(Title, Year, mean_rating, imdb_rating, gap)`
6. `movie_means |> mutate(gap = `mean_rating` - `imdb_rating`) |> arrange(desc(abs(gap))) |> slice_head(n=1) |> select(Title, Year, imdb_rating, mean_rating, gap)`
7. `movie_means |> mutate(gap = mean_rating - imdb_rating) |> arrange(desc(abs(gap))) |> head(1) |> select(`Title`, Year, mean_rating, imdb_rating, gap)`

Student responses

1. `movie_means |> mutate(gap = mean_rating - imdb_rating) |> arrange(desc(abs(gap))) |> head(1) |> select(Title, Year, mean_rating, imdb_rating, gap)`
2. `movie_means |> mutate(gap = mean_rating - imdb_rating) |> arrange(desc(abs(gap))) |> head(n = 1) |> select(Title, Year, imdb_rating, mean_rating, gap)`
3. `movie_means |> mutate(gap = mean_rating - imdb_rating) |> arrange(desc(abs(gap))) |> head(n=1) |> select(Title, Year, mean_rating, imdb_rating, gap)`
4. `movie_means |> mutate(gap = mean_rating - imdb_rating) |> arrange(desc(abs(gap))) |> slice_head(n = 1) |> select(Title, Year, mean_rating, imdb_rating, gap)`
5. `movie_means |> mutate(gap = mean_rating - imdb_rating) |> arrange(desc(abs(gap))) |> slice_head(n = 1) |> select(Title, Year, mean_rating, imdb_rating, gap)`
6. `movie_means |> mutate(gap = mean_rating - imdb_rating) |> arrange(desc(abs(gap))) |> slice_head(n = 1) |> select(Title, Year, mean_rating, imdb_rating, gap)`
7. `movie_means |> mutate(gap = mean_rating - imdb_rating) |> arrange(desc(abs(gap))) |> slice_head(n = 1) |> select(Title, Year, mean_rating, imdb_rating, gap)`

Student responses

1. `movie_means |> mutate(gap = mean_rating - imdb_rating) |> arrange(desc(abs(gap))) |> slice_head(n = 1) |> select(Title, Year, mean_rating, imdb_rating, gap)`
2. `movie_means |> mutate(gap = mean_rating - imdb_rating) |> arrange(desc(abs(gap))) |> slice_head(n=1) |> select(Title, Year, mean_rating, imdb_rating, gap)`
3. `movie_means |> mutate(gap = mean_rating - imdb_rating) |> arrange(desc(abs(gap))) |> slice_head(n=1) |> select(Title, Year, mean_rating, imdb_rating, gap)`
4. `movie_means |> mutate(gap = mean_rating - imdb_rating) |> arrange(desc(abs(gap))) |> slice_head(n=1) |> select(Title, Year, mean_rating, imdb_rating, gap)`
5. `movie_means |> mutate(gap = mean_rating - imdb_rating) |> arrange(desc(abs(gap))) |> slice_head(n=1) |> select(Title, Year, mean_rating, imdb_rating, gap)
#head()`
6. `movie_means |> mutate(gap = mean_rating - imdb_rating) |> arrange(desc(abs(gap))) |> slice_head(n=1) |> select(Title, Year, mean_rating, imdb_rating, gap)`
7. `movie_means |> mutate(gap = mean_rating - imdb_rating) |> arrange(desc(gap))`

Student responses

1. `movie_means |> mutate(gap = mean_rating - imdb_rating) |> arrange(desc(abs(gap))) |> slice_head(n = 1) |> select(Title, Year, mean_rating, imdb_rating, gap)`
2. `movie_means |> mutate(gap = mean_rating - imdb_rating)|> arrange(desc(abs(gap)))|>slice_head(n=1)|> select(Title,Year,mean_rating, imdb_rating, gap)`
3. `movie_means |> mutate(gap= mean_rating - imdb_rating) |> sort(desc(abs(gap))) |> slice_head(n=1)`
4. `movie_means |> mutate(rating_diff = mean_rating - imdb_rating) |> arrange(desc(abs(rating_diff))) |> slice_head(n=1) |> select(Title, Year, mean_rating, imdb_rating, rating_diff)`
5. `movie_means |> head()`
6. `movie_means |> head()`
7. `movie_means |> head() mutate(gap = mean_rating - imdb_rating)`
8. `movie_means |> head() movie_means %>% mutate(gap = mean_rating - imdb_rating) %>% arrange(desc(abs(gap))) %>% slice(1) %>% select(title, year, mean_rating, imdb_rating, gap)`
9. `movie_means |> head() movie_means %>% mutate(gap = mean_rating - imdb_rating) %>% arrange(desc(abs(gap))) %>% slice(1) %>% select(title, year, mean_rating, imdb_rating, gap)`

Student responses

1. `movie_means |> head() mutate(gap = mean_rating - imdb_rating)`
2. `movie_means |> head() mutate(gap = mean_rating - imdb_rating) |> arrange(desc(abs(gap))) |> slice_head(n = 1) |> select(Title, Year, mean_rating, imdb_rating, gap)`
3. `movie_means |> head() outliers <- movie_means |> mutate( gap = mean_rating - imdb_rating ) |> arrange(desc(abs(gap))) |> head(n=1) |> select(Title, Year, mean_rating, imdb_rating, gap) outliers`
4. `movie_means |> mutate(gap = mean_rating - imdb_rating) movie_means |> arrange(desc(abs(gap))) |> slice_head(n=1) |> select(Title, Year, mean_rating, imdb_rating, gap)`
5. `movie_means |> mutate(gap = mean_rating - imdb_rating) |> arrange(desc(abs(`gap`))) |> slice_head(n = 1)`
6. `movie_means |> mutate(gap = mean_rating - imdb_rating) |> arrange(desc(abs(gap))) |> slice_head(n = 1) |> select(Title, Year, mean_rating, imdb_rating, gap)`
7. `movie_means |> mutate(gap = mean_rating - imdb_rating) |> arrange(desc(abs(gap))) |> slice_head(n = 1) |> select(Title, Year, mean_rating, imdb_rating, gap)`

Student responses

1. `movie_means |> mutate(gap = mean_rating - imdb_rating) |> mutate(abs_gap = abs(gap)) |> arrange(desc(abs_gap)) |> slice_head(n=1)`
2. `movie_means |> mutate(gap = mean_rating - imdb_rating)|> arrange(desc(abs(gap))) |> filter(gap == max(abs(gap))) |> select(Title, Year, mean_rating, imdb_rating, gap)`
3. `movie_means |> mutate(gap=mean_rating - imdb_rating) |> arrange(desc(abs(gap))) |> slice_head(n=1) |> select(Title, Year, mean_rating, imdb_rating, gap)`
4. `movie_means |> mutate(gap=mean_rating-imdb_rating) |> arrange(desc(abs(gap))) |> slice_head(n=1) |> select(Title,Year,mean_rating, imdb_rating,gap)`
5. `movie_means|> mutate(gap = `mean_rating` - `imdb_rating`) |> arrange(desc(abs(gap))) |> slice_head(n = 1) |> select(Title, Year, mean_rating, imdb_rating, gap)`
6. `movie_means|> mutate(gap = mean_rating - imdb_rating)|> arrange(desc(abs(gap))) |> head(1) |> select(Title, Year, mean_rating, imdb_rating, gap)`
7. `result<-movie_means |> mutate(gap=mean_rating-imdb_rating)|> arrange(desc(abs(gap)))|> slice(1)|> select(Title, Year, mean_rating, imdb_rating, gap) result`

- Group the movies by `decade`. - Summarize into one row per decade, including the number of movies as `n\_movies`. - Calculate `avg\_rating` as the mean of `imdb\_rating` within each decade, giving every movie equal weight.

79 anonymous responses

Other responses

79

Student responses

1. `dec_result<-imdb_decade |> group_by(decade)|> summarize(n_movies=n(),avg_rating=mean(imdb_rating)) dec_result`
2. `imdb_decade |> group_by(decade) |> summarize(mean_rating = mean(imdb_rating), n_movies = n())`
3. `imdb_decade`
4. `imdb_decade |> group_by(decade) |> summarize( n_movies = n(), avg_rating = mean(imdb_rating, na.rm = TRUE), .groups = "drop" )`
5. `imdb_decade |> group_by(decade) |> summarize( n_movies = n(), avg_rating = mean(imdb_rating) )`
6. `imdb_decade |> group_by(decade) |> summarize( n_movies = n(), avg_rating = mean(imdb_rating) )`
7. `imdb_decade |> group_by(decade) |> summarize(mean_rating = mean(imdb_rating), n_movies = n())`
8. `imdb_decade |> imdb_decade |> group_by(decade) |> summarize( n_movies = n(), avg_rating = mean(imdb_rating) )`
9. `imdb_decade |> group_by(decade) |> summarize(mean_rating = mean(imdb_rating) , n_movies = n())`

Student responses

1. `imdb_decade |> group_by(decade) |> summarize( n_movies = n(), avg_rating = mean(imdb_rating) )`
2. `imdb_decade |> group_by(decade) |> summarize(avg_rating = mean(imdb_rating), n_movies = n())`
3. `imdb_decade |> group_by(decade) |> summarize(avg_rating = mean(imdb_rating), n_movies = n())`
4. `imdb_decade |> group_by(decade) |> summarize(avg_rating = mean(imdb_rating), n_movies = n())`
5. `imdb_decade |> group_by(decade) |> summarize(mean_rating = mean(imdb_rating), n_movies = n())`
6. `imdb_decade |> group_by(decade) |> summarize(n_movies = n(), avg_rating = mean(imdb_rating)) |> print()`
7. `imdb_decade |> group_by(decade) |> summarize(nmovies = n(), avg_rating = mean(imdb_rating))`
8. `imdb_decade |> group_by(decade) |> summarize(mean_rating = mean(imdb_rating), n_movies = n())`
9. `imdb_decade |> group_by(decade) |> summarize(n_movies = n(), avg_rating = mean(imdb_rating))`

Student responses

1. `imdb_decade |> group_by(decade) |> summarize(avg_rating = mean(imdb_rating), n_movies = n()) |> select(decade, avg_rating, n_movies)`
2. `imdb_decade |> group_by("decade") |> summarize(mean_rating = mean(imdb_rating), n_movies = n())`
3. `imdb_decade |> group_by(decade) |> summarize( mean_rating = mean(imdb_rating), n_movies = n() )`
4. `imdb_decade |> group_by(decade) |> summarize( n_movies = n(), avg_rating = mean(imdb_rating, na.rm = TRUE) )`
5. `imdb_decade |> group_by(decade) |> summarize(mean_rating = mean(imdb_rating), n_movies = n())`
6. `imdb_decade |> group_by(decade) |> summarize(mean_rating = mean(imdb_rating), n_movies = n())`
7. `imdb_decade |> group_by(decade) |> summarize(mean_rating = mean(imdb_rating), n_movies = n())`
8. `imdb_decade |> group_by(decade) |> summarise( mean_rating = mean(imdb_rating), n_movies = n(), avg_rating = mean(imdb_rating, na.rm = TRUE) )`
9. `imdb_decade |> group_by(decade) |> summarise(n_movies = n(), avg_rating = mean(imdb_rating)) head()`

Student responses

1. `imdb_decade |> group_by(decade) |> summarize( mean_rating = mean(imdb_rating), n_movies = n() )`
2. `imdb_decade |> group_by(decade) |> summarize( mean_rating = mean(imdb_rating), n_movies = n())`
3. `imdb_decade |> group_by(decade) |> summarize( nmovies = n(), avg_rating = mean(imdb_rating) )`
4. `imdb_decade |> group_by(decade) |> summarize(avg_rating = mean(imdb_rating), n_movies = n()) |> head(n = 10)`
5. `imdb_decade |> group_by(decade) |> summarize(mean_rating = mean(imdb_rating), n_movies = n())`
6. `imdb_decade |> group_by(decade) |> summarize(mean_rating = mean(imdb_rating), n_movies = n())`
7. `imdb_decade |> group_by(decade) |> summarize(mean_rating = mean(imdb_rating), n_movies = n())`
8. `imdb_decade |> group_by(decade) |> summarize(mean_rating = mean(imdb_rating), n_movies = n())`
9. `imdb_decade |> group_by(decade) |> summarize(mean_rating = mean(imdb_rating), n_movies = n())`

Student responses

1. `imdb_decade |> group_by(decade) |> summarize(mean_rating = mean(imdb_rating), n_movies = n())`
2. `imdb_decade |> group_by(decade) |> summarize(mean_rating = mean(imdb_rating), n_movies=n())`
3. `imdb_decade |> group_by(decade) |> summarize(mean_rating = mean(imdb_rating), n+movies = n()) |>`
4. `imdb_decade |> group_by(decade) |> summarize(mean_rating=mean(imdb_rating), n_movies=n())`
5. `imdb_decade |> group_by(decade) |> summarize(mean(imdb_rating), n_movies = n()) #head()`
6. `imdb_decade |> group_by(decade) |> mutate(n_movies = n(), mean_rating = mean(imdb_rating)) |> select(decade, mean_rating, n_movies)`
7. `imdb_decade |> group_by(decade) |> summarize( n_movies = n(), avg_rating = mean(imdb_rating) )`
8. `imdb_decade |> group_by(decade) |> summarize(avg_rating = mean(imdb_rating), n_movies = n())`
9. `imdb_decade |> group_by(decade) |> summarize(mean_rating = mean(imdb_rating, n_movies = n()))`

Student responses

1. `imdb_decade |> group_by(decade) |> summarize(mean_rating = mean(imdb_rating), n_movies = n())`
2. `imdb_decade |> group_by(decade) |> summarize(mean_rating = mean(imdb_rating), n_movies = n(),`
3. `imdb_decade |> group_by(decade) |> summarize(mean_rating = mean(imdb_rating), n_movies = n())`
4. `imdb_decade |> group_by(decade) |> summarize(mean_rating = mean(imdb_rating), n_movies = n())`
5. `imdb_decade |> group_by(decade) |> summarize(mean_rating = mean(imdb_rating), n_movies = n())`
6. `imdb_decade |> group_by(decade) |> n_movies = n()) summarize(avg_rating = mean(imdb_rating), n_movies = n())`
7. `imdb_decade |> group_by(decade) |> summarize(mean_rating = mean(imdb_rating), n_movies = n()) |> arrange(desc(mean_rating))`
8. `imdb_decade |> group_by(decade) |> summarize(mean_rating = mean(imdb_rating), n_movies = n())`
9. `imdb_decade |> group_by(decade) |> summarize(mean_rating = mean(imdb_rating), n_movies = n())`

Student responses

1. `imdb_decade |> group_by(decade) |> summarize(mean_rating = mean(imdb_rating), n_movies = n())`
2. `imdb_decade |> group_by(decade) |> summarize(mean_rating) = mean(imdb_rating), n_moves = n())`
3. `imdb_decade |> group_by(decade) |> summarize(mean_rating=mean(imdb_rating), n_movies=n())`
4. `imdb_decade |> group_by(decade) |> summarize(n_movies = n(), mean_rating = mean(imdb_rating))`
5. `imdb_decade |> group_by(decade) |> summarize(n_movies = n(), avg_rating = mean(imdb_rating))`
6. `imdb_decade |> group_by(decade) |> summarize(mean_rating = mean(imdb_rating), n_movies = n())`
7. `imdb_decade |> group_by(decade) |> summarize(mean_rating=mean(imdb_rating),n_movies=n())`
8. `imdb_decade |> group_by(decade) |> summarize( n_movies = n(), avg_rating = mean(imdb_rating) )`
9. `imdb_decade |> group_by(decade) |> summarize(avg_rating = mean(imdb_rating), n_movies = n()) hist(imdb_decade$avg_rating)`

Student responses

1. `imdb_decade |> group_by(decade) |> summarize(mean_rating=mean(imdb_rating), n_movies=n())`
2. `imdb_decade |> group_by(decade)|> summarize(mean_rating = mean(imdb_rating), n_movies = n())`
3. `imdb_decade |> group_by(decade)|> select(Title,decade)|> summarize(mean_rating=mean(imdb_rating), n_movies=n())`
4. `imdb_decade |> head()`
5. `imdb_decade |> head()`
6. `imdb_decade |> head()`
7. `imdb_decade |> head()`
8. `imdb_decade |> head() group_by(movies, decade)`
9. `imdb_decade |> head()|> group_by(decade) |> summarize(mean_rating = mean(imdb_rating), n_movies= n())`

Student responses

1. `imdb_decade |> head()movies %>% group_by(decade) %>% summarise( n_movies = n(), avg_rating = mean(imdb_rating, na.rm = TRUE) )` This groups by decade, counts movies in each decade, and computes the average IMDb rating with each movie weighted equally. If you want to check the result in a nice order, you could add: `movies %>% group_by(decade) %>% summarise( n_movies = n(), avg_rating = mean(imdb_rating, na.rm = TRUE) ) %>% arrange(decade)`
2. `imdb_decade |> print()`
3. `imdb_decade |> select(Title, decade)`
4. `imdb_decade |> select(Title, decade) |> group_by(decade) |> summarize(mean_rating = mean(imdb_rating), n_movies = n())`
5. `imdb_decade |> select(Title, decade) |> group_by(decade)|> summarize(mean_rating = mean(imdb_rating), n_movies = n())`
6. `imdb_decade |>group_by(decade) |> summarize(mean_rating = mean(imdb_rating), n_movies=n())`

Student responses

1. `result <- imdb_decade |> mutate( decade = cut( Year, breaks = breaks, labels = labels, right = FALSE ) ) |> group_by(decade) |> summarize( n_movies = sum(!is.na(decade)) )`
`result result2 <- imdb_decade |> mutate( decade = cut( Year, breaks = breaks, labels = labels, right = FALSE ) ) |> group_by(decade)`
`result2`

Improve this boxplot: add a descriptive title, labeled axes, readable decade labels, and a source. Keep the decades in chronological order. You can use the ? button to ask AI for help. Run your code, then submit your plot so we can compare designs.

67 anonymous responses

Other responses

67

Student responses

- 1.
- 2.
- 3.
- 4.
- 5.
- 6.
- 7.
- 8.
- 9.

Student responses

- 1.
- 2.
- 3.
- 4.
- 5.
- 6.
- 7.
- 8.
- 9.

Student responses

- 1.
- 2.
- 3.
- 4.
- 5.
- 6.
- 7.
- 8.
- 9.

Student responses

- 1.
- 2.
- 3.
- 4.
- 5.
- 6.
- 7.
- 8.
- 9.

Student responses

1. add better titles and labels axes
2. add labels rotated it
3. Added a title and more details
4. added title
5. Added title
6. Added titles and axes titles
7. `boxplot( imdb_rating ~ decade, data = imdb_decade, main = "IMDb Ratings by Release Decade", xlab = "Release Decade", ylab = "IMDb Rating", las = 2 )`
8. `boxplot( The imdb rating ~ The decade, data = imdb_decade)`. Color code the boxes too
9. `boxplot(imdb_rating ~ decade, data = imdb_decade, col = 'orange')`

Student responses

1. `boxplot(imdb_rating ~ decade, data = imdb_decade, horizontal = TRUE, main = "IMDb ratings by decade", xlab = "IMDb rating", ylab = "Decade", las = 1)`
2. `boxplot(imdb_rating ~ decade, data = imdb_decade, main = 'Ratings by Decade', xlab = 'Decade', ylab = 'IMDB Ratings' )`
3. `boxplot(imdb_rating ~ decade, data = imdb_decade, main = "Boxplot of Rating by Decade", xlab = "Decade", ylab = "IMDb Rating")`
4. `boxplot(imdb_rating ~ decade, data = imdb_decade, main = "Distribution of IMDb ratings by decade", xlab = "Decade", ylab = "IMDb rating", las = 2)`
`mtext("Source: IMDb movie data", side = 1, line = 4, cex = 0.8) las = 2` makes the decade labels easier to read by rotating them.
5. `boxplot(imdb_rating ~ decade, data = imdb_decade, main = "Distribution of IMDB Ratings for each decade, from the IMDB Database", xlab = "Decade", ylab = "IMDB Rating", col = "purple")`

Student responses

1. `boxplot(imdb_rating ~ decade, data = imdb_decade, main = "IMDb Ratings by Decade!!!", xlab = "Decade", ylab = "IMDb rating", las = 2, col = c("hotpink", "yellow", "cyan", "limegreen", "orange", "purple", "red", "blue"), border = "magenta") mtext("Source: IMDb data", side = 1, line = 4)`
2. `boxplot(imdb_rating ~ decade, data = imdb_decade, main = "IMDB Ratings by Decade", xlab = "Decade", ylab = "IMDB Rating")`
3. `boxplot(imdb_rating ~ decade, data = imdb_decade, main="IMDB rating distribution over decades", ylab='imdb rating', col='green', border='yellow')`
4. `decade_levels <- sort(unique(imdb_decade$decade)) imdb_decade$decade <- factor(imdb_decade$decade, levels = decade_levels) boxplot( imdb_rating ~ decade, data = imdb_decade, main = "IMDb ratings by decade", xlab = "Decade", ylab = "IMDb rating", las = 2, col = "lightblue", border = "gray40" )`

Student responses

1. `ggplot(imdb_decade, aes(x = decade, y = imdb_rating)) + geom_boxplot(fill = "skyblue", color = "navy") + labs( title = "IMDb Ratings by Decade", x = "Decade", y = "IMDb rating", caption = "Source: IMDb" ) + theme_minimal() + theme( axis.text.x = element_text(angle = 45, hjust = 1) )`
2. Here is my boxplot.
3. I added titles and axes names
4. i changed the color
5. I changed the title. It kicks me out every time so I'm leaving this at that
6. I made it blue.
7. `imdb_decade$decade <- factor(imdb_decade$decade, levels = sort(unique(imdb_decade$decade))) boxplot(imdb_rating ~ decade, data = imdb_decade, main = "IMDb movie ratings by decade", xlab = "Decade", ylab = "IMDb rating", las = 2, col = "lightblue") mtext("Source: IMDb data", side = 1, line = 4, cex = 0.8)`
8. `imdb_decade$decade <- factor(imdb_decade$decade, levels = sort(unique(imdb_decade$decade)) ) boxplot(imdb_rating ~ decade, data = imdb_decade, main = "IMDb ratings by decade", xlab = "Decade", ylab = "IMDb rating", las = 2 ) mtext("Source: IMDb", side = 1, line = 4, cex = 0.8)`

Student responses

1. `imdb_decade$decade <- factor(imdb_decade$decade, levels = sort(unique(imdb_decade$decade))) boxplot(imdb_rating ~ decade, data = imdb_decade, main = "IMDb ratings by decade", xlab = "Decade", ylab = "IMDb rating", las = 2, col = "lightblue" )`
2. `imdb_decade$decade <- factor(imdb_decade$decade, levels = sort(unique(imdb_decade$decade))) boxplot(imdb_rating ~ decade, data = imdb_decade, main = "Distribution of IMDb ratings by decade", xlab = "Decade", ylab = "IMDb rating", las = 2, col = "lightblue") mtext("Source: IMDb data", side = 1, line = 4, cex = 0.8)`
3. `imdb_decade$decade <- factor(imdb_decade$decade, levels = sort(unique(imdb_decade$decade))) boxplot(imdb_rating ~ decade, data = imdb_decade, main = "IMDb ratings by decade", xlab = "Decade", ylab = "IMDb rating", las = 2, col = "lightblue") mtext("Source: IMDb data", side = 1, line = 4, cex = 0.8)`

Student responses

1. `par(mar = c(8, 4, 4, 2) + 0.1) boxplot(imdb_rating ~ decade, data = imdb_decade, main = "IMDb ratings by decade", xlab = "Decade", ylab = "IMDb rating", las = 2, col = "lightblue", cex.axis = 0.9) mtext("Source: IMDb dataset", side = 1, line = 6)`
2. titles, Dif intervals

- Group all movies in `imdb` by release `Year`. - Summarize into one row per year, with `avg\_rating` equal to the mean of `imdb\_rating`, giving each movie equal weight. - Include `n\_movies`, the number of movies in each year. - Save the result as `yearly\_ratings` and display the first few rows.

64 anonymous responses

Other responses

64

Student responses

1. `colnames(imdb)`
2. `imdb |> group_by(Year) |> head()`
3. `imdb |> group_by(Year) |> summarize( avg_rating = mean(imdb_rating), n_movies = n()) -> yearly_ratings head(yearly_ratings)`
4. `imdb |> group_by(Year) |> summarize(avg_rating = mean(imdb_rating), n_movies = n() <- yearly_ratings) head(yearly_ratings)`
5. `imdb |> group_by(Year) |> summarize(avg_rating = mean(imdb_rating), n_movies = n()) -> yearly_ratings head(yearly_ratings)`
6. `imdb |> group_by(Year) |> summarize(avg_rating = mean(imdb_rating), nmovies = n()) -> yearly_ratings head(yearly_ratings)`
7. `imdb |> group_by(Year) |> summarize(avg_rating = mean(imdb_rating), n_movies = n()) -> yearly_ratings head(yearly_ratings)`
8. `imdb |> group_by(Year)`
9. `imdb |> group_by(Year) |> summarize( avg_rating = mean(imdb_rating), n_movies = n() ) -> yearly_ratings head(yearly_ratings)`

Student responses

1. `imdb |> group_by(Year) |> summarize( avg_rating = mean(imdb_rating), n_movies = n() ) -> yearly_ratings`
2. `imdb |> group_by(Year) |> summarize(avg_rating = mean(imdb_rating), n_movies = n())`
3. `imdb |> group_by(Year) |> summarize(avg_rating = mean(imdb_rating), n_movies = n()) -> yearly_ratings head(yearly_ratings)`
4. `imdb |> group_by(Year) |> summarize(avg_rating = mean(imdb_rating), n_movies = n() -> yearly_ratings) |>`
5. `imdb |> group_by(Year) |> summarize(n_movies = n(), avg_rating = mean(imdb_rating))`
6. `imdb |> group_by_(Year)|> summarize(avg_rating = mean(imdb_rating),n_movies = n())->yearly_ratings`
7. `imdb |> group_by(Year)`
8. `imdb |> group_by(Year) summarize(avg_rating = mean(imdb_rating), n_movies = n()) -> yearly_ratings`
9. `imdb |> group_by(year) |> summarize(avg_rating = mean(imdb_rating), n_movies = n()) -> yearly_ratings head(yearly_ratings)`

Student responses

1. `imdb |> group_by(Year) |> summarize(avg_rating = mean(imdb_rating), n_movies = n()) -> yearly_ratings head(yearly_ratings)`
2. `imdb |> group_by(Year) |> summarize(avg_rating = mean(imdb_rating))`
3. `imdb |> group_by(Year) |> summarize(avg_rating = mean(imdb_rating) n_movies = n()) -> yearly_ratings head(yearly_ratings)`
4. `imdb |> group_by(Year) |> summarize(avg_rating = mean(imdb_rating), n_movies = n()) -> yearly_ratings head(yearly_ratings)`
5. `imdb |> group_by(Year) |> summarize(avg_rating = mean(imdb_rating), n_movies = n()) -> yearly_ratings head(yearly_ratings)`
6. `imdb |> group_by(Year) |> summarize(avg_rating = mean(imdb_rating), n_movies = n()) -> yearly_ratings head(yearly_ratings)`
7. `imdb |> group_by(Year) |> summarize(avg_rating = mean(imdb_rating), n_movies = n()) -> yearly_ratings yearly_ratings`
8. `imdb |> group_by(Year) |> summarize(avg_rating = mean(imdb_rating), n_movies = n()) -> yearly_ratings |> head(yearly_ratings)`

Student responses

1. `imdb |> group_by(Year) |> summarize(avg_rating = mean(imdb_rating), n_movies = n()) -> yearly_ratings`
2. `imdb |> group_by(Year) |> summarize(avg_rating = mean(imdb_rating), n_movies = n()) -> yearly_ratings`
3. `imdb |> group_by(Year) |> summarize(avg_rating = mean(imdb_rating), n_movies = n()) -> yearly_ratings`
4. `imdb |> group_by(Year) |> summarize(avg_rating = mean(imdb_rating), n_movies = n()) -> yearly_ratings head(yearly_ratings)`
5. `imdb |> group_by(Year) |> summarize(avg_rating = mean(imdb)rating), n_movies = n()) -> yearly_ratings head(yearly_ratings)`
6. `imdb |> group_by(Year) |> summarize(avg_rating=mean(imdb_rating), n_movies=n()) -> yearly_ratings head(yearly_ratings)`
7. `imdb |> group_by(Year) |> summarize(avg_rating=mean(imdb_rating), n_movies=n()) |> slice_head(n=5)`
8. `imdb |> head()`
9. `imdb |> head()`

Student responses

1. `imdb |> head() summarize(avg_rating = mean(imdb_rating), n_movie = n()) -> yearly_ratings head(yearly_ratings)`
2. `imdb |> head() temporal <- imdb |> group_by(Year) |> summarize( avg_rating = mean(imdb_rating), n_movies = sum(!is.na(Title)), .drop = TRUE ) |>`
3. `imdb |> head() yearly_ratings <- imdb |> group_by(Year) |> summarize( avg_rating = mean(imdb_rating), n_movies = n() ) yearly_ratings |> head()`
4. `imdb |> head() |> group_by(Year)|> summarize(avg_rating = mean(imdb_rating), n_movies(n)) -> yearly_ratings head(yearly_ratings)`
5. `imdb |>group_by(Year) |> summarize(avg_rating = mean(imdb_rating), n_movies = n()) -> yearly_ratings`
6. `imdb|> group_by(Year) |> summarize(avg_rating = mean(imdb_rating), n_movies = n()) -> yearly_ratings head(yearly_ratings)`
7. `yearly_rating <- imdb |> group_by(Year) |> summarize(avg_rating = mean(imdb_rating), n_movies = n())`
8. `yearly_ratings <- imdb %>% group_by(Year) %>% summarize( avg_rating = mean(imdb_rating, na.rm = TRUE), n_movies = n() ) yearly_ratings |> slice_head(n=5)`

Student responses

1. `yearly_ratings <- imdb |> group_by(Year) |> summarize( avg_rating = mean(imdb_rating, na.rm = TRUE), n_movies = n() )`
2. `yearly_ratings <- imdb |> group_by(Year) |> summarize( avg_rating = mean(imdb_rating, na.rm = TRUE), n_movies = n(), .groups = "drop" ) yearly_ratings |> head()`
3. `yearly_ratings <- imdb |> group_by(Year) |> summarize( avg_rating = mean(imdb_rating), n_movies = n() ) yearly_ratings |> head()`
4. `yearly_ratings <- imdb |> group_by(Year) |> summarize( avg_rating = mean(imdb_rating), n_movies = n() ) yearly_ratings |> head()`
5. `yearly_ratings <- imdb |> group_by(Year) |> summarize(avg_rating = mean(imdb_rating), n_movies = n()) |> slice_head(n = 5)`
6. `yearly_ratings <- imdb |> group_by(Year) |> summarize(avg_rating = mean(imdb_rating), n_movies = n()) head(yearly_ratings)`
7. `yearly_ratings <- imdb |> group_by(Year) |> summarize(aveg_rating = mean(imdb_rating), n_movies = n()) |> slice_head(n = 5)`

Student responses

1. `yearly_ratings <- imdb |> group_by(year) |> summarize(avg_rating = mean(imdb_rating), n_movies = n()) |> head(n=3)`
2. `yearly_ratings <- imdb |> group_by(Year) |> summarize(avg_rating = mean(imdb_rating), n_movies = n()) head(yearly_ratings)`
3. `yearly_ratings <- imdb |> group_by(Year) |> summarize(avg_rating = mean(imdb_rating), n_movies = n()) head(yearly_ratings, n = 5)`
4. `yearly_ratings <- imdb |> group_by(Year) |> summarize(avg_rating = mean(imdb_rating)), n_movies = n()) yearly_ratings |> head()`
5. `yearly_ratings <- imdb |> group_by(Year) |> summarize(avg_rating = mean(imdb_rating), n_movies = n()) head(yearly_ratings, n = 5)`
6. `yearly_ratings <- imdb |> group_by(Year) |> summarize(avg_rating = mean(imdb_rating), n_movies = n()) glimpse(yearly_ratings)`
7. `yearly_ratings <- imdb |> group_by(Year) |> summarize(avg_rating = mean(imdb_rating), n_movies = n()) head(yearly_ratings)`
8. `yearly_ratings <- imdb |> group_by(Year) |> summarize(avg_rating = mean(imdb_rating), n_movies = n()) yearly_ratings |> head()`

Student responses

1. `yearly_ratings <- imdb |> group_by(Year) |> summarize(avg_rating = mean(imdb_rating), nmovies=n()) head(yearly_ratings)`
2. `yearly_ratings <- imdb |> group_by(Year) |> summarize(mean_rating = mean(imdb_rating), n_movies = n()) yearly_ratings`
3. `yearly_ratings <- imdb |> group_by(Year) |> summarize(avg_rating = mean(imdb_rating), n_movies = n()) head(yearly_ratings)`
4. `yearly_ratings <- imdb |> group_by(Year) |> summarize(avg_rating = mean(imdb_rating), n_movies = n()) |> slice_head(n = 5)`
5. `yearly_ratings<-imdb |> group_by(Year)|> summarize(avg_rating = mean(imdb_rating), n_movies = n()) head(yearly_ratings)`
6. `yearly_ratings<-imdb |> group_by(Year)|> summarize(avg_rating=mean(imdb_rating), n_movies=n()) yearly_ratings|>head()`

This file includes only movies with at least 25,000 votes. Which older movies are likely to reach that threshold today? Which might be missing? How does this selection process affect the trend in average ratings over time?

80 anonymous responses

Selection favors famous older films

39

Students note that older movies reaching the vote threshold are typically well-known, popular, or classics, so the dataset includes higher-profile older titles.

Availability and dataset coverage issues

12

Students mention practical availability (streaming, theater presence, sequels) and dataset coverage affecting which older films reach vote thresholds and thus bias results.

Selection makes apparent rating trend changes

9

Students link the vote threshold to apparent changes in average ratings over time, e.g., newer films lowering means or older films appearing overly positive due to selection bias or growing movie counts.

Response themes for question 18

High-rated low-vote movies exist	6
A minority view noting some films may have very high (or low) average ratings despite not reaching the vote threshold, so notable titles could be excluded based on vote count alone.	
Only good older films get watched	5
Students suggest viewers primarily watch older films that are high quality, implying poorer older films lack votes and are excluded.	
Unclear or irrelevant response	4
Responses that are not informative or do not address the question meaningfully.	

Response themes for question 18

Older films lack early rating mechanisms

3

Some older movies may be missing simply because rating platforms or widespread rating behavior did not exist when they were released, so they never accumulated enough votes despite potential interest.

Other responses

2

Student responses

1. A lot of older movies will not have at least 25000 votes because a lot less people watched movies back then. It is plausible that only the super good movies got a lot of votes
2. Better older movies are more likely to reach that threshold, and worse ones are less likely. This skews it such that the mean of older movies is higher than it actually is
3. classics, missing might be forgotten older movies that wasn't as popular back then
4. creates selection bias the older movies are dis
5. Essentially, only "classic" older movies will be the ones with at least 25,000 votes. Today, many movies that are widely not considered good will garner that many votes; however, people nowadays who watch an older movie will almost certainly do so because that movie is a "classic" and therefore such movies will have a higher imdb_rating than the average movie released today.

Student responses

1. For older movies, it is less likely for them to reach over 25,000 votes, and those that do are likely more watched because they are considered better. Thus, older worse movies may exist but not be counted because they aren't even popular enough to meet the threshold, whereas all modern movies are spread through social media and can more easily be heavily voted on.
2. I think the Godfather would meet that threshold today for sure, this selection process affects the trend in average ratings over time because more recent movies have higher amounts of data on them, instead of older movies
3. I think the older movies that are more likely to reach this threshold today would be movies that are considered "classics". Movies that people didn't enjoy or know about would be missing. This selection process affects the trend in average ratings over time because before the internet existed, people wouldn't have been able to easily input their ratings, leading to less values.

Student responses

1. More earlier movies not that popular nowadays could be ignored even though they are much better in quality of the movie. This selection process would affect the trend negatively in average ratings over time as the movies which don't reach this threshold is ignored
2. Most votes may come from the time of the movie's release when it is the most popular, older movies don't have that. very popular famous old movies have lots of votes
3. Movies that are considered a "classic" would be more visible on imdb, and then continue to get more ratings because of it. It is a bias that makes them more present.
4. Movies that are old and classical enough may be able to reach the threshold. Movies that are too old that few people watched today or that are too new that few have watched it may be missing.
5. Obsession is likely to reach that threshold, and the 25,000 vote selection process might skew rating higher toward better movies that have been watched more.

Student responses

1. Older movies have more time to reach 25,000 votes, while some newer movies may not have reached that threshold yet, even if they're really good movies, which brings down the mean rating.
2. Older movies in this dataset may be selectively popular or well-known films because only movies with at least 25,000 votes are included.
3. Older movies like wizard of oz would likely make the cut, and other good movies. This would lower average
4. Older movies that are popular or considered classics are more likely to have at least 25,000 votes, while less popular older movies may be missing. This creates selection bias, making older movies appear to have higher average ratings than newer movies because only the more popular older films are included.
5. Older movies that are still popular or well-known are more likely to have 25,000 votes. Less popular older movies may be missing. This creates selection bias, so the average ratings for older movies may not represent all movies from that time.

Student responses

1. Older movies that are still popular today will have higher votes, which lead to biased result, considering they will get higher ratings.
2. Older movies that have that many ratings are much more accessible and likely are more highly regarded than less accessible and worse films. This would give them a higher average rating than newer films, which are highly accessible regardless of quality
3. Older movies that were popular and well regarded are more likely to reach that threshold today. The older bad movies are likely missing which could give a false downward trend.
4. Older movies with at least 25,000 votes are likely to be the more famous or popular ones, while many less-known older movies are missing. This creates selection bias and may make average ratings for older movies look artificially high compared with newer movies.
5. older popular movies are more likely to reach that threshold, because if they are not good, less people are inclined to watch those films. Bad movies that people did not like, will not show up here because people realized they were bad and stopped watching them

Student responses

1. Only older, really popular movies that are highly regarded would show up and bad movies might be missing whereas in modern times more movies are rated by 25,000 votes so that could explain the lower ratings trend as we move closer to modern times.
2. Only popular movies likely met the 25,000 vote threshold. This leaves less mainstream movies to be voted on. Since popular movies are the ones being voted on, those years likely have an inflated rating overall, compared to all movies from today
3. only very good old movies are likely to have this many votes, which could lead older data to be skewed towards higher ratings
4. only very popular movies from the 1900s are going to reach that many, so they are going to have higher reviews. There will be lots of missing data from movies with 24,000, etc. This will show averages going down as more and more people vote and there are more movies to be judged about nowadays. All the data we collect from the past will probably be overly positive.

Student responses

1. Popular mvvoies will more likely to pass this threshold since they will have more votes than the unpopular ones. Since popular movies are more likely to be a better movie and have better rating, it will overestimate the actual rating of the movies.
2. revenges
3. Some old well-known movies like Citizen Kane could not have received enough votes. This hurts our findings when looking over time
4. The 'classics' which tend to be critically acclaimed, thus bringing up their averages. Less-watched older films will be ignored.
5. The 25,00 vote cutoff only keeps old movies that are still famous today
6. The Godfather or any older popular classics, makes the trend more noticable for today's movies
7. the most popular movies are likely to reach that threshold today, especially ones people still rewatch today. The movies that were popular with the older generation but not anymore will be missing, and even though they were well liked, they won't be represented.

Student responses

1. The movies that are likely to reach the 25,000 votes that are older are likely to be movies that people consider classics or are on a movie list for being very well done. This is different because we do not actually see a random amount of people just rating movies from all time but people deliberately seeking out good ones
2. The older movies to hit that many votes are probalby the more highly rates ones, which skews the adta to say that the older movies are generally higher rated
3. The only older movies that are likely to reach 25k votes are those that are spread through word of mouth as being classics, which are probably going to be better than the median movie today. It's survivorship bias, only the best movies showing up in the data, making it look like recent movies just aren't as good.
4. The only older movies that are receiving lots of ratings today are probably one that were really popular and really good so the average ratings for those movies are skewed

Student responses

1. We are likely missing movies that may have been popular at their time of release, but are not as popular now. Older movies must have retained popularity for many years to be included.

Student responses

1. As the years go on, the more movies are released so maybe people have more movies to compare others to. Older movies could be missing which would make the average rating higher
2. Movies in the middle of 1925 to 1950 are likely to reach the mean rating that we see now. There is a relationship between accuracy and the number of data points available
3. Movies that are not easily accessible online like via netflix
4. Movies that are popular, the ones that didnt make it to theaters or a large stremaign sevrice, it doesnt give accurate reuslts
5. Movies with higher ratings, lower rating movies and various reasons
6. New movie may not have enough votes since they are newer. This will reduce the voting number by the time.
7. newer movies might be missing because they haven't reached the threshold yet and older movies are more likely to make the threshold because of their longevity, so this might mean that there is a larger sample size for older movies
8. Not really a movie person. But the selection process would let the averge rating going down since the movie are better now

Student responses

1. older movies are harder to reach that threshold. this will make the ones that do make it a higher weight on them which will skew the ratings trends
2. older movies will not have that many votes
3. Older movies that are popular are likely to reach that threshold, however, movies that haven't reached the box-office or aren't available on media could be missed, considering how some movies are stored in older methods. I don't think this would severely impact the trend in average ratings over time though considering it is already decreasing.
4. The older movies that have a sequel are likely to reach that, older movies when technology wasn't really accessible could be ignored.

Student responses

1. 25,000 is more extreme amount of votes compared to today. Therefore in the past, a movie with over 25,000 most likely could have been an exceptional movie to create that many. This distorts the average because the movies with less than 25k are thrown out
2. As the older population decreases over time, some classic movies won't be displayed because they don't meet the 25k threshold. Some of the old movies that are likely to reach the threshold include beauty and the beast, romeo and juliet, etc.
3. Movies are becoming more common, allowing lower-quality movies to be made. Older movies of poor qualities might also be ignored entirely.
4. Movies with higher ratings peobably have the 25000 ratings, and the ones missing are probably the unpopular ones because people might not want to write reviews on negative movies because not many people know aobut them. This might reflect why there is a downward trend?
5. People are more online now and movies trend differently then they used to
6. some movies that might be missing are really old movies

Student responses

1. The average will be affected because the list will be growing. It will become more of a normal distribution
2. There may be new considerations for the imdb rating than what was considered before. With more variables causing the rating to change, it would cause average ratings to decline over time.
3. well-known older movies may still continue to attract audience but less known old movies may be forgot, so there might be biases in the selection of movies

Student responses

1. Movies are more popular now than back then, so people watching now are more diverse, with a larger set of opinions and are less likely to collectively agree on rating a single movie as really high due to needing to cater to so many people. Meanwhile, I would assume older movies were much more niche to a specific film-watching community who were incredibly passionate, so the movies that do exist are rated higher simply due to their passion
2. Movies that didn't have 25k in the year they were released?
3. movies with great advertising. indie movies are missing. This makes better movies
4. Older movies likely to reach that threshold today would be western films, as global media tends to have a bias of attention toward the western world. Films from countries with more developing film industries might be missing. More global exposure means more votes for films not from the west, and the audience may have a bias. More variety as well.

Student responses

1. The movies with less votes are missing, which could change the result as less votes movies may be better. And movies get more and more votes nowadays, so there's some bias in the results
2. There could be some movies that don't have as many votes but still have a very high average rating, or vice versa. Some movies weren't as popular and didn't get as many votes but were still very good movies with good ratings. Old movies didn't get as many votes since those aren't common today

Student responses

1. older movies that people had stronger feelings about are more likely to receive enough votes to make the threshold of 25000 votes, as opposed to forgettable, middle of the pack movies that wouldn't, this would make older movies trend upwards in rating
2. Only the older movies that are considered to be good are likely to have 25,000 votes today, because people have heard about their greatness. Thus older movies that weren't good are not included because they do not have enough ratings. Today there are more people to watch and rate movies on these websites, so they are more likely to be rated accordingly.
3. people are only going to watch the old films that are good
4. people are unlikely to go back and watch old movies that are bad or average, so old movies that do get ratings are more likely to get positive ratings, so it seems like ratings are on a downward trend
5. The older movies likely to reach 25,000 reviews are likely to be very good, as people would have to go out of their way to watch them. This drives the average ratings for older movies up, even if that isn't actually the case.

Student responses

1. 1
2. I feel like something like nostalgia could play a larger factor in how people look back at these movies and rank them. Also, there are so many more movies out nowadays in my opinion so that could be the difference. Also I bet much older movies have less ratings unless they are good movies.
3. movies before 1970 like the Japanese movie seven warriors
4. Older movies

Student responses

1. All the famous older movies that were popular before rating movies on the internet became popular would change the ratings over time
2. Older movies that aren't already super popular might not have a lot of ratings because rating technology didn't exist when these movies were released.
3. Probably very few movies from back then actually reached that threshold, so they were not accounted for in this experiment. Therefore, the mean/summaries were skewed heavily without our necessary data.