431 Classes 09 and 10

https://thomaselove.github.io/431-2024/

2024-09-24

## This Week’s Agenda

* Managing the Favorite Movies Data
  + Ingesting data from a Google Sheet
  + Checking the variables
  + Some semi-sophisticated cleaning: imdb\_categories
  + What makes a “good” research question?
* Review of Independent Samples Comparisons (including ANOVA)
* Simple Linear Regression Models
  + Fitting an OLS model
  + Performance of an OLS model
  + Checking the Fit
  + Transformations
  + What makes a “good” fit?

## Load packages and set theme

library(ggrepel) ## new: for building plots with text  
library(glue) ## new: for combining strings  
library(googlesheets4) ## new: importing Google Sheets data  
library(naniar) ## new: counting missingness  
library(knitr)  
library(kableExtra) ## for neatening tables in slides  
library(janitor)  
library(car) ## for boxCox function  
library(infer) ## bootstrapping  
library(patchwork)  
library(rstanarm)  
library(easystats)  
library(tidyverse)  
  
theme\_set(theme\_bw())  
knitr::opts\_chunk$set(comment = NA)  
  
source("c09/data/Love-431.R") # for the lovedist() function

## Importing Data on Favorite Movies

gs4\_deauth() # indicates to Google Drive that you're reading a public file  
  
url <- "https://docs.google.com/spreadsheets/d/155iHDSUr8ZixX4nVcNq9HMkKbBizUhCsXHcteIdNddU"  
  
mov1 <- read\_sheet(url)

✔ Reading from "movies\_2024-09-17".

✔ Range '2024-09-17 Data'.

dim(mov1)

[1] 228 18

names(mov1)

[1] "mov\_id" "movie" "year" "length"   
 [5] "imdb\_categories" "imdb\_ratings" "imdb\_stars" "mpa"   
 [9] "dr\_love" "mentions" "list\_20" "list\_21"   
[13] "list\_22" "list\_23" "list\_24" "imdb\_synopsis"   
[17] "imdb\_link" "imdb\_cat1"

## Any missing values?

n\_miss(mov1)

[1] 0

miss\_var\_summary(mov1)

# A tibble: 18 × 3  
 variable n\_miss pct\_miss  
 <chr> <int> <num>  
 1 mov\_id 0 0  
 2 movie 0 0  
 3 year 0 0  
 4 length 0 0  
 5 imdb\_categories 0 0  
 6 imdb\_ratings 0 0  
 7 imdb\_stars 0 0  
 8 mpa 0 0  
 9 dr\_love 0 0  
10 mentions 0 0  
11 list\_20 0 0  
12 list\_21 0 0  
13 list\_22 0 0  
14 list\_23 0 0  
15 list\_24 0 0  
16 imdb\_synopsis 0 0  
17 imdb\_link 0 0  
18 imdb\_cat1 0 0

## Looking at just our key variables

Restricting the data to our six key variables (plus the identifiers mov\_id and movie):

mov2 <- mov1 |> select(mov\_id, imdb\_stars, imdb\_ratings, length,  
 mpa, year, imdb\_categories, movie)  
glimpse(mov2)

Rows: 228  
Columns: 8  
$ mov\_id <chr> "M-001", "M-002", "M-003", "M-004", "M-005", "M-006", …  
$ imdb\_stars <dbl> 8.4, 8.0, 7.3, 8.3, 7.9, 7.8, 8.5, 8.4, 7.9, 8.4, 8.4,…  
$ imdb\_ratings <dbl> 442000, 127000, 397000, 730000, 58000, 396000, 978000,…  
$ length <dbl> 170, 138, 97, 149, 119, 123, 117, 160, 162, 149, 181, …  
$ mpa <chr> "PG-13", "NR", "PG-13", "G", "TV-PG", "R", "R", "PG", …  
$ year <dbl> 2009, 1963, 1999, 1968, 2009, 2013, 1979, 1984, 2009, …  
$ imdb\_categories <chr> "Comedy, Drama", "Epic, Psychological Drama, Showbiz D…  
$ movie <chr> "3 Idiots", "8 1/2", "10 Things I Hate About You", "20…

## Assessing variables by type

* Identification variables: mov\_id and movie.
  + Are these *distinct* (do we have a different value of these in every row of our data?

nrow(mov2); n\_distinct(mov2$mov\_id); n\_distinct(mov2$movie)

[1] 228

[1] 228

[1] 228

OK.

## Assessing variables by type

* Quantities (check ranges)

mov2 |> reframe(range(imdb\_stars), range(imdb\_ratings),   
 range(length), range(year))

# A tibble: 2 × 4  
 `range(imdb\_stars)` `range(imdb\_ratings)` `range(length)` `range(year)`  
 <dbl> <dbl> <dbl> <dbl>  
1 3.4 282 70 1942  
2 9.3 2900000 207 2024

* imdb\_stars = weighted average movie score (1 - 10)
* imdb\_ratings = # of users who’ve rated movie
* length = length of movie, in minutes
* year = year movie was released

## Assessing variables by type

* Categorical (assess levels, create factors)

mov2 |> count(mpa)

# A tibble: 9 × 2  
 mpa n  
 <chr> <int>  
1 G 7  
2 NR 13  
3 PG 62  
4 PG-13 74  
5 R 67  
6 TV-14 1  
7 TV-G 2  
8 TV-MA 1  
9 TV-PG 1

## Dealing with mpa

* Let’s create a four-level factor, as follows:

mov2 <- mov2 |>  
 mutate(mpa = fct\_lump\_n(mpa, n = 3, other\_level = "Other"))  
  
mov2 |> count(mpa)

# A tibble: 4 × 2  
 mpa n  
 <fct> <int>  
1 PG 62  
2 PG-13 74  
3 R 67  
4 Other 25

* Here, fct\_lump\_n() with n = 3 collapses all mpa values that occur less often than the top 3 mpa values.

## Final Key Variable: imdb\_categories

mov2 |> count(imdb\_categories) |> arrange(desc(n))

# A tibble: 215 × 2  
 imdb\_categories n  
 <chr> <int>  
 1 Comedy, Drama 3  
 2 Drama 3  
 3 Adventure Epic, Desert Adventure, Globetrotting Adventure, Quest, Acti… 2  
 4 Comedy 2  
 5 Comedy, Drama, Music 2  
 6 Comedy, Drama, Romance 2  
 7 Drama, Romance 2  
 8 Feel-Good Romance, Comedy, Drama, Romance 2  
 9 Feel-Good Romance, Romantic Comedy, Comedy, Drama, Romance 2  
10 Period Drama, Drama, Romance 2  
# ℹ 205 more rows

* In 228 films, we see 215 different combinations of genres in imdb\_categories.

## Separate imdb\_categories

mov2 <- mov2 |>  
 separate\_wider\_delim(imdb\_categories, delim = ",", cols\_remove = FALSE,   
 too\_few = "align\_start",  
 names = c("genre01", "genre02", "genre03", "genre04",  
 "genre05", "genre06", "genre07", "genre08",  
 "genre09", "genre10"))  
  
names(mov2)

[1] "mov\_id" "imdb\_stars" "imdb\_ratings" "length"   
 [5] "mpa" "year" "genre01" "genre02"   
 [9] "genre03" "genre04" "genre05" "genre06"   
[13] "genre07" "genre08" "genre09" "genre10"   
[17] "imdb\_categories" "movie"

## Missing genre values?

miss\_var\_summary(mov2)

# A tibble: 18 × 3  
 variable n\_miss pct\_miss  
 <chr> <int> <num>  
 1 genre10 211 92.5   
 2 genre09 195 85.5   
 3 genre08 176 77.2   
 4 genre07 151 66.2   
 5 genre06 116 50.9   
 6 genre05 80 35.1   
 7 genre04 44 19.3   
 8 genre03 15 6.58  
 9 genre02 5 2.19  
10 mov\_id 0 0   
11 imdb\_stars 0 0   
12 imdb\_ratings 0 0   
13 length 0 0   
14 mpa 0 0   
15 year 0 0   
16 genre01 0 0   
17 imdb\_categories 0 0   
18 movie 0 0

## Is this a Superhero movie?

* Create a variable called superhero which is 1 if the movie’s imdb\_categories list includes Superhero and 0 otherwise.

mov2 <- mov2 |>  
 mutate(superhero = as.numeric(  
 str\_detect(imdb\_categories, fixed("Superhero"))))  
  
mov2 |> count(superhero)

# A tibble: 2 × 2  
 superhero n  
 <dbl> <int>  
1 0 215  
2 1 13

## Our 13 “Superhero” Movies

mov2 |> select(superhero, movie) |> filter(superhero == 1)

# A tibble: 13 × 2  
 superhero movie   
 <dbl> <chr>   
 1 1 Avengers: Infinity War   
 2 1 Avengers: Endgame   
 3 1 Big Hero 6   
 4 1 Black Panther   
 5 1 Captain Marvel   
 6 1 The Dark Knight   
 7 1 The Dark Knight Rises   
 8 1 Doctor Strange   
 9 1 Iron Man   
10 1 Mystery Men   
11 1 Spider Man: Into The Spider-Verse  
12 1 Thor: Love and Thunder   
13 1 V for Vendetta

## Indicators of 12 Most Common Genres

mov2 <- mov2 |>   
 mutate(action = as.numeric(str\_detect(imdb\_categories, fixed("Action"))),  
 adventure = as.numeric(str\_detect(imdb\_categories, fixed("Adventure"))),  
 animation = as.numeric(str\_detect(imdb\_categories, fixed("Animation"))),  
 comedy = as.numeric(str\_detect(imdb\_categories, fixed("Comedy"))),  
 crime = as.numeric(str\_detect(imdb\_categories, fixed("Crime"))),  
 drama = as.numeric(str\_detect(imdb\_categories, fixed("Drama"))),  
 family = as.numeric(str\_detect(imdb\_categories, fixed("Family"))),  
 fantasy = as.numeric(str\_detect(imdb\_categories, fixed("Fantasy"))),  
 mystery = as.numeric(str\_detect(imdb\_categories, fixed("Mystery"))),  
 romance = as.numeric(str\_detect(imdb\_categories, fixed("Romance"))),  
 scifi = as.numeric(str\_detect(imdb\_categories, fixed("Sci-Fi"))),  
 thriller = as.numeric(str\_detect(imdb\_categories, fixed("Thriller")))  
 )

## Genre Counts (across our 228 movies)

* For the 12 most common genres…

mov2 |>   
 select(action:thriller) |>  
 colSums()

action adventure animation comedy crime drama family fantasy   
 60 80 25 97 29 132 36 53   
 mystery romance scifi thriller   
 22 55 44 46

### Build a tibble of genre counts

genre\_counts <- mov2 |>   
 select(action:thriller) |>  
 colSums() |>   
 t() |> as\_tibble() |> pivot\_longer(action:thriller) |>  
 rename(genre = name, movies = value) |>   
 arrange(desc(movies))

## The genre\_counts tibble

genre\_counts

# A tibble: 12 × 2  
 genre movies  
 <chr> <dbl>  
 1 drama 132  
 2 comedy 97  
 3 adventure 80  
 4 action 60  
 5 romance 55  
 6 fantasy 53  
 7 thriller 46  
 8 scifi 44  
 9 family 36  
10 crime 29  
11 animation 25  
12 mystery 22

## How many movies are

* both Romance and Comedy?

mov2 |> count(comedy, romance)

# A tibble: 4 × 3  
 comedy romance n  
 <dbl> <dbl> <int>  
1 0 0 110  
2 0 1 21  
3 1 0 63  
4 1 1 34

## 34 Romantic Comedies?

mov2 |> filter(comedy == 1, romance == 1) |>  
 select(movie) |> slice(1:18)

# A tibble: 18 × 1  
 movie   
 <chr>   
 1 10 Things I Hate About You   
 2 About Time   
 3 Clueless   
 4 Coming To America   
 5 Crazy Rich Asians   
 6 Elf   
 7 Flipped   
 8 Harold and Maude   
 9 High School Musical 2   
10 Jab We Met (When We Met)   
11 La La Land   
12 Legally Blonde   
13 Life As We Know It   
14 Life Is Beautiful   
15 The Lobster   
16 Mamma Mia!   
17 Mamma Mia: Here We Go Again  
18 Monte Carlo

mov2 |> filter(comedy == 1, romance == 1) |>  
 select(movie) |> slice(19:34)

# A tibble: 16 × 1  
 movie   
 <chr>   
 1 Murder Mystery   
 2 My Big Fat Greek Wedding   
 3 Notting Hill   
 4 Om Shanti Om (Peace Be With You)   
 5 The Parent Trap   
 6 Real Genius   
 7 Scott Pilgrim vs. The World   
 8 The Secret Life of Walter Mitty   
 9 Shrek   
10 Shrek 2   
11 Tangled   
12 Thor: Love and Thunder   
13 Trolls   
14 When Harry Met Sally   
15 Yeh Jawaani hai Deewani (Youth is Crazy)   
16 Zindagi Na Milegi Dobara (You Only Live Once)

## Any romance + comedy + superhero?

mov2 |> filter(romance == 1, comedy == 1, superhero == 1) |>  
 select(mov\_id, movie, imdb\_categories)

# A tibble: 1 × 3  
 mov\_id movie imdb\_categories   
 <chr> <chr> <chr>   
1 M-209 Thor: Love and Thunder Superhero, Action, Adventure, Comedy, Fantasy, …

mov2 |> filter(mov\_id == "M-209") |> select(imdb\_categories, mov\_id)

# A tibble: 1 × 2  
 imdb\_categories mov\_id  
 <chr> <chr>   
1 Superhero, Action, Adventure, Comedy, Fantasy, Romance, Sci-Fi M-209

|  |
| --- |
| Note |
| Thor: Love and Thunder does not have the “Romantic Comedy” genre. Only nine of our movies do (About Time, Clueless, Coming to America, Crazy Rich Asians, Harold and Maude, Legally Blonde, Mamma Mia!, My Big Fat Greek Wedding and Notting Hill.) |

## Superhero, Sci-Fi & IMDB Stars (1/4)

mov2 |> group\_by(superhero) |> reframe(lovedist(imdb\_stars))

# A tibble: 2 × 11  
 superhero n miss mean sd med mad min q25 q75 max  
 <dbl> <int> <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
1 0 215 0 7.54 0.872 7.7 0.741 3.4 7.1 8.1 9.3  
2 1 13 0 7.72 0.898 7.9 0.741 6.1 7.3 8.4 9

mov2 |> group\_by(scifi) |> reframe(lovedist(imdb\_stars))

# A tibble: 2 × 11  
 scifi n miss mean sd med mad min q25 q75 max  
 <dbl> <int> <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
1 0 184 0 7.52 0.902 7.7 0.741 3.4 7.1 8.1 9.3  
2 1 44 0 7.63 0.734 7.7 0.890 6.1 7.1 8.3 8.8

* But what if a movie is in both genres?

## Superhero, Sci-Fi & IMDB Stars (2/4)

What to do about the movies with “Superhero” *and* “Sci-Fi”?

mov2 |> filter(superhero == 1, scifi == 1) |>  
 select(mov\_id, movie, superhero, scifi, imdb\_stars)

# A tibble: 10 × 5  
 mov\_id movie superhero scifi imdb\_stars  
 <chr> <chr> <dbl> <dbl> <dbl>  
 1 M-010 Avengers: Infinity War 1 1 8.4  
 2 M-011 Avengers: Endgame 1 1 8.4  
 3 M-019 Big Hero 6 1 1 7.8  
 4 M-021 Black Panther 1 1 7.3  
 5 M-029 Captain Marvel 1 1 6.8  
 6 M-049 Doctor Strange 1 1 7.5  
 7 M-105 Iron Man 1 1 7.9  
 8 M-151 Mystery Men 1 1 6.1  
 9 M-209 Thor: Love and Thunder 1 1 6.2  
10 M-217 V for Vendetta 1 1 8.1

## Superhero, Sci-Fi & IMDB Stars (3/4)

mov2 |> group\_by(superhero, scifi) |>   
 reframe(lovedist(imdb\_stars))

# A tibble: 4 × 12  
 superhero scifi n miss mean sd med mad min q25 q75 max  
 <dbl> <dbl> <int> <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
1 0 0 181 0 7.51 0.898 7.7 0.741 3.4 7.1 8.1 9.3  
2 0 1 34 0 7.69 0.705 7.7 0.890 6.5 7.12 8.3 8.8  
3 1 0 3 0 8.6 0.346 8.4 0 8.4 8.4 8.7 9   
4 1 1 10 0 7.45 0.842 7.65 0.890 6.1 6.92 8.05 8.4

* What groups might we want to compare here?

## Superhero, Sci-Fi & IMDB Stars (4/4)

mov2 |> filter(superhero == 1, scifi == 0) |>  
 select(movie)

# A tibble: 3 × 1  
 movie   
 <chr>   
1 The Dark Knight   
2 The Dark Knight Rises   
3 Spider Man: Into The Spider-Verse

mov2 |> filter(superhero == 0, scifi == 1) |>  
 select(movie) |> slice(1:14)

# A tibble: 14 × 1  
 movie   
 <chr>   
 1 2001: A Space Odyssey   
 2 About Time   
 3 Alien   
 4 Avatar   
 5 Back to the Future   
 6 Back to the Future Part II   
 7 Blade Runner 2049   
 8 Cloud Atlas   
 9 Despicable Me   
10 Divergent   
11 Eternal Sunshine of the Spotless Mind  
12 Everything, Everywhere, All at Once   
13 Face/Off   
14 Gattaca

mov2 |> filter(superhero == 0, scifi == 1) |>  
 select(movie) |> slice(15:34)

# A tibble: 20 × 1  
 movie   
 <chr>   
 1 Gravity   
 2 The Hunger Games   
 3 The Hunger Games: Catching Fire   
 4 Inception   
 5 Interstellar   
 6 Jurassic Park   
 7 The Lobster   
 8 The Matrix   
 9 Minions: The Rise of Gru   
10 The Platform (El Hoyo)   
11 Real Genius   
12 Real Steel   
13 Resident Evil   
14 Rise of the Guardians   
15 Sorry To Bother You   
16 Star Wars Episode III: Revenge of the Sith   
17 Star Wars: Episode IV: A New Hope   
18 Star Wars: Episode V: The Empire Strikes Back  
19 Star Wars: Episode VI: Return of the Jedi   
20 War of the Worlds

* Are these groups comparable?

## Movie Questions for Today

1. Do movies released in 1942-2010 have more user ratings than movies released after 2010? (imdb\_ratings, year)
2. How do movie lengths vary by MPA ratings? (length, mpa)
3. How strong is the association between how often a movie is rated on IMDB and its number of stars? (imdb\_ratings, imdb\_stars)

## Question 1

Do movies released in 1942-2010 have more user ratings than movies released after 2010? (imdb\_ratings, year)

### Numerical Summaries

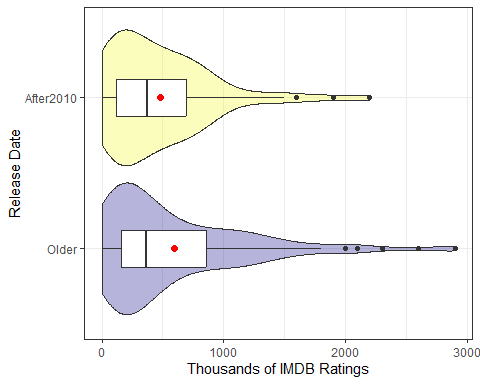
mov2 <- mov2 |>  
 mutate(release = fct\_recode(factor(year > 2010),  
 After2010 = "TRUE", Older = "FALSE"))  
  
mov2 |> group\_by(release) |> reframe(lovedist(imdb\_ratings))

# A tibble: 2 × 11  
 release n miss mean sd med mad min q25 q75 max  
 <fct> <int> <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
1 Older 152 0 592217. 604231. 367000 4.11e5 4800 161750 860750 2.90e6  
2 After2010 76 0 482916. 442536. 369500 3.87e5 282 122000 689500 2.20e6

## Plotting the Two Samples

Let’s plot # of ratings by thousands, to avoid some scientific notation.

ggplot(mov2, aes(x = imdb\_ratings/1000, y = release)) +  
 geom\_violin(aes(fill = release)) +  
 geom\_boxplot(width = 0.25) +  
 stat\_summary(fun = mean, geom = "point", col = "red",   
 shape = 16, size = 2) +  
 scale\_fill\_viridis\_d(option = "C", alpha = 0.3) +  
 guides(fill = "none") +  
 labs(y = "Release Date", x = "Thousands of IMDB Ratings")



## Data not close to Normal

Each sample shows right skew in the # of IMDB user ratings:

| Group | Sample Mean | Sample Median |
| --- | --- | --- |
| Older | 592,217.1 | 367,000 |
| After2010 | 482,915.6 | 369,500 |
| Difference | 109,301.5 | -2,500 |

* Could we use a bootstrap to compare the means without worrying much about the skew (instead compare medians?)
* Could we use a non-linear transformation?
* Let’s use an 89% uncertainty interval. (Why not?)

## Bootstrap difference in means

# point estimate   
mov2 |> specify(imdb\_ratings ~ release) |>  
 calculate(stat = "diff in means", order = c("Older", "After2010"))

Response: imdb\_ratings (numeric)  
Explanatory: release (factor)  
# A tibble: 1 × 1  
 stat  
 <dbl>  
1 109302.

# 89% confidence interval  
set.seed(202409241)  
mov2 |> specify(imdb\_ratings ~ release) |>  
 generate(reps = 2500, type = "bootstrap") |>  
 calculate(stat = "diff in means", order = c("Older", "After2010")) |>  
 get\_ci(level = 0.89, type = "percentile")

# A tibble: 1 × 2  
 lower\_ci upper\_ci  
 <dbl> <dbl>  
1 -7165. 219881.

## Bootstrap difference in medians

# point estimate   
mov2 |> specify(imdb\_ratings ~ release) |>  
 calculate(stat = "diff in medians", order = c("Older", "After2010"))

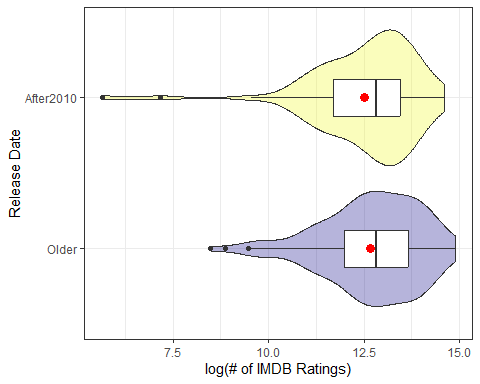
Response: imdb\_ratings (numeric)  
Explanatory: release (factor)  
# A tibble: 1 × 1  
 stat  
 <dbl>  
1 -2500

# 89% confidence interval  
set.seed(202409242)  
mov2 |> specify(imdb\_ratings ~ release) |>  
 generate(reps = 2500, type = "bootstrap") |>  
 calculate(stat = "diff in medians", order = c("Older", "After2010")) |>  
 get\_ci(level = 0.89, type = "percentile")

# A tibble: 1 × 2  
 lower\_ci upper\_ci  
 <dbl> <dbl>  
1 -124943. 122500

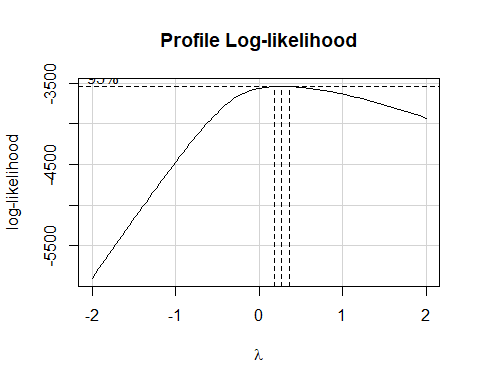
## Right Skew: Try a logarithm?

ggplot(mov2, aes(x = log(imdb\_ratings), y = release)) +  
 geom\_violin(aes(fill = release)) +  
 geom\_boxplot(width = 0.25) +  
 stat\_summary(fun = mean, geom = "point", col = "red",   
 shape = 16, size = 3) +  
 scale\_fill\_viridis\_d(option = "C", alpha = 0.3) +  
 guides(fill = "none") +  
 labs(y = "Release Date", x = "log(# of IMDB Ratings)")



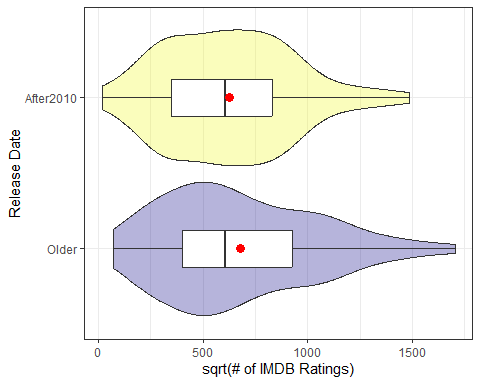
## Box-Cox suggestion?

fit0 <- lm(imdb\_ratings ~ release, data = mov2)  
boxCox(fit0)



## Right Skew: Try a square root?

ggplot(mov2, aes(x = sqrt(imdb\_ratings), y = release)) +  
 geom\_violin(aes(fill = release)) +  
 geom\_boxplot(width = 0.25) +  
 stat\_summary(fun = mean, geom = "point", col = "red",   
 shape = 16, size = 3) +  
 scale\_fill\_viridis\_d(option = "C", alpha = 0.3) +  
 guides(fill = "none") +  
 labs(y = "Release Date", x = "sqrt(# of IMDB Ratings)")



## OLS model

fit1 <- lm(sqrt(imdb\_ratings) ~ release, data = mov2)  
  
model\_parameters(fit1, ci = 0.89)

Parameter | Coefficient | SE | 89% CI | t(226) | p  
-------------------------------------------------------------------------------  
(Intercept) | 675.78 | 28.45 | [ 630.14, 721.43] | 23.76 | < .001  
release [After2010] | -52.72 | 49.27 | [-131.78, 26.33] | -1.07 | 0.286

Uncertainty intervals (equal-tailed) and p-values (two-tailed) computed  
 using a Wald t-distribution approximation.

estimate\_contrasts(fit1, contrast = "release", ci = 0.89)

Marginal Contrasts Analysis  
  
Level1 | Level2 | Difference | 89% CI | SE | t(226) | p  
---------------------------------------------------------------------------  
Older | After2010 | 52.72 | [-26.33, 131.78] | 49.27 | 1.07 | 0.286  
  
Marginal contrasts estimated at release  
p-value adjustment method: Holm (1979)

* Of course, these results are on the square root scale.

## Bayesian model

set.seed(202409213)  
fit2 <- stan\_glm(sqrt(imdb\_ratings) ~ release, data = mov2, refresh = 0)  
model\_parameters(fit2, ci = 0.89)

Parameter | Median | 89% CI | pd | Rhat | ESS | Prior  
----------------------------------------------------------------------------------------------------  
(Intercept) | 676.41 | [ 628.71, 722.50] | 100% | 1.000 | 3717.00 | Normal (658.21 +- 877.10)  
releaseAfter2010 | -54.06 | [-131.15, 25.31] | 85.60% | 1.001 | 3658.00 | Normal (0.00 +- 1856.51)

Uncertainty intervals (equal-tailed) and p-values (two-tailed) computed  
 using a MCMC distribution approximation.

estimate\_contrasts(fit2, contrast = "release", ci = 0.89)

Marginal Contrasts Analysis  
  
Level1 | Level2 | Difference | 89% CI | pd | % in ROPE  
-----------------------------------------------------------------------  
Older | After2010 | 54.06 | [-25.31, 131.15] | 85.60% | 0.08%  
  
Marginal contrasts estimated at release

## Making Predictions (1/2)

* Use our OLS model for sqrt(imdb\_ratings) to make predictions on the square root scale.

estimate\_means(fit1, ci = 0.89, by = "release", transform = "none")

Estimated Marginal Means  
  
release | Mean | SE | 89% CI  
---------------------------------------------  
Older | 675.78 | 28.45 | [630.14, 721.43]  
After2010 | 623.06 | 40.23 | [558.51, 687.61]  
  
Marginal means estimated at release

Note that and

## Making Predictions (2/2)

* Use our model for sqrt(imdb\_ratings) to make predictions on the original scale of imdb\_ratings.

estimate\_means(fit1, ci = 0.89, by = "release",   
 transform = "response")

Estimated Marginal Means  
  
release | Mean | SE | 89% CI  
------------------------------------------------------  
Older | 4.57e+05 | 38448.77 | [3.97e+05, 5.20e+05]  
After2010 | 3.88e+05 | 50132.42 | [3.12e+05, 4.73e+05]  
  
Marginal means estimated at release

## Summary for Question 1

Do movies released in 1942-2010 have more user ratings than movies released after 2010?

| Group | Sample Mean | Sample Median |
| --- | --- | --- |
| Older | 592,217.1 | 367,000 |
| After2010 | 482,915.6 | 369,500 |
| Difference | 109,301.5 | -2,500 |

* Bootstrap means: diff = 109302, 89% CI (-7165, 219881)
* Bootstrap medians: diff = -2500, 89% CI (-124943, 122500)

## Question 1 Models

Do movies released in 1942-2010 have more user ratings than movies released after 2010?

* **OLS**: Square root of user ratings for a movie released in 1942-2010 is, on average, 52.72 (89% CI: -26, 132) higher than for a movie released after 2010.
* **Bayes**: Square root of user ratings for a movie released in 1942-2010 is, on average, 54.06 (89% CI: -25, 131) higher than for a movie released after 2010.

## Question 2

How do movie lengths vary by MPA ratings? (length, mpa)

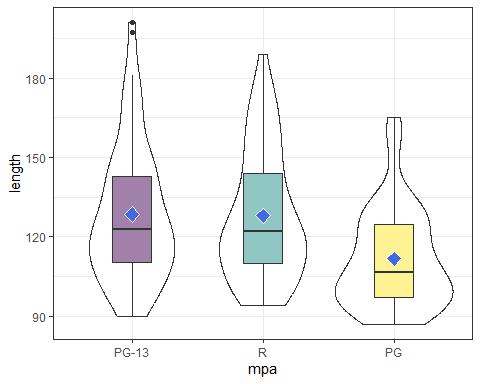
### We’ll focus on PG, PG-13 and R

mov3 <- mov2 |>  
 filter(mpa != "Other") |>  
 mutate(mpa = fct\_reorder(mpa, length, .desc = TRUE))  
  
mov3 |> group\_by(mpa) |> reframe(lovedist(length))

# A tibble: 3 × 11  
 mpa n miss mean sd med mad min q25 q75 max  
 <fct> <int> <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
1 PG-13 74 0 128. 25.5 123 24.5 90 110. 143 201  
2 R 67 0 128. 24.4 122 23.7 94 110 144 189  
3 PG 62 0 112. 18.8 106. 17.8 87 97 125. 165

## Three Independent Samples

ggplot(mov3, aes(x = mpa, y = length)) +  
 geom\_violin() +   
 geom\_boxplot(aes(fill = mpa), width = 0.3) +  
 stat\_summary(fun = mean, geom = "point", size = 4,   
 shape = 23, col = "snow", fill = "royalblue") +  
 scale\_fill\_viridis\_d(option = "D", alpha = 0.5) +  
 guides(fill = "none")



## OLS model

fit3 <- lm(length ~ mpa, data = mov3)  
  
anova(fit3)

Analysis of Variance Table  
  
Response: length  
 Df Sum Sq Mean Sq F value Pr(>F)   
mpa 2 11714 5857.1 10.842 3.386e-05 \*\*\*  
Residuals 200 108045 540.2   
---  
Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

estimate\_means(fit3, ci = 0.89)

We selected `by = c("mpa")`.

Estimated Marginal Means  
  
mpa | Mean | SE | 89% CI  
----------------------------------------  
PG-13 | 128.45 | 2.70 | [124.11, 132.78]  
R | 127.96 | 2.84 | [123.40, 132.51]  
PG | 111.73 | 2.95 | [106.99, 116.46]  
  
Marginal means estimated at mpa

## Pairwise Comparisons

estimate\_contrasts(fit3, contrast = "mpa", ci = 0.89, p\_adjust = "Holm")

Marginal Contrasts Analysis  
  
Level1 | Level2 | Difference | 89% CI | SE | t(200) | p  
-----------------------------------------------------------------------  
(PG-13) | PG | 16.72 | [ 8.30, 25.14] | 4.00 | 4.18 | < .001  
(PG-13) | R | 0.49 | [-7.75, 8.74] | 3.92 | 0.13 | 0.900   
R | PG | 16.23 | [ 7.61, 24.85] | 4.10 | 3.96 | < .001  
  
Marginal contrasts estimated at mpa  
p-value adjustment method: Holm (1979)

## Build a tibble of contrasts

con\_holm <- estimate\_contrasts(fit3, ci = 0.89, p\_adjust = "holm")

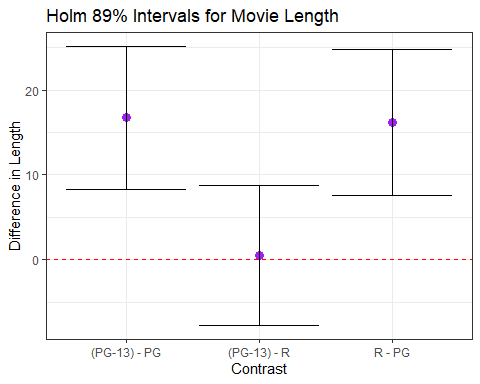
No variable was specified for contrast estimation. Selecting `contrast =  
 "  
 mpa  
 "`.

con\_holm\_tib <- tibble(con\_holm) |>   
 mutate(contr = str\_c(Level1, " - ", Level2))  
  
con\_holm\_tib

# A tibble: 3 × 10  
 Level1 Level2 Difference CI\_low CI\_high SE df t p contr   
 <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <chr>   
1 (PG-13) PG 16.7 8.30 25.1 4.00 200 4.18 0.000132 (PG-13) -…  
2 (PG-13) R 0.491 -7.75 8.74 3.92 200 0.125 0.900 (PG-13) -…  
3 R PG 16.2 7.61 24.8 4.10 200 3.96 0.000206 R - PG

## Plot Holm comparisons

ggplot(con\_holm\_tib, aes(x = contr, y = Difference)) +  
 geom\_point(size = 3, col = "purple") +  
 geom\_errorbar(aes(ymin = CI\_low, ymax = CI\_high)) +  
 geom\_hline(yintercept = 0, col = "red", lty = "dashed") +  
 labs(title = "Holm 89% Intervals for Movie Length",  
 x = "Contrast",   
 y = "Difference in Length")



## Summary for Question 2

How do movie lengths vary by MPA ratings?

| MPA |  | Sample Mean | Sample Median |
| --- | --- | --- | --- |
| PG-13 | 74 | 128.4 | 123 |
| R | 67 | 128.0 | 122 |
| PG | 62 | 111.7 | 106.5 |

* In the sample, PG movies are shorter by 16-17 minutes.
* Pairwise 89% contrasts yield larger differences between PG and the other mpa groups, than between PG-13 and R.

# Simple Linear Regression

## Question 3

How strong is the association between how often a movie is rated on IMDB and its number of stars? (imdb\_ratings, imdb\_stars)

mov2 |> reframe(lovedist(imdb\_ratings))

# A tibble: 1 × 10  
 n miss mean sd med mad min q25 q75 max  
 <int> <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
1 228 0 555783. 556984. 367000 391406. 282 158000 783750 2900000

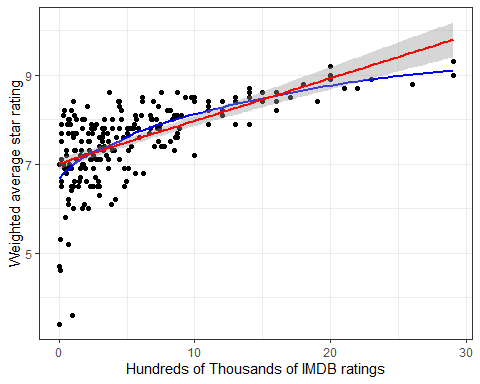
mov2 |> reframe(lovedist(imdb\_stars))

# A tibble: 1 × 10  
 n miss mean sd med mad min q25 q75 max  
 <int> <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
1 228 0 7.55 0.872 7.7 0.741 3.4 7.1 8.1 9.3

## Scatterplot of 228 movies

We’ll look at stars (on the y axis) vs. ratings (in 100,000s, on x).

ggplot(mov2, aes(x = imdb\_ratings/100000, y = imdb\_stars)) +  
 geom\_point() +  
 geom\_smooth(method = "loess",   
 formula = y ~ x, se = FALSE, col = "blue") +  
 geom\_smooth(method = "lm",   
 formula = y ~ x, se = TRUE, col = "red") +  
 labs(x = "Hundreds of Thousands of IMDB ratings",  
 y = "Weighted average star rating")



## Pearson Correlation

cor(mov2$imdb\_stars, mov2$imdb\_ratings)

[1] 0.6131559

cor(mov2$imdb\_stars, (mov2$imdb\_ratings/100000))

[1] 0.6131559

* We can add or multiply by a constant without changing the Pearson correlation coefficient.

cor\_test(mov2, "imdb\_stars", "imdb\_ratings")

Parameter1 | Parameter2 | r | 95% CI | t(226) | p  
--------------------------------------------------------------------  
imdb\_stars | imdb\_ratings | 0.61 | [0.53, 0.69] | 11.67 | < .001\*\*\*  
  
Observations: 228

## OLS model with imdb\_ratings

fit5 <- lm(imdb\_stars ~ imdb\_ratings, data = mov2)  
  
model\_parameters(fit5, ci = 0.89)

Parameter | Coefficient | SE | 89% CI | t(226) | p  
----------------------------------------------------------------------  
(Intercept) | 7.01 | 0.06 | [6.91, 7.12] | 108.44 | < .001  
imdb ratings | 9.60e-07 | 8.23e-08 | [0.00, 0.00] | 11.67 | < .001

Uncertainty intervals (equal-tailed) and p-values (two-tailed) computed  
 using a Wald t-distribution approximation.

Very hard to conceptualize in any practical context, plus the 89% CI for the slope of imdb\_ratings is 0.

* What if we rescaled the imdb\_ratings maintaining the linear relationship?
  + We can add or multiply by any constant we like.
  + Divide # of ratings by 100,000?

## OLS with rescaled imdb\_ratings

mov4 <- mov2 |>  
 mutate(users\_100k = imdb\_ratings/100000)  
  
fit6 <- lm(imdb\_stars ~ users\_100k, data = mov4)  
  
model\_parameters(fit6, ci = 0.89)

Parameter | Coefficient | SE | 89% CI | t(226) | p  
---------------------------------------------------------------------  
(Intercept) | 7.01 | 0.06 | [6.91, 7.12] | 108.44 | < .001  
users 100k | 0.10 | 8.23e-03 | [0.08, 0.11] | 11.67 | < .001

Uncertainty intervals (equal-tailed) and p-values (two-tailed) computed  
 using a Wald t-distribution approximation.

* When comparing any two movies whose # of IMDB user ratings are 100,000 apart, we see a star rating that is 0.10 stars (89% CI 0.08, 0.11) higher, on average, for the movie with more user ratings, according to this model.

## Performance of the OLS model

model\_performance(fit6)

# Indices of model performance  
  
AIC | AICc | BIC | R2 | R2 (adj.) | RMSE | Sigma  
---------------------------------------------------------------  
482.090 | 482.197 | 492.378 | 0.376 | 0.373 | 0.687 | 0.690

* Key summaries for an OLS model with one predictor, like fit6, are and Sigma (which is similar to RMSE.)
* tells us model fit6 accounts for 37.6% of the variation in imdb\_stars that we observe in our mov2 data.

## Performance of the OLS model

model\_performance(fit6)

# Indices of model performance  
  
AIC | AICc | BIC | R2 | R2 (adj.) | RMSE | Sigma  
---------------------------------------------------------------  
482.090 | 482.197 | 492.378 | 0.376 | 0.373 | 0.687 | 0.690

* Our model fit6 assumes that our errors (residuals) come from a Normal distribution with mean 0 and standard deviation Sigma () = 0.69.
* Thus, about 68% of our predictions should be within 0.69 stars of the correct outcome, and 95% of our predictions should be within , or 1.38 stars.

# Planned Break between Classes 09 and 10.

## Performance Measures (1/5)

* AIC: Akaike’s Information Criterion
* AICc: Second-order (or small sample) AIC with a correction for small sample sizes
* BIC: Bayesian Information Criterion

AIC, AICc and BIC are used when comparing one or more models for the same outcome. When comparing models fit using maximum likelihood (like OLS linear models), the smaller the AIC or BIC, the better the fit.

## Performance Measures (2/5)

R2: r-squared value = 0.376

* The R-squared () measure for an OLS fit describes how much of the variation in our outcome can be explained using our model (and its predictors.) falls between 0 and 1, and the closer it is to 1, the better the model fits our data.
* In a simple (one-predictor) OLS model like this, the value is also the square of the Pearson correlation coefficient, .
* We called this “eta-squared” () in ANOVA.

## Performance Measures (3/5)

R2 (adj.): adjusted r-squared value = 0.373

* Adjusted R-squared is an index (so it’s not a proportion of anything) for comparing different models (different predictor sets) for the same outcome.
* The idea is to reduce the temptation to overfit the data, by penalizing the value a little for each predictor.
* Adjusted is usually between 0 and 1, but can be negative.
* Its formula accounts for the number of observations and the number of predictors in the model.
* The adjusted measure can never be larger than .

## Performance Measures (4/5)

RMSE = 0.687

* The RMSE is the square root of the variance of the residuals and summarizes the difference between the observed data and the model’s predicted values.
* It can be interpreted as the standard deviation of the unexplained variance, and has the same units as the outcome.
* When comparing models using the same data for the same outcome (but, for instance, with different predictor sets), lower RMSE values indicate better model fit.

## Performance Measures (5/5)

Sigma = 0.690

* Linear models assume that their residuals are drawn from a Normal distribution with mean 0 and standard deviation equal to sigma ().
* This indicates that the predicted outcome will be within units of the observed outcome for approximately 68% of the data points, for example.

## Checking OLS Model Fit

Main assumptions of any simple linear regression are:

1. **linearity**: we assume that the outcome is linearly related to our predictor
2. **constant variance** (homoscedasticity): we assume that the variation of our outcome is about the same regardless of the value of our predictor
3. **normal distribution**: we assume that the errors around the regression model at any specified values of the x-variables follow an approximately Normal distribution.

To check these assumptions, consider the following plots.

## Fitting the diagnostic plots

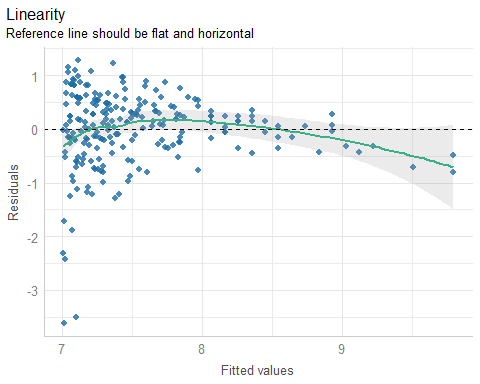
fit6\_diagnostic\_plots <-   
 plot(check\_model(fit6, panel = FALSE))

For confidence bands, please install `qqplotr`.

* I don’t worry about confidence bands in these plots.

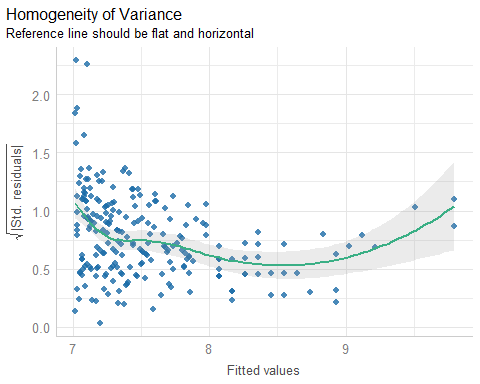
## Checking Linearity

fit6\_diagnostic\_plots[[2]]



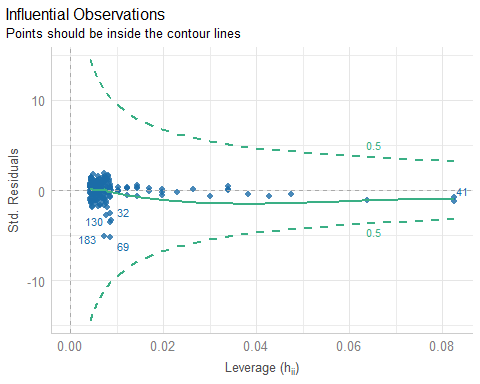
## Checking Equal Variances

fit6\_diagnostic\_plots[[3]]



## Checking for Influential Points

fit6\_diagnostic\_plots[[4]]



## Which points are listed in the plot?

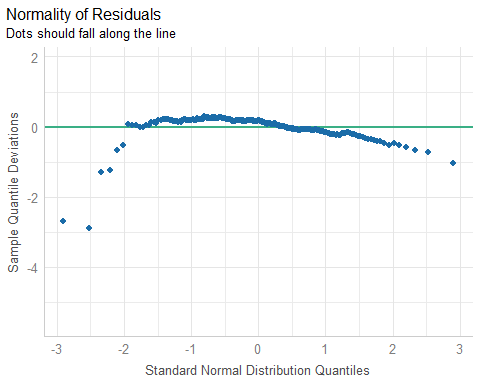
None are anywhere near the contours, as it turns out.

mov2 |> slice(c(41, 32, 69, 130, 183)) |> select(mov\_id, movie)

# A tibble: 5 × 2  
 mov\_id movie   
 <chr> <chr>   
1 M-041 The Dark Knight   
2 M-032 Chinese Doctors (Zhong guo yi sheng)  
3 M-069 The Gingerdead Man   
4 M-130 Madea Goes To Jail   
5 M-183 The Room

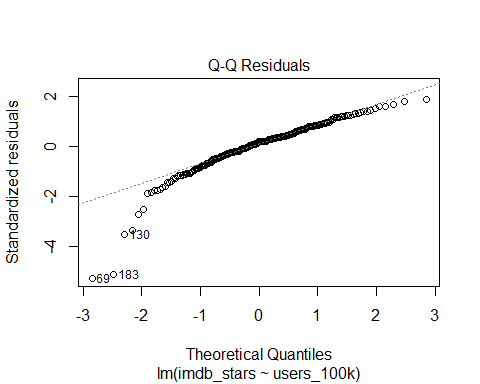
## Checking Normality

fit6\_diagnostic\_plots[[5]]



## Alternative: Normal Q-Q

plot(fit6, which = 2)



## Which points are low outliers?

mov2 |> slice(c(69, 183, 130)) |> select(mov\_id, movie, imdb\_stars)

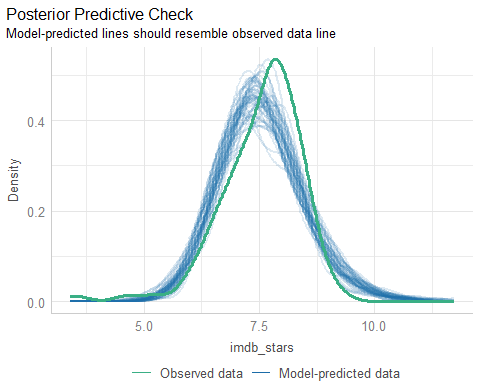
# A tibble: 3 × 3  
 mov\_id movie imdb\_stars  
 <chr> <chr> <dbl>  
1 M-069 The Gingerdead Man 3.4  
2 M-183 The Room 3.6  
3 M-130 Madea Goes To Jail 4.6

mov2 |> select(mov\_id, movie, imdb\_stars) |>   
 arrange(imdb\_stars) |> head(3)

# A tibble: 3 × 3  
 mov\_id movie imdb\_stars  
 <chr> <chr> <dbl>  
1 M-069 The Gingerdead Man 3.4  
2 M-183 The Room 3.6  
3 M-130 Madea Goes To Jail 4.6

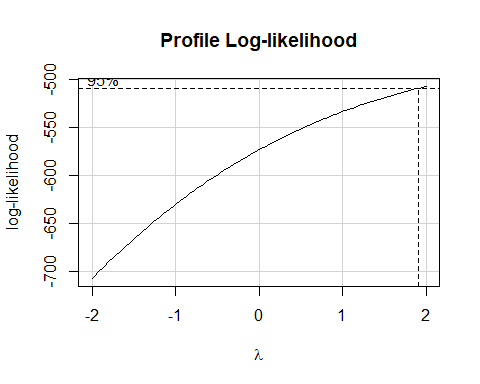
## Posterior Predictive Checks

fit6\_diagnostic\_plots[[1]]



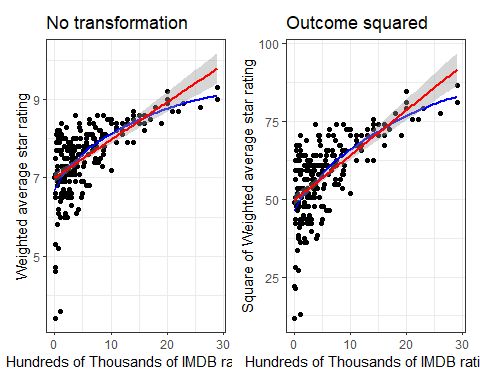
## Box-Cox suggestion?

boxCox(fit6)



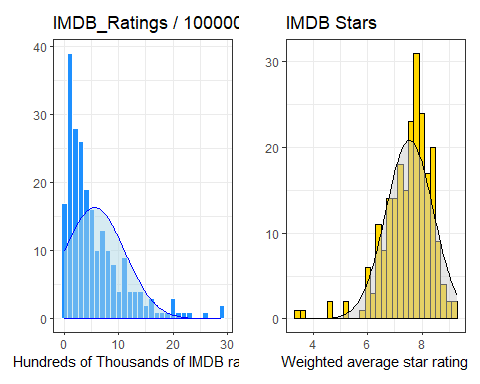
## imdb\_stars squared?

p1 <- ggplot(mov2, aes(x = imdb\_ratings/100000, y = imdb\_stars)) +  
 geom\_point() +  
 geom\_smooth(method = "loess",   
 formula = y ~ x, se = FALSE, col = "blue") +  
 geom\_smooth(method = "lm",   
 formula = y ~ x, se = TRUE, col = "red") +  
 labs(x = "Hundreds of Thousands of IMDB ratings",  
 y = "Weighted average star rating",  
 title = "No transformation")  
  
p2 <- ggplot(mov2, aes(x = imdb\_ratings/100000, y = imdb\_stars^2)) +  
 geom\_point() +  
 geom\_smooth(method = "loess",   
 formula = y ~ x, se = FALSE, col = "blue") +  
 geom\_smooth(method = "lm",   
 formula = y ~ x, se = TRUE, col = "red") +  
 labs(x = "Hundreds of Thousands of IMDB ratings",  
 y = "Square of Weighted average star rating",  
 title = "Outcome squared")  
  
p1 + p2



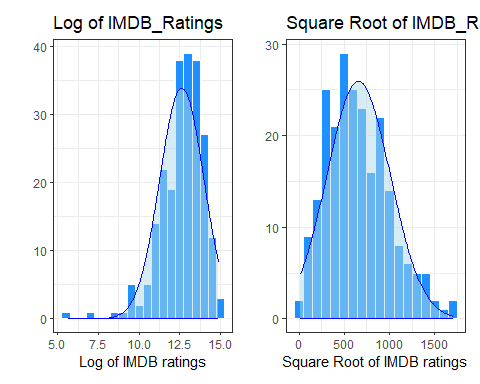
## Could we transform either variable?

p3 <- ggplot(mov2, aes(x = imdb\_ratings/100000)) +  
 geom\_histogram(binwidth = 1, col = "white", fill = "dodgerblue") +  
 stat\_function(fun = function(x)  
 dnorm(x, mean = mean(mov2$imdb\_ratings/100000,   
 na.rm = TRUE),  
 sd = sd(mov2$imdb\_ratings/100000,   
 na.rm = TRUE)) \*  
 length(mov2$imdb\_ratings/100000) \* 1,   
 geom = "area", alpha = 0.5,  
 fill = "lightblue", col = "blue") +  
 labs(x = "Hundreds of Thousands of IMDB ratings", y = "",  
 title = "IMDB\_Ratings / 100000")  
  
p4 <- ggplot(mov2, aes(x = imdb\_stars)) +  
 geom\_histogram(binwidth = 0.2, col = "black", fill = "gold") +  
 stat\_function(fun = function(x)  
 dnorm(x, mean = mean(mov2$imdb\_stars,  
 na.rm = TRUE),  
 sd = sd(mov2$imdb\_stars,   
 na.rm = TRUE)) \*  
 length(mov2$imdb\_stars) \* 0.2,   
 geom = "area", alpha = 0.5,  
 fill = "grey80", col = "black") +  
 labs(x = "Weighted average star rating", y = "",  
 title = "IMDB Stars")  
  
p3 + p4



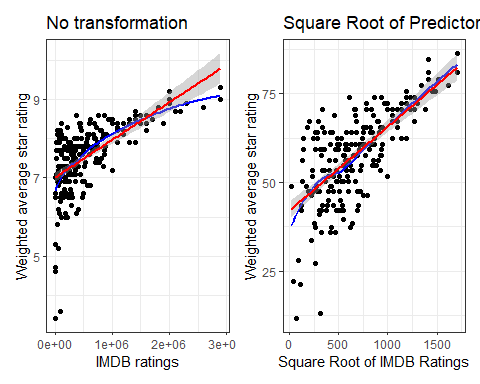
## Transforming IMDB\_ratings?

p3a <- ggplot(mov2, aes(x = log(imdb\_ratings))) +  
 geom\_histogram(binwidth = 0.5, col = "white", fill = "dodgerblue") +  
 stat\_function(fun = function(x)  
 dnorm(x, mean = mean(log(mov2$imdb\_ratings),   
 na.rm = TRUE),  
 sd = sd(log(mov2$imdb\_ratings),   
 na.rm = TRUE)) \*  
 length(log(mov2$imdb\_ratings)) \* 0.5,   
 geom = "area", alpha = 0.5,  
 fill = "lightblue", col = "blue") +  
 labs(x = "Log of IMDB ratings", y = "",   
 title = "Log of IMDB\_Ratings")  
  
p3b <- ggplot(mov2, aes(x = sqrt(imdb\_ratings))) +  
 geom\_histogram(binwidth = 100, col = "white", fill = "dodgerblue") +  
 stat\_function(fun = function(x)  
 dnorm(x, mean = mean(sqrt(mov2$imdb\_ratings),   
 na.rm = TRUE),  
 sd = sd(sqrt(mov2$imdb\_ratings),   
 na.rm = TRUE)) \*  
 length(sqrt(mov2$imdb\_ratings)) \* 100,   
 geom = "area", alpha = 0.5,  
 fill = "lightblue", col = "blue") +  
 labs(x = "Square Root of IMDB ratings", y = "",   
 title = "Square Root of IMDB\_Ratings")  
  
p3a + p3b



## New Scatterplot?

p5 <- ggplot(mov2, aes(x = imdb\_ratings, y = imdb\_stars)) +  
 geom\_point() +  
 geom\_smooth(method = "loess",   
 formula = y ~ x, se = FALSE, col = "blue") +  
 geom\_smooth(method = "lm",   
 formula = y ~ x, se = TRUE, col = "red") +  
 labs(x = "IMDB ratings",  
 y = "Weighted average star rating",  
 title = "No transformation")  
  
p6 <- ggplot(mov2, aes(x = sqrt(imdb\_ratings), y = imdb\_stars^2)) +  
 geom\_point() +  
 geom\_smooth(method = "loess",   
 formula = y ~ x, se = FALSE, col = "blue") +  
 geom\_smooth(method = "lm",   
 formula = y ~ x, se = TRUE, col = "red") +  
 labs(x = "Square Root of IMDB Ratings",  
 y = "Weighted average star rating",  
 title = "Square Root of Predictor")  
  
p5 + p6



## Model with

fit7 <- lm(imdb\_stars ~ sqrt(imdb\_ratings), data = mov2)  
  
model\_parameters(fit7, ci = 0.89)

Parameter | Coefficient | SE | 89% CI | t(226) | p  
-----------------------------------------------------------------------------  
(Intercept) | 6.49 | 0.09 | [6.34, 6.64] | 68.94 | < .001  
imdb ratings [sqrt] | 1.60e-03 | 1.26e-04 | [0.00, 0.00] | 12.71 | < .001

Uncertainty intervals (equal-tailed) and p-values (two-tailed) computed  
 using a Wald t-distribution approximation.

* Same problem as before.
* Tiny slope coefficients are needlessly hard to interpret.

## Rescaled Model

mov4 <- mov4 |>  
 mutate(sqrtratK = sqrt(imdb\_ratings)/1000)  
  
fit8 <- lm(imdb\_stars ~ sqrtratK, data = mov4)  
model\_parameters(fit8, ci = 0.89)

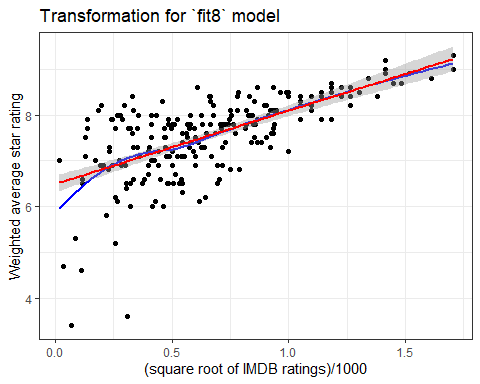
Parameter | Coefficient | SE | 89% CI | t(226) | p  
-----------------------------------------------------------------  
(Intercept) | 6.49 | 0.09 | [6.34, 6.64] | 68.94 | < .001  
sqrtratK | 1.60 | 0.13 | [1.40, 1.81] | 12.71 | < .001

Uncertainty intervals (equal-tailed) and p-values (two-tailed) computed  
 using a Wald t-distribution approximation.

* Suppose we have two movies whose square root of # of IMDB user ratings is 1000 apart. On average, the star rating is 1.60 stars (89% CI 1.40, 1.81) higher for the movie with more IMDB user ratings, according to model fit8.

## Scatterplot for model fit8

ggplot(mov4, aes(x = sqrtratK, y = imdb\_stars)) +  
 geom\_point() +  
 geom\_smooth(method = "loess",   
 formula = y ~ x, se = FALSE, col = "blue") +  
 geom\_smooth(method = "lm",   
 formula = y ~ x, se = TRUE, col = "red") +  
 labs(x = "(square root of IMDB ratings)/1000",  
 y = "Weighted average star rating",  
 title = "Transformation for `fit8` model")



## Model fit8 vs. fit6 performance

fit6 <- lm(imdb\_stars ~ users\_100k, data = mov4)  
fit8 <- lm(imdb\_stars ~ sqrtratK, data = mov4)  
model\_performance(fit6)

# Indices of model performance  
  
AIC | AICc | BIC | R2 | R2 (adj.) | RMSE | Sigma  
---------------------------------------------------------------  
482.090 | 482.197 | 492.378 | 0.376 | 0.373 | 0.687 | 0.690

model\_performance(fit8)

# Indices of model performance  
  
AIC | AICc | BIC | R2 | R2 (adj.) | RMSE | Sigma  
---------------------------------------------------------------  
466.628 | 466.735 | 476.916 | 0.417 | 0.414 | 0.664 | 0.667

* Which model looks better here?
* fit8, with lower AIC, BIC, RMSE, Sigma, and higher .

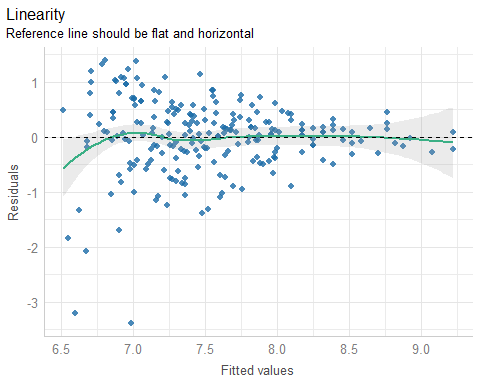
## Model Checking for fit8

fit8\_diagnostic\_plots <-   
 plot(check\_model(fit8, panel = FALSE))

For confidence bands, please install `qqplotr`.

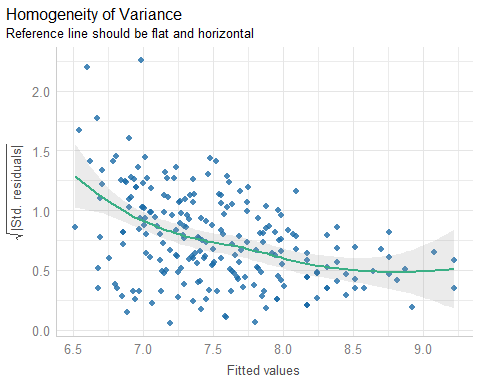
## Checking Linearity

fit8\_diagnostic\_plots[[2]]



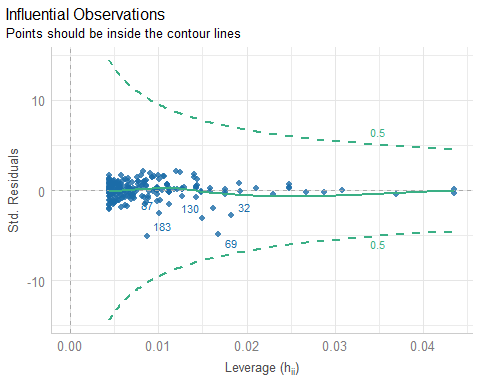
## Checking Equal Variances

fit8\_diagnostic\_plots[[3]]



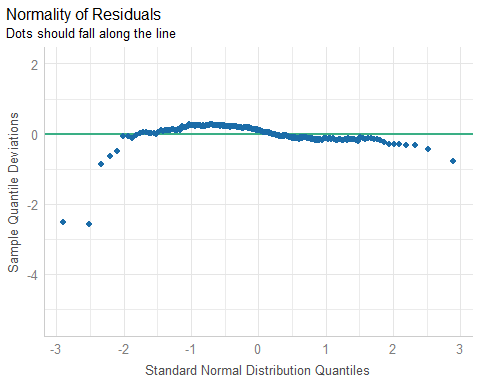
## Checking for Influential Points

fit8\_diagnostic\_plots[[4]]



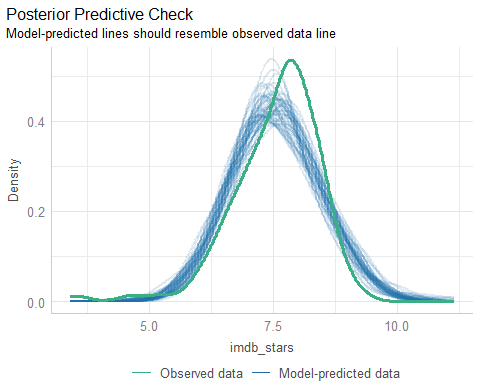
## Checking Normality

fit8\_diagnostic\_plots[[5]]



## Posterior Predictive Checks

fit8\_diagnostic\_plots[[1]]



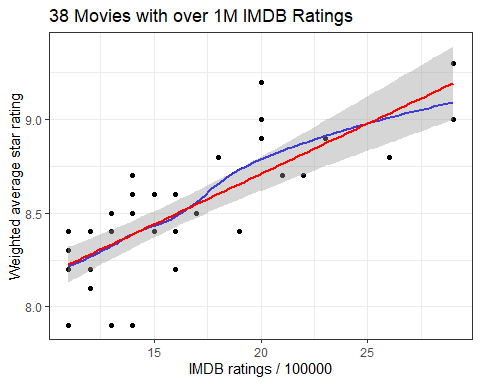
## Restricting the Sample???

What if instead of doing a transformation, we only looked at the subset of movies with over 1,000,000 IMDB user ratings?

mov5 <- mov4 |> filter(imdb\_ratings > 1000000)  
dim(mov5)

[1] 38 34

ggplot(mov5, aes(x = imdb\_ratings/100000, y = imdb\_stars)) +  
 geom\_point() +  
 geom\_smooth(method = "loess",   
 formula = y ~ x, se = FALSE, col = "blue") +  
 geom\_smooth(method = "lm",   
 formula = y ~ x, se = TRUE, col = "red") +  
 labs(x = "IMDB ratings / 100000",  
 y = "Weighted average star rating",  
 title = glue(nrow(mov5), " Movies with over 1M IMDB Ratings"))



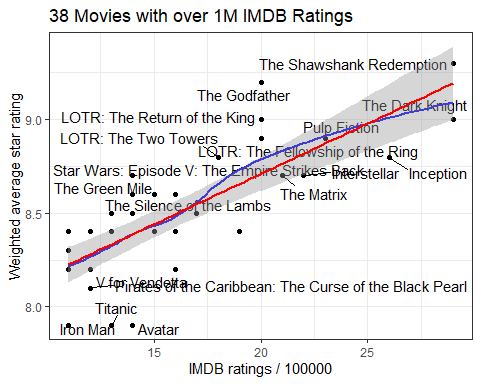
* Is this a good idea?

## Labeling the Movies in the Scatterplot

ggplot(mov5, aes(x = imdb\_ratings/100000, y = imdb\_stars,   
 label = movie)) +  
 geom\_point() +  
 geom\_text\_repel() +  
 geom\_smooth(method = "loess",   
 formula = y ~ x, se = FALSE, col = "blue") +  
 geom\_smooth(method = "lm",   
 formula = y ~ x, se = TRUE, col = "red") +  
 labs(x = "IMDB ratings / 100000",  
 y = "Weighted average star rating",  
 title = glue(nrow(mov5), " Movies with over 1M IMDB Ratings"))

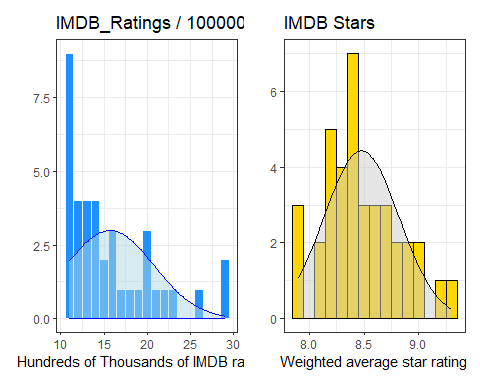
Warning: The following aesthetics were dropped during statistical transformation: label.  
ℹ This can happen when ggplot fails to infer the correct grouping structure in  
 the data.  
ℹ Did you forget to specify a `group` aesthetic or to convert a numerical  
 variable into a factor?  
The following aesthetics were dropped during statistical transformation: label.  
ℹ This can happen when ggplot fails to infer the correct grouping structure in  
 the data.  
ℹ Did you forget to specify a `group` aesthetic or to convert a numerical  
 variable into a factor?

Warning: ggrepel: 20 unlabeled data points (too many overlaps). Consider  
increasing max.overlaps



## Our subset of 38 movies

p5a <- ggplot(mov5, aes(x = imdb\_ratings/100000)) +  
 geom\_histogram(binwidth = 1, col = "white", fill = "dodgerblue") +  
 stat\_function(fun = function(x)  
 dnorm(x, mean = mean(mov5$imdb\_ratings/100000,   
 na.rm = TRUE),  
 sd = sd(mov5$imdb\_ratings/100000,   
 na.rm = TRUE)) \*  
 length(mov5$imdb\_ratings/100000) \* 1,   
 geom = "area", alpha = 0.5,  
 fill = "lightblue", col = "blue") +  
 labs(x = "Hundreds of Thousands of IMDB ratings", y = "",  
 title = "IMDB\_Ratings / 100000")  
  
p5b <- ggplot(mov5, aes(x = imdb\_stars)) +  
 geom\_histogram(binwidth = 0.1, col = "black", fill = "gold") +  
 stat\_function(fun = function(x)  
 dnorm(x, mean = mean(mov5$imdb\_stars,  
 na.rm = TRUE),  
 sd = sd(mov5$imdb\_stars,   
 na.rm = TRUE)) \*  
 length(mov5$imdb\_stars) \* 0.1,   
 geom = "area", alpha = 0.5,  
 fill = "grey80", col = "black") +  
 labs(x = "Weighted average star rating", y = "",  
 title = "IMDB Stars")  
  
p5a + p5b



## Model fit9 for 38 movies

fit9 <- lm(imdb\_stars ~ users\_100k, data = mov5)  
  
model\_parameters(fit9, ci = 0.89)

Parameter | Coefficient | SE | 89% CI | t(36) | p  
--------------------------------------------------------------------  
(Intercept) | 7.63 | 0.11 | [7.45, 7.82] | 67.96 | < .001  
users 100k | 0.05 | 6.81e-03 | [0.04, 0.07] | 7.91 | < .001

Uncertainty intervals (equal-tailed) and p-values (two-tailed) computed  
 using a Wald t-distribution approximation.

* If two movies each have over 1 million IMDB ratings, and the movies have a 100,000 user difference in IMDB ratings, then the movie with more ratings will, on average, have a star rating that is 0.05 stars higher (89% CI: 0.04, 0.07) than the movie with fewer ratings, according to model fit9.

## fit9 Performance and Checking

model\_performance(fit9)

# Indices of model performance  
  
AIC | AICc | BIC | R2 | R2 (adj.) | RMSE | Sigma  
------------------------------------------------------------  
-6.848 | -6.142 | -1.935 | 0.635 | 0.625 | 0.204 | 0.210

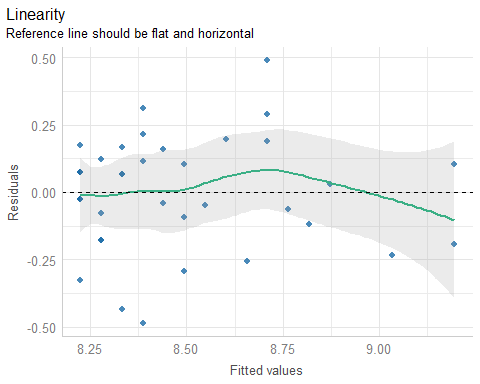
* These results with 38 movies cannot be compared to our prior results when we included all 228 movies.

fit9\_diagnostic\_plots <-   
 plot(check\_model(fit9, panel = FALSE))

For confidence bands, please install `qqplotr`.

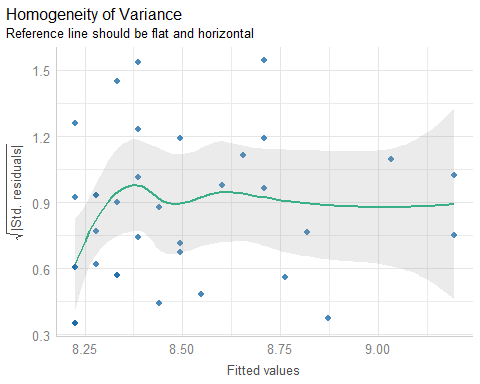
## Checking Linearity

fit9\_diagnostic\_plots[[2]]



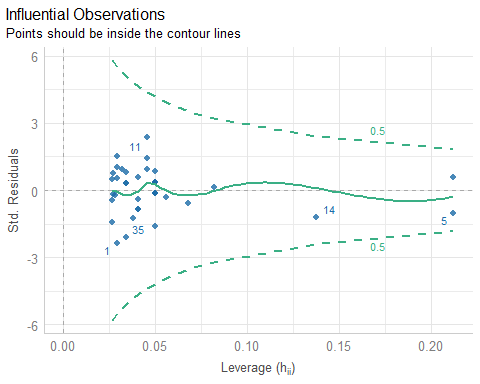
## Checking Equal Variances

fit9\_diagnostic\_plots[[3]]



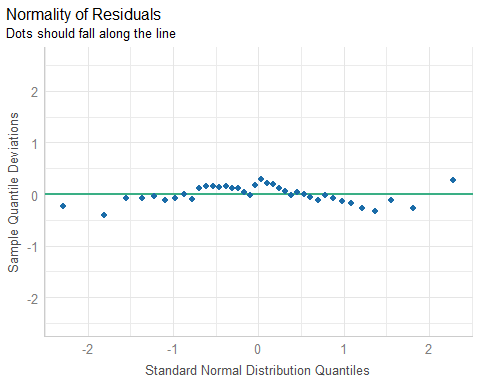
## Checking for Influential Points

fit9\_diagnostic\_plots[[4]]



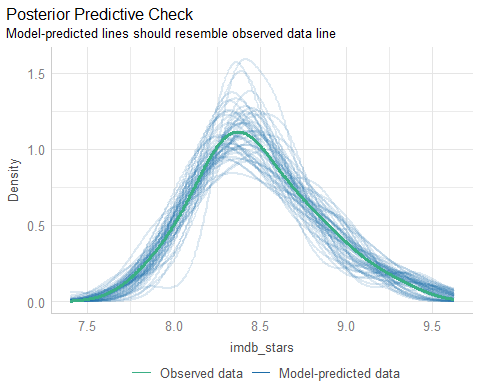
## Checking Normality

fit9\_diagnostic\_plots[[5]]



## Posterior Predictive Checks

fit9\_diagnostic\_plots[[1]]



## Additional Checks

If you’re desperate, there are some tests / checks…

check\_heteroscedasticity(fit9)

OK: Error variance appears to be homoscedastic (p = 0.645).

check\_outliers(fit9)

OK: No outliers detected.  
- Based on the following method and threshold: cook (0.707).  
- For variable: (Whole model)

check\_normality(fit9)

OK: residuals appear as normally distributed (p = 0.723).

|  |
| --- |
| Note |
| * Models fit6 and fit8 passed only check\_outliers(). * Models for movies with at least 200K, 300K and 500K IMDB ratings also passed only the outlier check. |

## Bayesian Model fit10 for 38 movies

Remember: Set seed; switch to stan\_glm(), use refresh = 0.

set.seed(20240926)  
fit10 <- stan\_glm(imdb\_stars ~ users\_100k, data = mov5, refresh = 0)  
model\_parameters(fit10, ci = 0.89)

Parameter | Median | 89% CI | pd | Rhat | ESS | Prior  
------------------------------------------------------------------------------------  
(Intercept) | 7.63 | [7.45, 7.82] | 100% | 1.000 | 3670.00 | Normal (8.48 +- 0.86)  
users\_100k | 0.05 | [0.04, 0.06] | 100% | 1.000 | 3572.00 | Normal (0.00 +- 0.17)

Uncertainty intervals (equal-tailed) and p-values (two-tailed) computed  
 using a MCMC distribution approximation.

* If two movies with over 1 million IMDB ratings have a 100,000 user difference in IMDB ratings, then the movie with more ratings will, on average, have a star rating that is 0.05 stars higher (89% CI: 0.04, 0.06) than the movie with fewer ratings, according to fit10.

## fit10 Performance and Checking

model\_performance(fit10)

# Indices of model performance  
  
ELPD | ELPD\_SE | LOOIC | LOOIC\_SE | WAIC | R2 | R2 (adj.) | RMSE | Sigma  
--------------------------------------------------------------------------------  
3.286 | 4.555 | -6.572 | 9.109 | -6.636 | 0.620 | 0.598 | 0.204 | 0.214

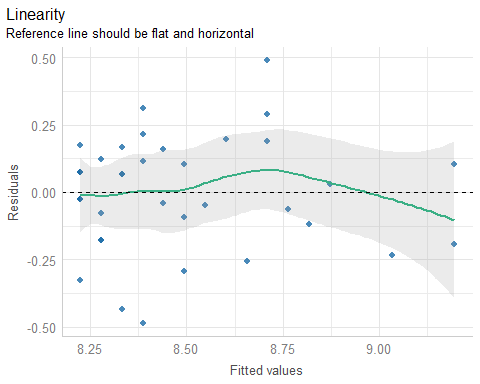
* Note that we have some new summaries now. The , RMSE and Sigma values can be compared to fit9 which used the same data. On the whole, fit10 looks *slightly* worse than fit9 on these metrics.
* Let’s get the diagnostic plots for fit10.

fit10\_diagnostic\_plots <-   
 plot(check\_model(fit10, panel = FALSE))

For confidence bands, please install `qqplotr`.

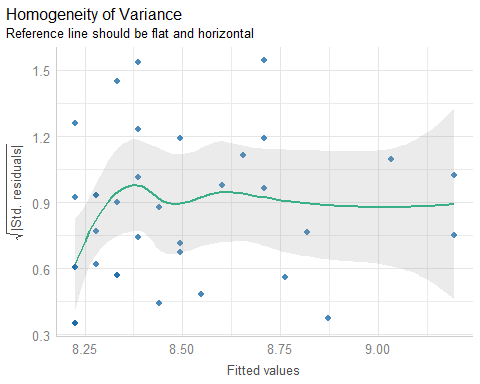
## Checking Linearity

fit10\_diagnostic\_plots[[2]]



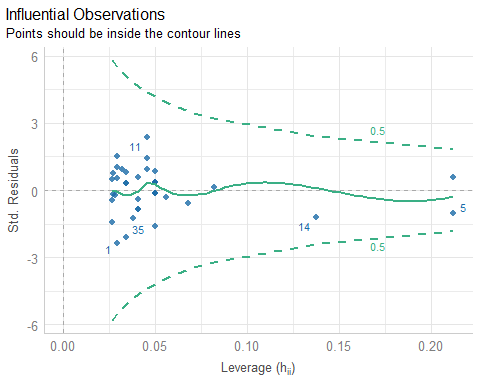
## Checking Equal Variances

fit10\_diagnostic\_plots[[3]]



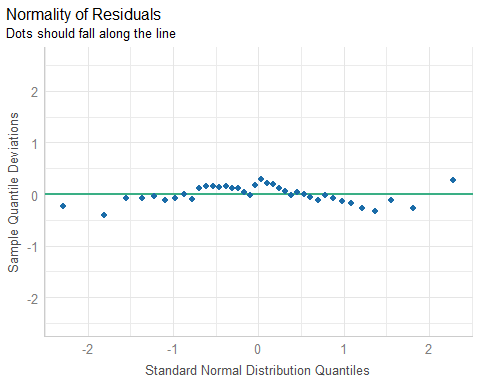
## Checking for Influential Points

fit10\_diagnostic\_plots[[4]]



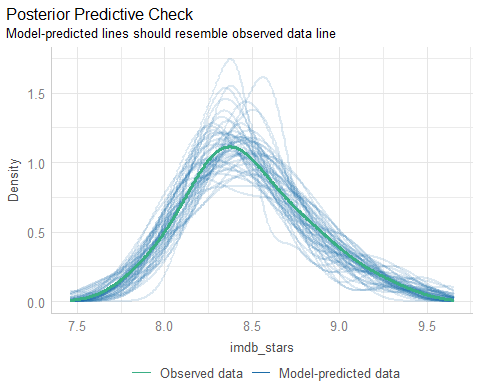
## Checking Normality

fit10\_diagnostic\_plots[[5]]



## Posterior Predictive Checks

fit10\_diagnostic\_plots[[1]]



## Not much to choose from here…

fit10 and fit9 are pretty similar in terms of estimated parameters, performance metrics, and diagnostic checks.

check\_heteroscedasticity(fit10)

OK: Error variance appears to be homoscedastic (p = 0.626).

check\_outliers(fit10)

OK: No outliers detected.  
- Based on the following method and threshold: pareto (0.7).  
- For variable: (Whole model)

## Session Information

xfun::session\_info()

R version 4.4.1 (2024-06-14 ucrt)  
Platform: x86\_64-w64-mingw32/x64  
Running under: Windows 11 x64 (build 22631)  
  
Locale:  
 LC\_COLLATE=English\_United States.utf8   
 LC\_CTYPE=English\_United States.utf8   
 LC\_MONETARY=English\_United States.utf8  
 LC\_NUMERIC=C   
 LC\_TIME=English\_United States.utf8   
  
Package version:  
 abind\_1.4-5 askpass\_1.2.0 backports\_1.5.0   
 base64enc\_0.1-3 bayesplot\_1.11.1 bayestestR\_0.14.0   
 BH\_1.84.0.0 bit\_4.0.5 bit64\_4.0.5   
 blob\_1.2.4 boot\_1.3-31 broom\_1.0.6   
 bslib\_0.8.0 cachem\_1.1.0 callr\_3.7.6   
 car\_3.1-2 carData\_3.0-5 cellranger\_1.1.0   
 checkmate\_2.3.2 cli\_3.6.3 clipr\_0.8.0   
 coda\_0.19-4.1 codetools\_0.2-20 colorspace\_2.1-1   
 colourpicker\_1.3.0 commonmark\_1.9.1 compiler\_4.4.1   
 conflicted\_1.2.0 correlation\_0.8.5 cowplot\_1.1.3   
 cpp11\_0.5.0 crayon\_1.5.3 crosstalk\_1.2.1   
 curl\_5.2.2 data.table\_1.16.0 datasets\_4.4.1   
 datawizard\_0.12.3 DBI\_1.2.3 dbplyr\_2.5.0   
 Deriv\_4.1.3 desc\_1.4.3 digest\_0.6.37   
 distributional\_0.4.0 doBy\_4.6.22 dplyr\_1.1.4   
 DT\_0.33 dtplyr\_1.3.1 dygraphs\_1.1.1.6   
 easystats\_0.7.3 effectsize\_0.8.9 emmeans\_1.10.4   
 estimability\_1.5.1 evaluate\_0.24.0 fansi\_1.0.6   
 farver\_2.1.2 fastmap\_1.2.0 fontawesome\_0.5.2   
 forcats\_1.0.0 fs\_1.6.4 gargle\_1.5.2   
 generics\_0.1.3 ggplot2\_3.5.1 ggrepel\_0.9.6   
 ggridges\_0.5.6 glue\_1.7.0 googledrive\_2.1.1   
 googlesheets4\_1.1.1 graphics\_4.4.1 grDevices\_4.4.1   
 grid\_4.4.1 gridExtra\_2.3 gtable\_0.3.5   
 gtools\_3.9.5 haven\_2.5.4 highr\_0.11   
 hms\_1.1.3 htmltools\_0.5.8.1 htmlwidgets\_1.6.4   
 httpuv\_1.6.15 httr\_1.4.7 ids\_1.0.1   
 igraph\_2.0.3 infer\_1.0.7 inline\_0.3.19   
 insight\_0.20.4 isoband\_0.2.7 janitor\_2.2.0   
 jquerylib\_0.1.4 jsonlite\_1.8.8 kableExtra\_1.4.0   
 knitr\_1.48 labeling\_0.4.3 later\_1.3.2   
 lattice\_0.22-6 lazyeval\_0.2.2 lifecycle\_1.0.4   
 lme4\_1.1-35.5 loo\_2.8.0 lubridate\_1.9.3   
 magrittr\_2.0.3 markdown\_1.13 MASS\_7.3-61   
 Matrix\_1.7-0 MatrixModels\_0.5.3 matrixStats\_1.4.0   
 memoise\_2.0.1 methods\_4.4.1 mgcv\_1.9-1   
 microbenchmark\_1.5.0 mime\_0.12 miniUI\_0.1.1.1   
 minqa\_1.2.8 modelbased\_0.8.8 modelr\_0.1.11   
 multcomp\_1.4-26 munsell\_0.5.1 mvtnorm\_1.3-1   
 naniar\_1.1.0 nlme\_3.1-164 nloptr\_2.1.1   
 nnet\_7.3.19 norm\_1.0.11.1 numDeriv\_2016.8.1.1   
 openssl\_2.2.1 parallel\_4.4.1 parameters\_0.22.2   
 patchwork\_1.2.0 pbkrtest\_0.5.3 performance\_0.12.3   
 pillar\_1.9.0 pkgbuild\_1.4.4 pkgconfig\_2.0.3   
 plyr\_1.8.9 posterior\_1.6.0 prettyunits\_1.2.0   
 processx\_3.8.4 progress\_1.2.3 promises\_1.3.0   
 ps\_1.7.7 purrr\_1.0.2 quantreg\_5.98   
 QuickJSR\_1.3.1 R6\_2.5.1 ragg\_1.3.2   
 rappdirs\_0.3.3 RColorBrewer\_1.1.3 Rcpp\_1.0.13   
 RcppEigen\_0.3.4.0.2 RcppParallel\_5.1.9 readr\_2.1.5   
 readxl\_1.4.3 rematch\_2.0.0 rematch2\_2.1.2   
 report\_0.5.9 reprex\_2.1.1 reshape2\_1.4.4   
 rlang\_1.1.4 rmarkdown\_2.28 rstan\_2.32.6   
 rstanarm\_2.32.1 rstantools\_2.4.0 rstudioapi\_0.16.0   
 rvest\_1.0.4 sandwich\_3.1-0 sass\_0.4.9   
 scales\_1.3.0 see\_0.9.0 selectr\_0.4.2   
 shiny\_1.9.1 shinyjs\_2.1.0 shinystan\_2.6.0   
 shinythemes\_1.2.0 snakecase\_0.11.1 sourcetools\_0.1.7.1   
 SparseM\_1.84.2 splines\_4.4.1 StanHeaders\_2.32.10   
 stats\_4.4.1 stats4\_4.4.1 stringi\_1.8.4   
 stringr\_1.5.1 survival\_3.7-0 svglite\_2.1.3   
 sys\_3.4.2 systemfonts\_1.1.0 tensorA\_0.36.2.1   
 textshaping\_0.4.0 TH.data\_1.1-2 threejs\_0.3.3   
 tibble\_3.2.1 tidyr\_1.3.1 tidyselect\_1.2.1   
 tidyverse\_2.0.0 timechange\_0.3.0 tinytex\_0.52   
 tools\_4.4.1 tzdb\_0.4.0 UpSetR\_1.4.0   
 utf8\_1.2.4 utils\_4.4.1 uuid\_1.2.1   
 V8\_5.0.0 vctrs\_0.6.5 viridis\_0.6.5   
 viridisLite\_0.4.2 visdat\_0.6.0 vroom\_1.6.5   
 withr\_3.0.1 xfun\_0.47 xml2\_1.3.6   
 xtable\_1.8-4 xts\_0.14.0 yaml\_2.3.10   
 zoo\_1.8-12