Oct 2024

R Shiny Masterclass Series - Introduction

Data sources and data processing in R Shiny

EPI-interactive

Agenda

- **Session 1** | 30 September | Getting started with Posit Cloud and your first R Shiny app
- **Session 2** | 01 October | R Shiny core concepts and mobile ready layout
- **Session 3** | 03 October | R Shiny user interface components, reactivity and debugging
- **Session 4** | 07 October | Data sources and data processing in R Shiny
- **Session 5** | 08 October | Maps and spatial visualisation with Leaflet: adding map layers, annotations, pins, filters and legend
- **Session 6** | 10 October | Interactive charts with Plotly: chart types, customising hover boxes and chart styling
- **Session 7** | 14 October | Publishing R Shiny apps, design considerations and case study
- **Session 8** | 15 October | Case study, top 10 tips for data visualisation with R Shiny and wrap-up

Today

Recap: Session 3 challenge

Goals:

- Understand pros and cons of different data formats
- Load data from different formats into R Shiny
- Process data and use in tables
- Export processed data

Debugging

Debugging in Shiny

It is challenging:

- Reactivity, code execution isn't as linear
- Separate environment for each user session, doesn't last after session ends
- Code runs behind the Shiny framework
- R terminal is busy running the Shiny app

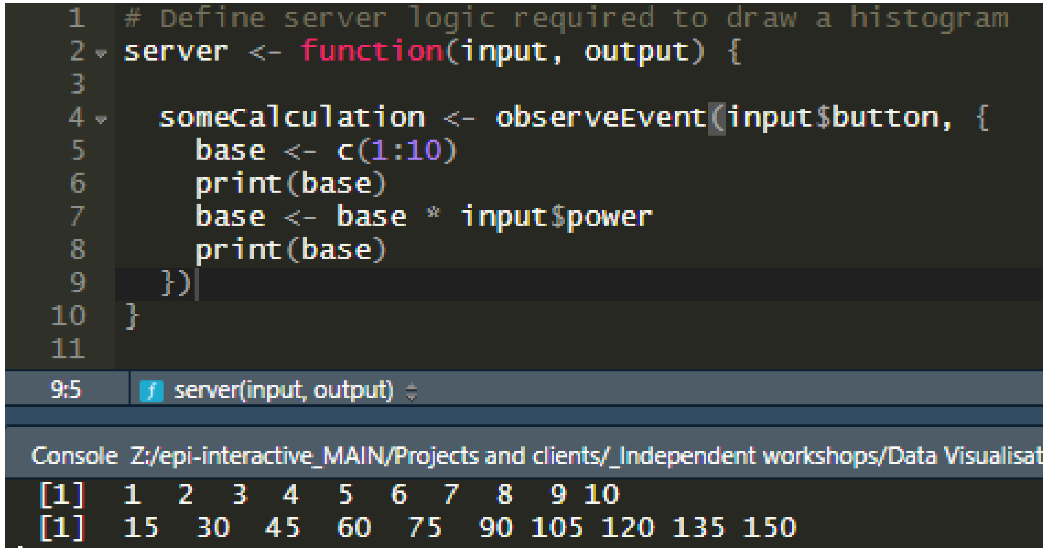
Debugging approaches

- Resetting
- Debugging
- Tracing
- Reprex
- Error handling

Debugging – Reset

- “Have you tried turning it off and on again”
- Need to check if you can reproduce the issue to debug effectively.
- Clear Environment
  - Objects created in global.R or console are stored in the global environment.
  - Clearing environment can prevent issues with left-over variables etc.
  - *R --no-save --no-restore-data*
- Restart R session
  - Can be useful for fixing caching issues (especially theming related)
  - Last resort
- *“Environment should be like Livestock, not house pets”*

Debugging – print()



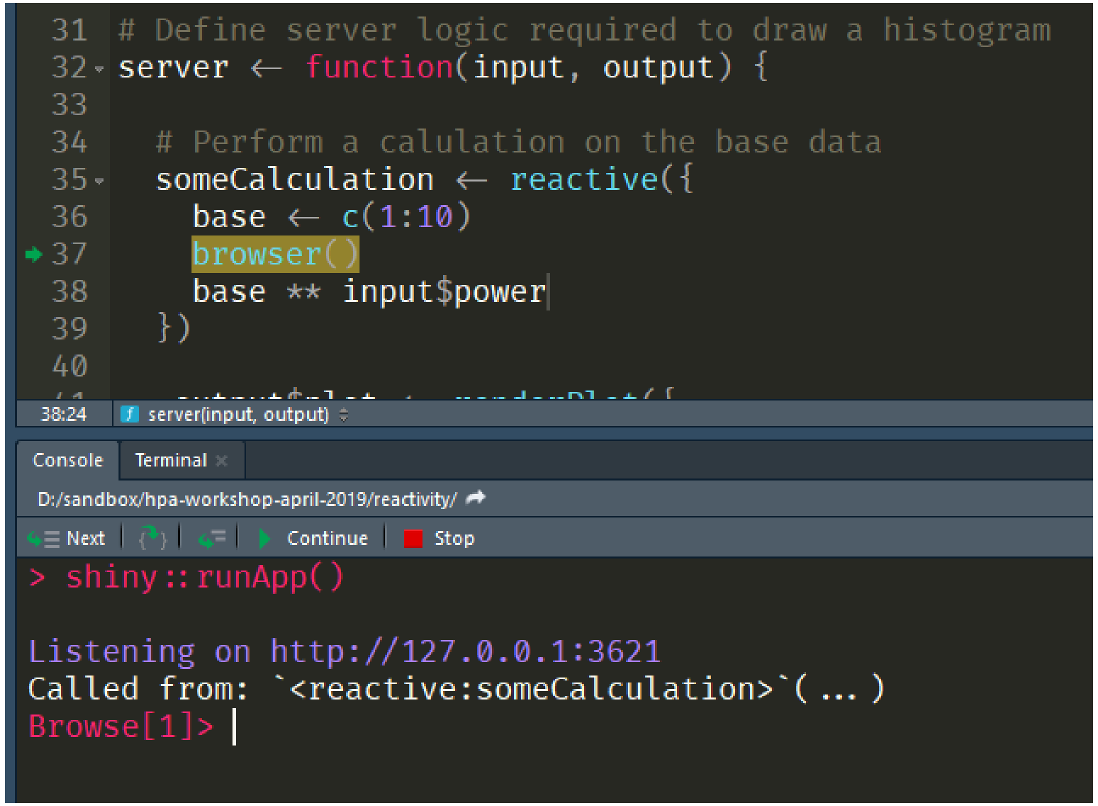
```r
# Define server logic required to draw a histogram
server <- function(input, output) {

  someCalculation <- observeEvent(input$button, {
    base <- c(1:10)
    print(base)
    base <- base * input$power
    print(base)
  })
}

```

9:5
server(input, output)
Console Z:/epi-interactive_MAIN/Projects and clients/_Independent workshops/Data Visualisat

```
 [1]  1   2   3   4   5   6   7   8   9  10
 [1]  15   30   45   60   75   90  105  120  135  150
```

- Simple and versatile
- Can check the control flow of an application.
- Can check values during execution
- Good for quick checks

Debugging – browser()



```
# Define server logic required to draw a histogram
server <- function(input, output) {

  # Perform a calulation on the base data
  someCalculation <- reactive({
    base <- c(1:10)
    browser()
    base ** input$power
  })

  output$plot <- renderPlot({
```

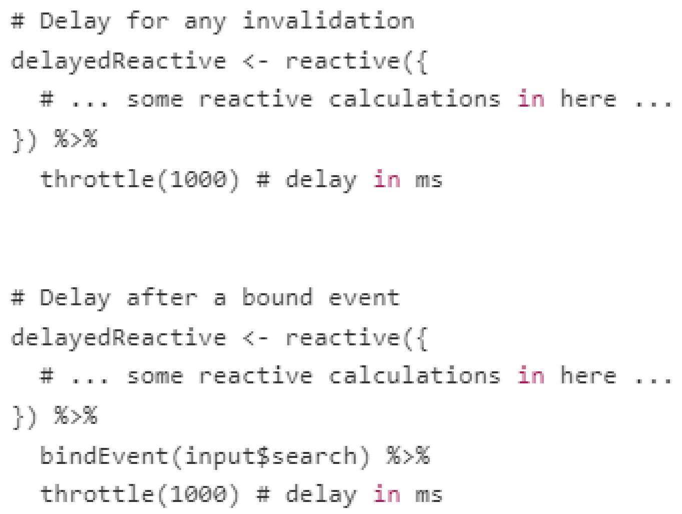
D:/sandbox/hpa-workshop-april-2019/reactivity/

```
> shiny::runApp()

Listening on http://127.0.0.1:3621
Called from: `<reactive:someCalculation>`(...)
Browse[1]> 
```

- Stops the app and lets us step through each line of code manually
- Great for examining reactive values or for more complex checks
- Execute code line by line, enter functions, stop the app, and use the console

Reproducable **Ex**amples (Reprex)

- Code snippets
- Often used in case of error occurring
- Displayed for simplest case
- Remove unnecessary/excess code



```
# Delay for any invalidation
delayedReactive <- reactive({
  # ... some reactive calculations in here ...
}) %>%
  throttle(1000) # delay in ms


# Delay after a bound event
delayedReactive <- reactive({
  # ... some reactive calculations in here ...
}) %>%
  bindEvent(input$search) %>%
  throttle(1000) # delay in ms
```

Data sources in Shiny

Data Types

- **Arbitrary data** can be stored as a **file** in some sort of a file system (local file system, Dropbox, Amazon S3)
- **Structured rectangular data** can be stored as **tables** in a **relational database or table-storage** service (SQLite, MySQL, Google Sheets)
- **Semi-structured data** can be stored as a **collection** in a **NoSQL database**

Data Source

Local:

-Hosted in the same environment as Shiny application

-Use *write.csv()*, *write.table()*, and *saveRDS()* to implement local storage

-Faster than remote storage

Remote:

-Hosted on another server (Amazon, Azure, Google, External database)

Arbitrary data – plain text / binary

Comma-separated files (CSV, Excel):

- Be easily imported/exported by R and other applications
- In an ASCII format, not very efficient

R single object (RDS)

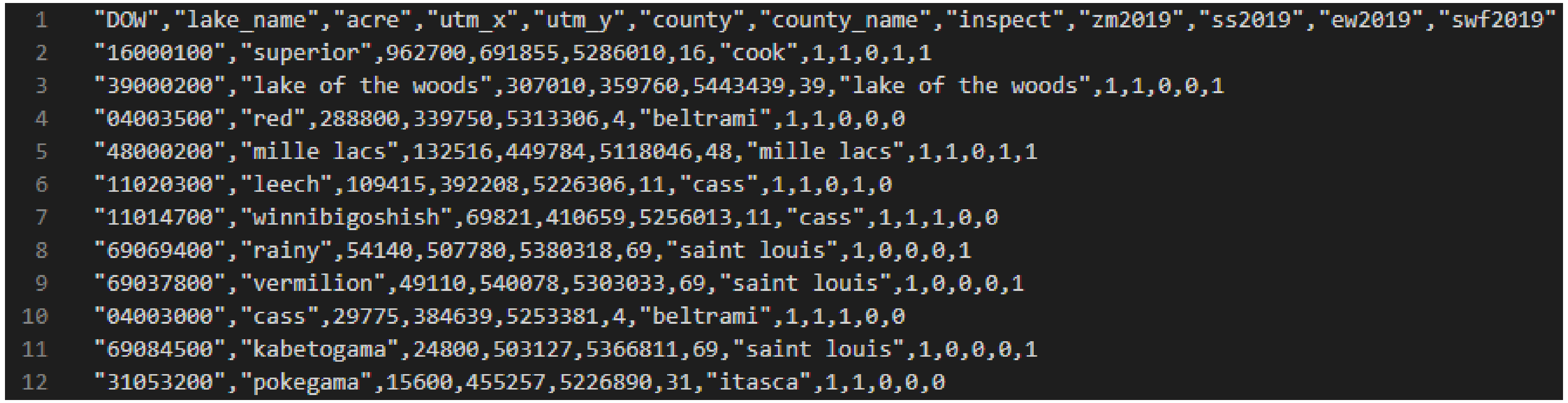
- Mainly designed for R
- In binary format, fairly efficient
- Less disk space used

R workspace object (RData)

- Collection of single R objects
- Also stored in binary format
- Rstudio global environment can be saved as RData

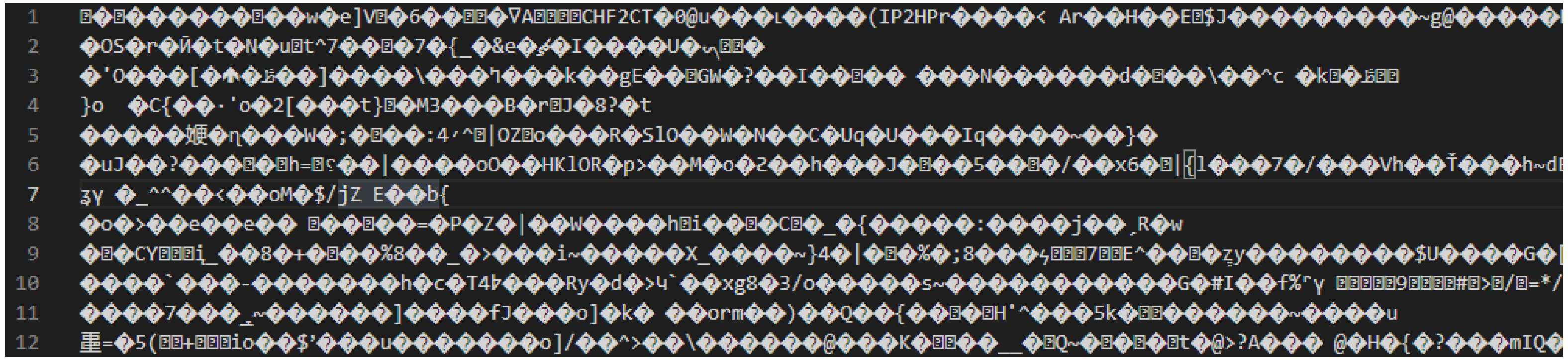
CSV vs RDS

- AIS lake config
- 9000 rows:
    - CSV: 568 KB
    - RDS: 256 KB



```
"DOW","lake_name","acre","utm_x","utm_y","county","county_name","inspect","zm2019","ss2019","ew2019","swf2019"
"16000100","superior",962700,691855,5286010,16,"cook",1,1,0,1,1
"39000200","lake of the woods",307010,359760,5443439,39,"lake of the woods",1,1,0,0,1
"04003500","red",288800,339750,5313306,4,"beltrami",1,1,0,0,0
"48000200","mille lacs",132516,449784,5118046,48,"mille lacs",1,1,0,1,1
"11020300","leech",109415,392208,5226306,11,"cass",1,1,0,1,0
"11014700","winnibigoshish",69821,410659,5256013,11,"cass",1,1,1,0,0
"69069400","rainy",54140,507780,5380318,69,"saint louis",1,0,0,0,1
"69037800","vermilion",49110,540078,5303033,69,"saint louis",1,0,0,0,1
"04003000","cass",29775,384639,5253381,4,"beltrami",1,1,1,0,0
"69084500","kabetogama",24800,503127,5366811,69,"saint louis",1,0,0,0,1
"31053200","pokegama",15600,455257,5226890,31,"itasca",1,1,0,0,0
```

- CSV: readable
- RDS: not



```
▯�▯�����▯��w�e]V▯�6��▯▯�∇A▯▯▯▯CHF2CT�0@u���ɩ����(IP2HPr����< Ar��H��E▯$J����������~g@������
�OS�r�Й�t�N�u▯t^7���▯�7�{_�&e�ێ�I�����U�ᵧ▯▯�
�'O���[�↑�Ɇ��]����\���ᄀ���k��gE��▯GW�?��I��▯�� ���N��������d�▯��\��^c �k▯�ᴟ▯▯
}o  �C{��·'o�2[���t}▯�M3���B�r▯J�8?�t
������婹�η���W�;�▯��:4ʻ^▯|OZ▯o����R�SlO��W�N���C�Uq�U����Iq�����~��}�
�uJ��?�����▯�▯h=▯ჺ��|�����oO��HKlOR�p>��M�o�2��h����J�▯��5���▯�/��x6�▯|{l����7�/����Vh��Ť����h~dE
ʒγ �_^^��<���oM�$/jZ E��b{
�o�>��e��e�� ▯��▯��=�P�Z�|��W����h▯i���▯�C▯�_�{������:����j�� ˛R�w
�▯�CY▯▯▯i̜_��8�+�▯��%8��_�>����i~�������X_������~}4�|�▯�%�;8����ϟ▯▯▯7▯▯E^���▯�ẓy�����������$U�����G�
�����`����-����������h�c�T4♭����Ry�d�>Ч`��xg8�3/o������s~��������������G�#I���f%ˀγ ▯▯▯▯▯9▯▯▯▯#▯>▯/▯=*/
�����7����_̥~�������]�����fJ����o]�k� ��orm���)���Q���{���▯��▯H'^����5k�▯▯���������~�����u
重=�5(▯▯+▯▯▯io���$'����u����������o]/���^>���\��������@����K�▯▯��__�▯Q~�▯�▯�▯t�@>?A���� @�H�{�?����mIQ�
```

Things to consider

- Size of data (data points, type of information)
- Structure of the data
- Does it change? If yes, how often?
- Privacy (cloud vs local)

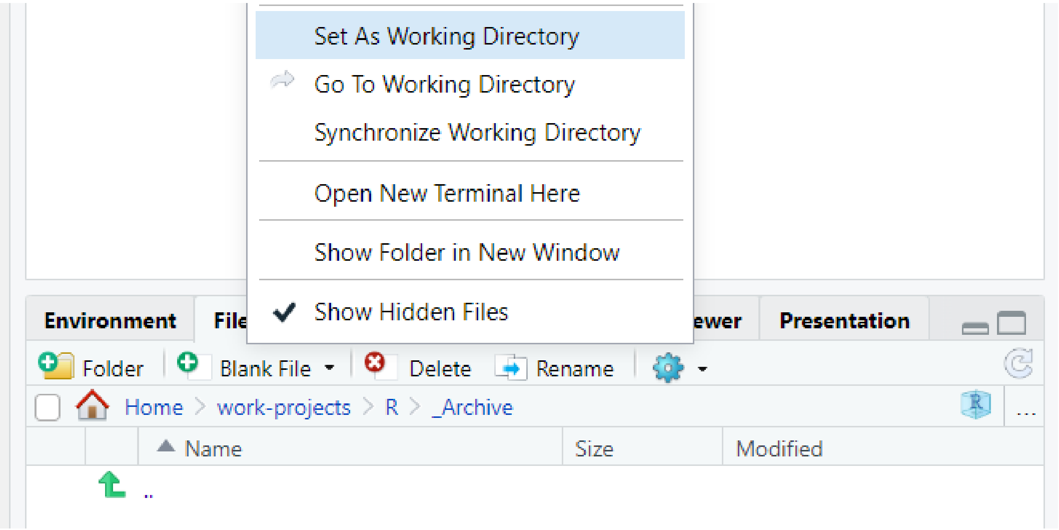
Global.R

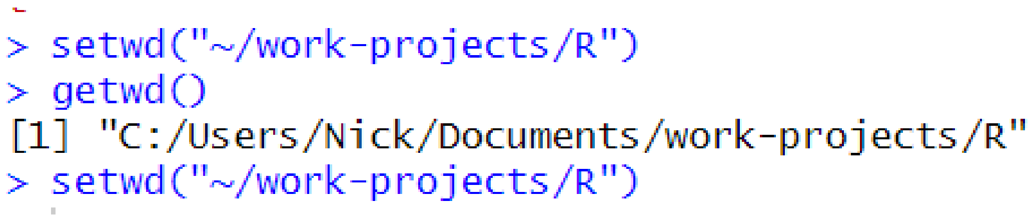
Addition to the ui.R & server.R pattern:

- Run R code before application launches
- Variables save to global environment, persist between sessions
- Accessible throughout entire project / scope
- More useful in complex applications

Working directory

- Where in your file system R is currently looking to read and write files
- When running a Shiny app, the level in the file structure containing ui.R and server.R becomes the working directory
- Any file paths must be in relation to working directory





```
> setwd("~/work-projects/R")
> getwd()
[1] "C:/Users/Nick/Documents/work-projects/R"
> setwd("~/work-projects/R")
```

Loading data from source

We need to load data into the R Shiny application before it can be used.

Some common functions:

- For CSV files: read.csv(“filepath”, header = TRUE)
- For RDS files: readRDS(“filepath”)

- Open **Session 4**, then */stage1*
- Create a new file called global.R
  - Move any library() function calls into this file
- In *global.R*, use read.csv or readRDS to load the CSV / RDS data
- In *server.R*, use the data loaded in global.R to create a data table
  - (hint: we can use datatable() to create a DT table)

Viewing data

Can be useful to inspect the data manually before processing it

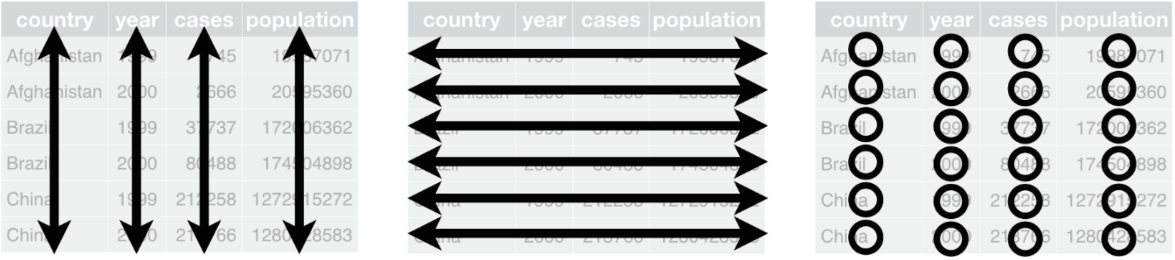
- Figure out data layout / structure
- View contents of data frame
- Work out processing steps for data
- View(data)

Make sure to remove View() from the code when you are done!

Data processing in Shiny

Manipulating data

- Modifying data after loading to make it suitable for presenting or using in visualisations.
- Variable: values with a common attribute
- Observation: all values measured on same unit
- Each column should represent a single variable, each row should represent a single observation.



variables observations values

https://r4ds.had.co.nz/tidy-data.html

Manipulating data

The “Tidyverse” packages from RStudio make data manipulation easy.

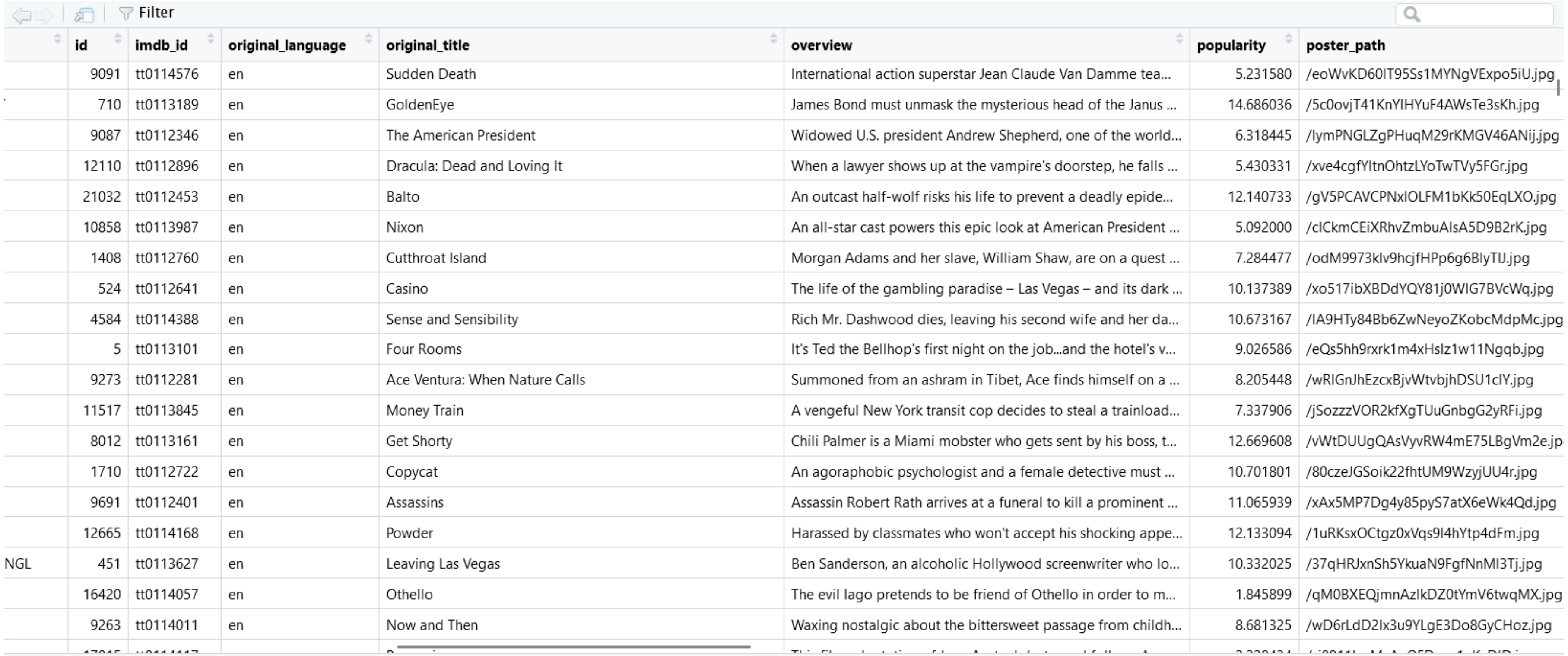
Some of the most useful tidyverse functions:

- rename(“new_name” = “old_name”)
- mutate(col_name = col_name * 5)
- select(col_name, …)
- filter(col_name == …)

- group_by(col_name)
- summarise(…)

https://www.tidyverse.org/

Movie dataset example - Start

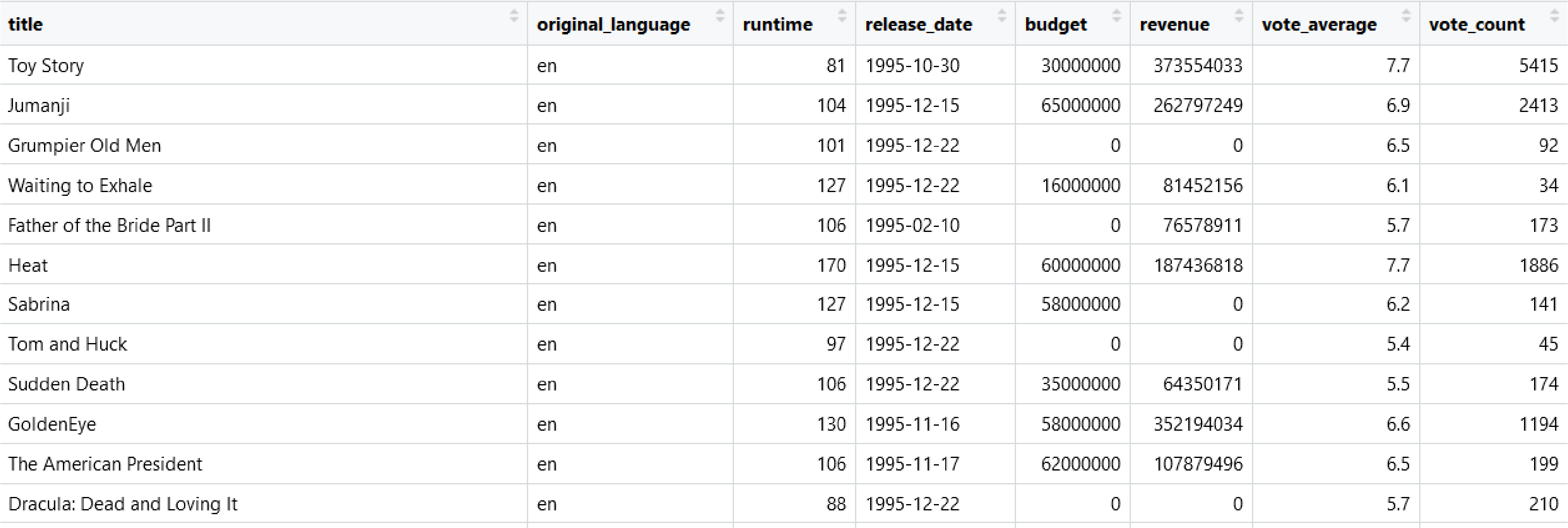


	id	imdb_id	original_language	original_title	overview	popularity	poster_path
	9091	tt0114576	en	Sudden Death	International action superstar Jean Claude Van Damme tea...	5.231580	/eoWvKD60IT95Ss1MYNgVExpo5iU.jpg
	710	tt0113189	en	GoldenEye	James Bond must unmask the mysterious head of the Janus ...	14.686036	/5c0ovjT41KnYIHYuF4AWsTe3sKh.jpg
	9087	tt0112346	en	The American President	Widowed U.S. president Andrew Shepherd, one of the world...	6.318445	/lymPNGLZgPHuqM29rKMGV46ANij.jpg
	12110	tt0112896	en	Dracula: Dead and Loving It	When a lawyer shows up at the vampire's doorstep, he falls ...	5.430331	/xve4cgfYItnOhtzLYoTwTVy5FGr.jpg
	21032	tt0112453	en	Balto	An outcast half-wolf risks his life to prevent a deadly epide...	12.140733	/gV5PCAVCPNxlOLFM1bKk50EqLXO.jpg
	10858	tt0113987	en	Nixon	An all-star cast powers this epic look at American President ...	5.092000	/cICkmCEiXRhvZmbuAIsA5D9B2rK.jpg
	1408	tt0112760	en	Cutthroat Island	Morgan Adams and her slave, William Shaw, are on a quest ...	7.284477	/odM9973kIv9hcjfHPp6g6BlyTIJ.jpg
	524	tt0112641	en	Casino	The life of the gambling paradise – Las Vegas – and its dark ...	10.137389	/xo517ibXBDdYQY81j0WIG7BVcWq.jpg
	4584	tt0114388	en	Sense and Sensibility	Rich Mr. Dashwood dies, leaving his second wife and her da...	10.673167	/lA9HTy84Bb6ZwNeyoZKobcMdpMc.jpg
	5	tt0113101	en	Four Rooms	It's Ted the Bellhop's first night on the job...and the hotel's v...	9.026586	/eQs5hh9rxrk1m4xHsIz1w11Ngqb.jpg
	9273	tt0112281	en	Ace Ventura: When Nature Calls	Summoned from an ashram in Tibet, Ace finds himself on a ...	8.205448	/wRlGnJhEzcxBjvWtvbjhDSU1cIY.jpg
	11517	tt0113845	en	Money Train	A vengeful New York transit cop decides to steal a trainload...	7.337906	/jSozzzVOR2kfXgTUuGnbgG2yRFi.jpg
	8012	tt0113161	en	Get Shorty	Chili Palmer is a Miami mobster who gets sent by his boss, t...	12.669608	/vWtDUUgQAsVyvRW4mE75LBgVm2e.jp
	1710	tt0112722	en	Copycat	An agoraphobic psychologist and a female detective must ...	10.701801	/80czeJGSoik22fhtUM9WzyjUU4r.jpg
	9691	tt0112401	en	Assassins	Assassin Robert Rath arrives at a funeral to kill a prominent ...	11.065939	/xAx5MP7Dg4y85pyS7atX6eWk4Qd.jpg
	12665	tt0114168	en	Powder	Harassed by classmates who won't accept his shocking appe...	12.133094	/1uRKsxOCtgz0xVqs9I4hYtp4dFm.jpg
NGL	451	tt0113627	en	Leaving Las Vegas	Ben Sanderson, an alcoholic Hollywood screenwriter who lo...	10.332025	/37qHRJxnSh5YkuaN9FgfNnMI3Tj.jpg
	16420	tt0114057	en	Othello	The evil Iago pretends to be friend of Othello in order to m...	1.845899	/qM0BXEQjmnAzIkDZ0tYmV6twqMX.jpg
	9263	tt0114011	en	Now and Then	Waxing nostalgic about the bittersweet passage from childh...	8.681325	/wD6rLdD2Ix3u9YLgE3Do8GyCHoz.jpg

```
> colnames(g_start_movie_data)
 [1] "adult"                 "belongs_to_collection" "budget"                "genres"                "homepage"              "id"
 [7] "imdb_id"               "original_language"     "original_title"        "overview"              "popularity"            "poster_path"
[13] "production_companies"  "production_countries"  "release_date"          "revenue"               "runtime"               "spoken_languages"
[19] "status"                "tagline"               "title"                 "video"                 "vote_average"          "vote_count"
```

Movie dataset example - Select

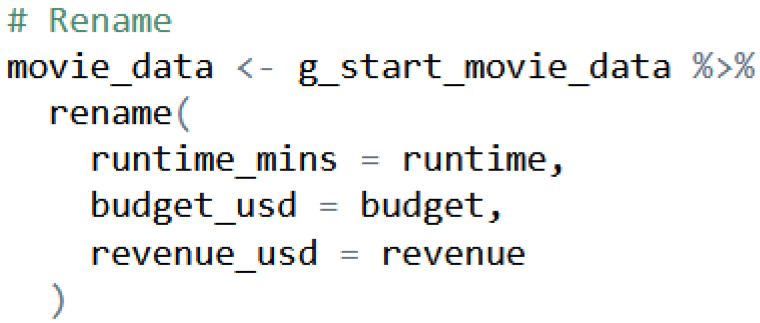
```r
g_start_movie_data <- readr::read_csv("data/movies_metadata.csv") %>%
  select(
    title,
    original_language,
    runtime,
    release_date,
    budget,
    revenue,
    vote_average,
    vote_count
  )
```



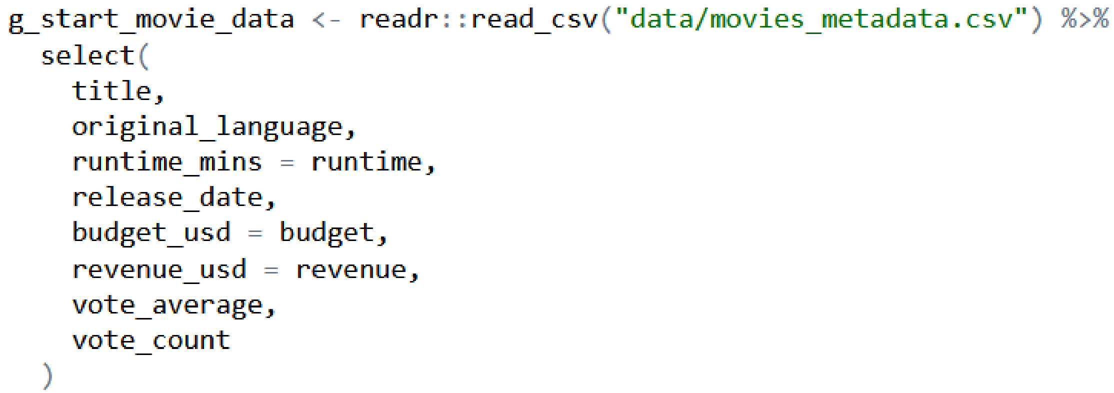
title	original_language	runtime	release_date	budget	revenue	vote_average	vote_count
Toy Story	en	81	1995-10-30	30000000	373554033	7.7	5415
Jumanji	en	104	1995-12-15	65000000	262797249	6.9	2413
Grumpier Old Men	en	101	1995-12-22	0	0	6.5	92
Waiting to Exhale	en	127	1995-12-22	16000000	81452156	6.1	34
Father of the Bride Part II	en	106	1995-02-10	0	76578911	5.7	173
Heat	en	170	1995-12-15	60000000	187436818	7.7	1886
Sabrina	en	127	1995-12-15	58000000	0	6.2	141
Tom and Huck	en	97	1995-12-22	0	0	5.4	45
Sudden Death	en	106	1995-12-22	35000000	64350171	5.5	174
GoldenEye	en	130	1995-11-16	58000000	352194034	6.6	1194
The American President	en	106	1995-11-17	62000000	107879496	6.5	199
Dracula: Dead and Loving It	en	88	1995-12-22	0	0	5.7	210

EPI-interactive

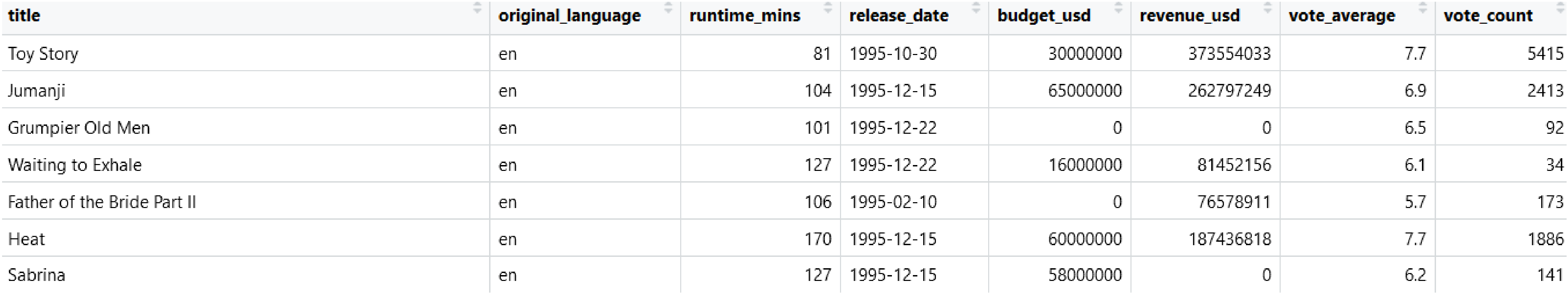
Movie dataset example - Rename



```
# Rename
movie_data <- g_start_movie_data %>%
  rename(
    runtime_mins = runtime,
    budget_usd = budget,
    revenue_usd = revenue
  )
```

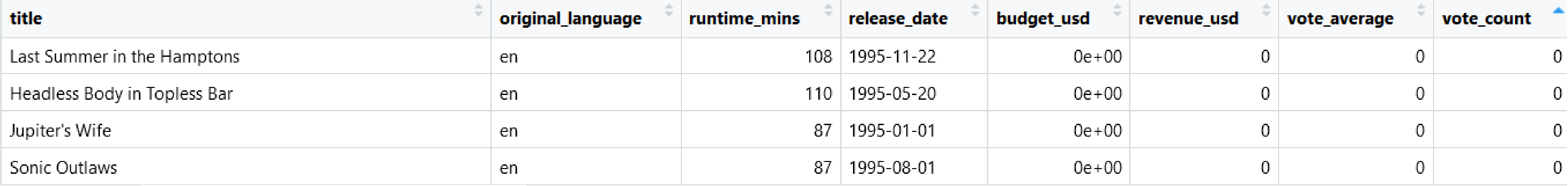


```
g_start_movie_data <- readr::read_csv("data/movies_metadata.csv") %>%
  select(
    title,
    original_language,
    runtime_mins = runtime,
    release_date,
    budget_usd = budget,
    revenue_usd = revenue,
    vote_average,
    vote_count
  )
```

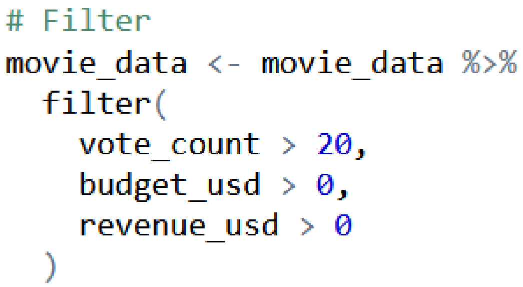


title	original_language	runtime_mins	release_date	budget_usd	revenue_usd	vote_average	vote_count
Toy Story	en	81	1995-10-30	30000000	373554033	7.7	5415
Jumanji	en	104	1995-12-15	65000000	262797249	6.9	2413
Grumpier Old Men	en	101	1995-12-22	0	0	6.5	92
Waiting to Exhale	en	127	1995-12-22	16000000	81452156	6.1	34
Father of the Bride Part II	en	106	1995-02-10	0	76578911	5.7	173
Heat	en	170	1995-12-15	60000000	187436818	7.7	1886
Sabrina	en	127	1995-12-15	58000000	0	6.2	141

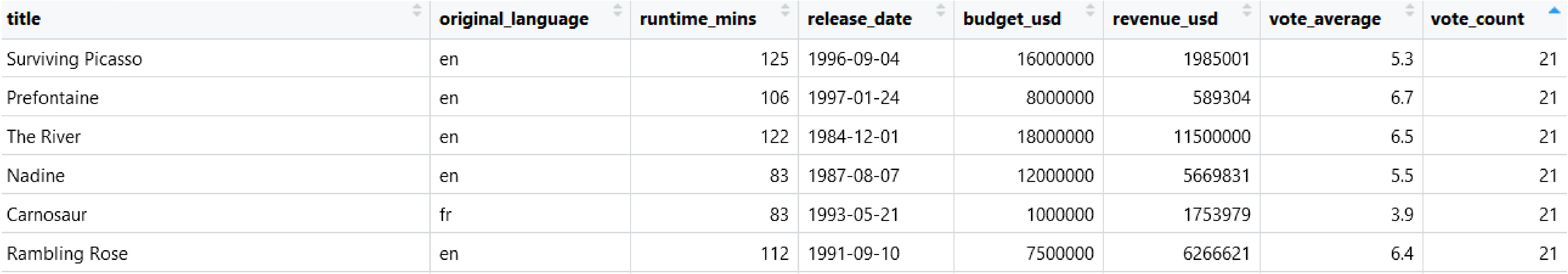
Movie dataset example - Filter



title	original_language	runtime_mins	release_date	budget_usd	revenue_usd	vote_average	vote_count
Last Summer in the Hamptons	en	108	1995-11-22	0e+00	0	0	0
Headless Body in Topless Bar	en	110	1995-05-20	0e+00	0	0	0
Jupiter's Wife	en	87	1995-01-01	0e+00	0	0	0
Sonic Outlaws	en	87	1995-08-01	0e+00	0	0	0



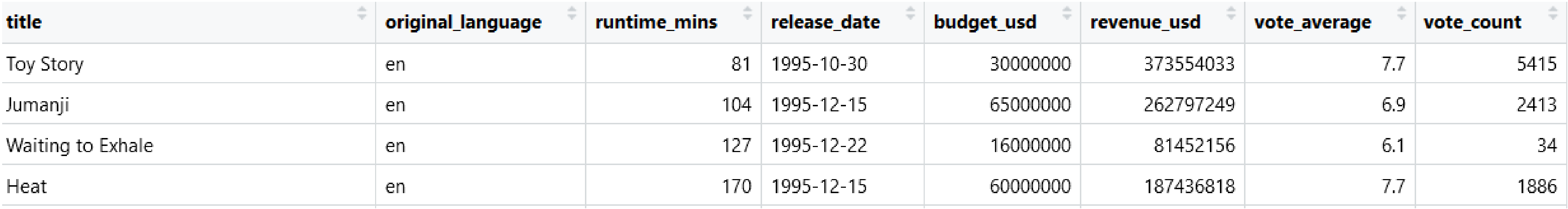
```r
# Filter
movie_data <- movie_data %>%
  filter(
    vote_count > 20,
    budget_usd > 0,
    revenue_usd > 0
  )
```



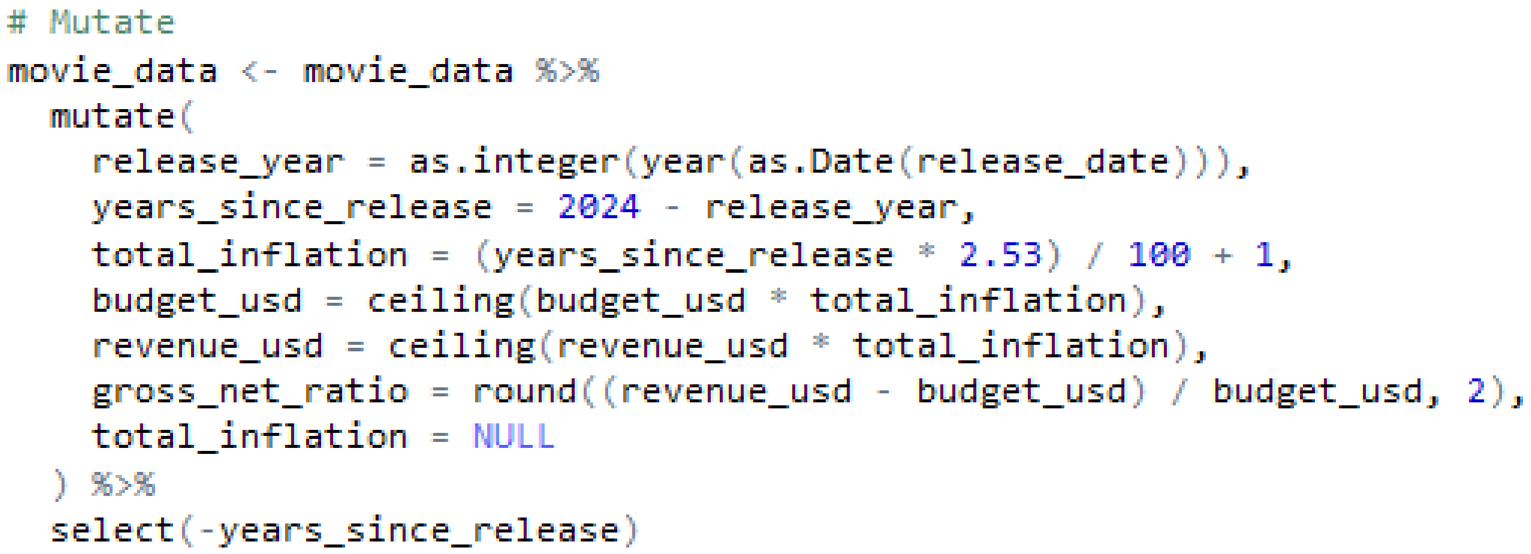
title	original_language	runtime_mins	release_date	budget_usd	revenue_usd	vote_average	vote_count
Surviving Picasso	en	125	1996-09-04	16000000	1985001	5.3	21
Prefontaine	en	106	1997-01-24	8000000	589304	6.7	21
The River	en	122	1984-12-01	18000000	11500000	6.5	21
Nadine	en	83	1987-08-07	12000000	5669831	5.5	21
Carnosaur	fr	83	1993-05-21	1000000	1753979	3.9	21
Rambling Rose	en	112	1991-09-10	7500000	6266621	6.4	21

EPI-interactive

Movie dataset example - Mutate



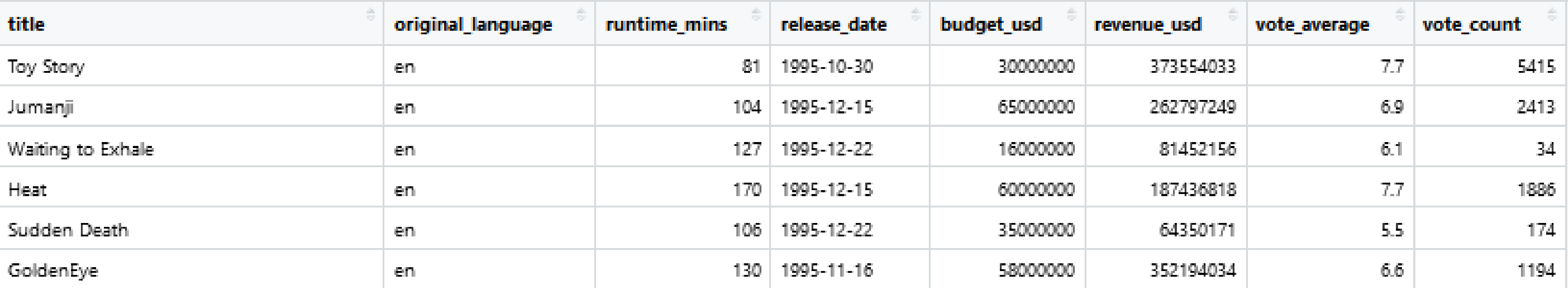
title	original_language	runtime_mins	release_date	budget_usd	revenue_usd	vote_average	vote_count
Toy Story	en	81	1995-10-30	30000000	373554033	7.7	5415
Jumanji	en	104	1995-12-15	65000000	262797249	6.9	2413
Waiting to Exhale	en	127	1995-12-22	16000000	81452156	6.1	34
Heat	en	170	1995-12-15	60000000	187436818	7.7	1886



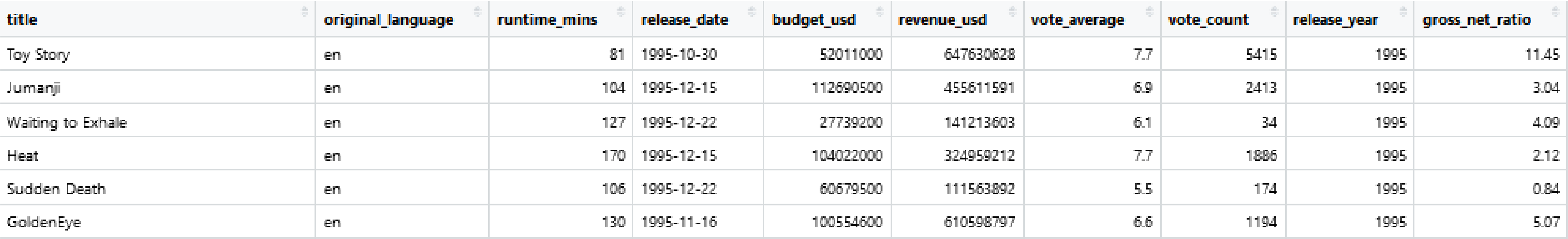
```r
# Mutate
movie_data <- movie_data %>%
  mutate(
    release_year = as.integer(year(as.Date(release_date))),
    years_since_release = 2024 - release_year,
    total_inflation = (years_since_release * 2.53) / 100 + 1,
    budget_usd = ceiling(budget_usd * total_inflation),
    revenue_usd = ceiling(revenue_usd * total_inflation),
    gross_net_ratio = round((revenue_usd - budget_usd) / budget_usd, 2),
    total_inflation = NULL
  ) %>%
  select(-years_since_release)
```

EPI-interactive

Movie dataset example - Mutate



title	original_language	runtime_mins	release_date	budget_usd	revenue_usd	vote_average	vote_count
Toy Story	en	81	1995-10-30	30000000	373554033	7.7	5415
Jumanji	en	104	1995-12-15	65000000	262797249	6.9	2413
Waiting to Exhale	en	127	1995-12-22	16000000	81452156	6.1	34
Heat	en	170	1995-12-15	60000000	187436818	7.7	1886
Sudden Death	en	106	1995-12-22	35000000	64350171	5.5	174
GoldenEye	en	130	1995-11-16	58000000	352194034	6.6	1194



title	original_language	runtime_mins	release_date	budget_usd	revenue_usd	vote_average	vote_count	release_year	gross_net_ratio
Toy Story	en	81	1995-10-30	52011000	647630628	7.7	5415	1995	11.45
Jumanji	en	104	1995-12-15	112690500	455611591	6.9	2413	1995	3.04
Waiting to Exhale	en	127	1995-12-22	27739200	141213603	6.1	34	1995	4.09
Heat	en	170	1995-12-15	104022000	324959212	7.7	1886	1995	2.12
Sudden Death	en	106	1995-12-22	60679500	111563892	5.5	174	1995	0.84
GoldenEye	en	130	1995-11-16	100554600	610598797	6.6	1194	1995	5.07

Group by & summarise

- `group_by(columns)`
  - Given one or many columns to use as grouping variables
  - Will assign groups based on unique combinations of grouping variables
  - Further operations will be applied “by group”
    - E.g. summary statistics (sum, max, mean…) applied per group instead of over entire dataset)

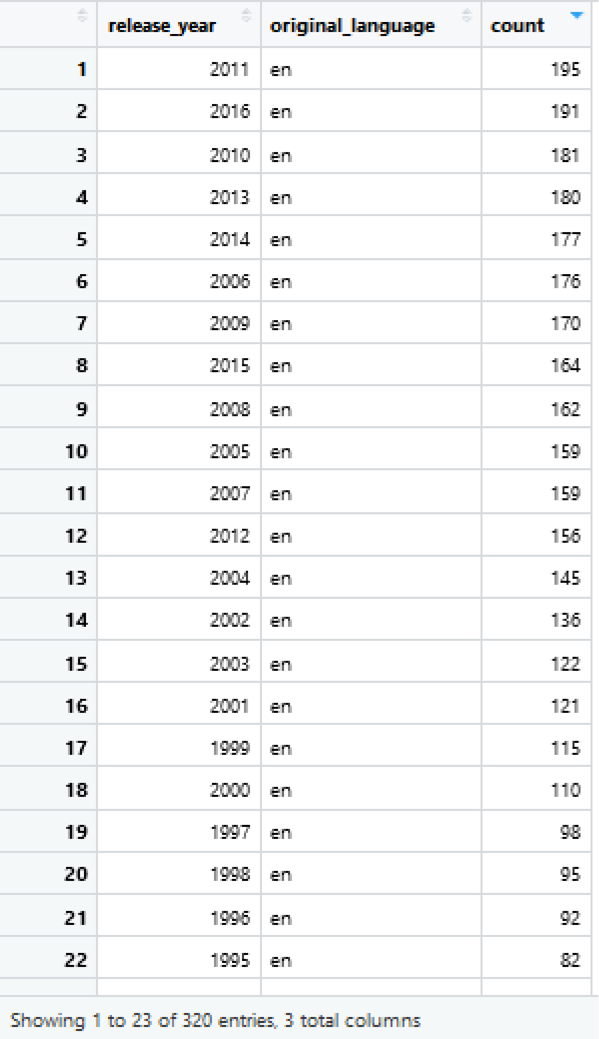
  - *`group_by(data, gender)`* will create two groups – male and female
  - *`mutate(data, count = n())`* will then have different count values for male / female rows in data.

Group by & summarise

- Creates a new data frame with one (or more) rows for each unique combination of grouping variables
  - If input data was not grouped with group_by, one row output
  - If input data was grouped with group_by, then num rows == num groups
- Create summary statistics in new data frame
- Example:

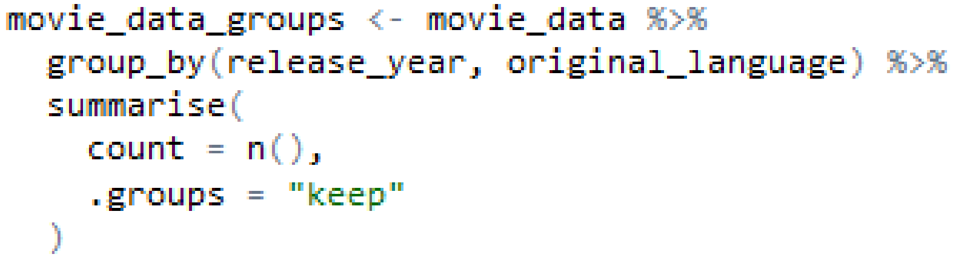
  -
    ```
    dataGrouped <- data %>%
        group_by(gender) %>%
        summarise(count = n())
    ```

Movie dataset example - Group & Summarise

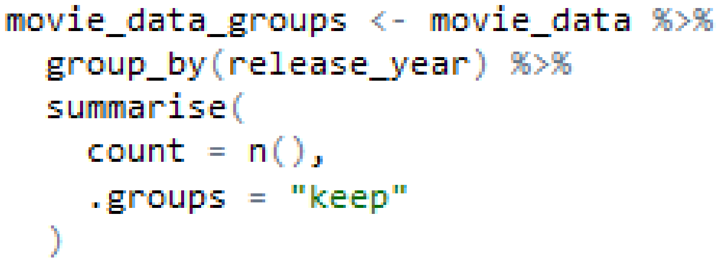


	release_year	original_language	count
1	2011	en	195
2	2016	en	191
3	2010	en	181
4	2013	en	180
5	2014	en	177
6	2006	en	176
7	2009	en	170
8	2015	en	164
9	2008	en	162
10	2005	en	159
11	2007	en	159
12	2012	en	156
13	2004	en	145
14	2002	en	136
15	2003	en	122
16	2001	en	121
17	1999	en	115
18	2000	en	110
19	1997	en	98
20	1998	en	95
21	1996	en	92
22	1995	en	82

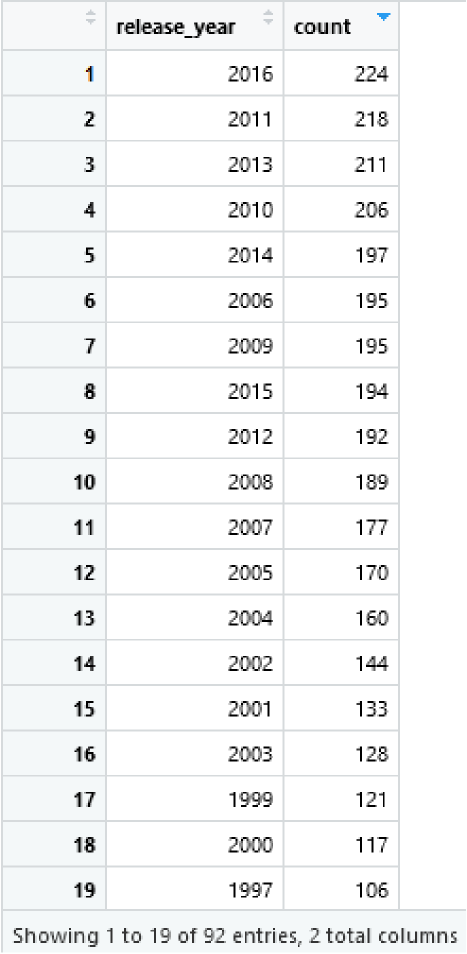
Showing 1 to 23 of 320 entries, 3 total columns



```
movie_data_groups <- movie_data %>%
  group_by(release_year, original_language) %>%
  summarise(
    count = n(),
    .groups = "keep"
  )
```



```
movie_data_groups <- movie_data %>%
  group_by(release_year) %>%
  summarise(
    count = n(),
    .groups = "keep"
  )
```

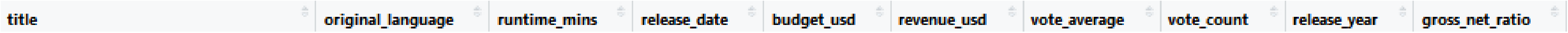


	release_year	count
1	2016	224
2	2011	218
3	2013	211
4	2010	206
5	2014	197
6	2006	195
7	2009	195
8	2015	194
9	2012	192
10	2008	189
11	2007	177
12	2005	170
13	2004	160
14	2002	144
15	2001	133
16	2003	128
17	1999	121
18	2000	117
19	1997	106

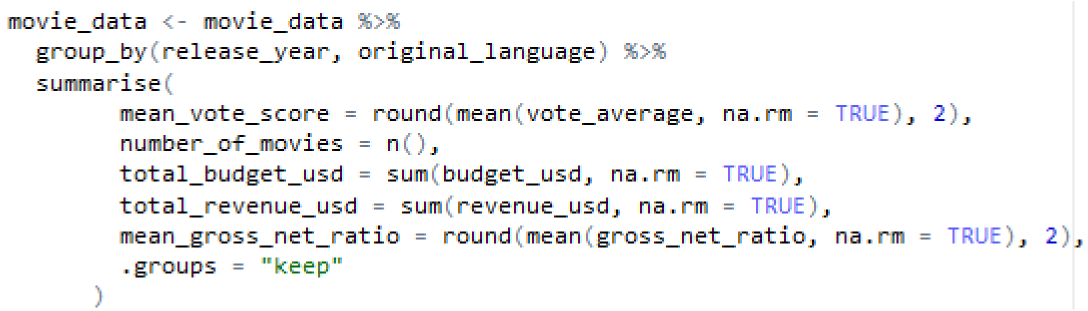
Showing 1 to 19 of 92 entries, 2 total columns

Movie dataset example - Group & Summarise

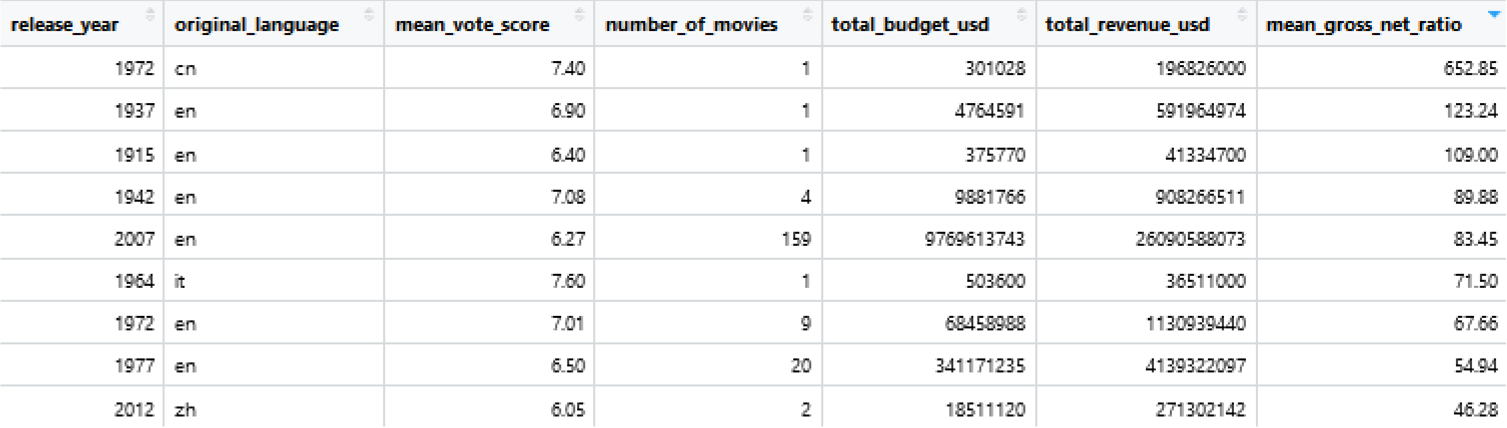
Column names before `group_by` and `summarise`:



title	original_language	runtime_mins	release_date	budget_usd	revenue_usd	vote_average	vote_count	release_year	gross_net_ratio



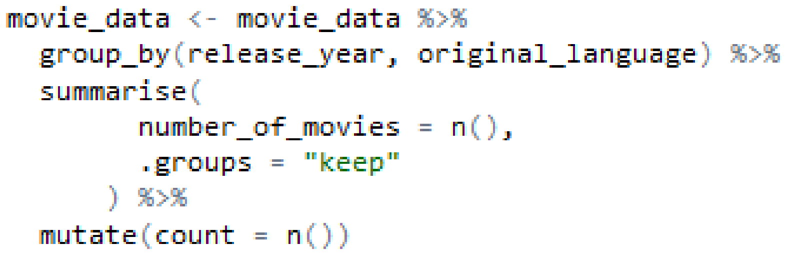
```
movie_data <- movie_data %>%
  group_by(release_year, original_language) %>%
  summarise(
        mean_vote_score = round(mean(vote_average, na.rm = TRUE), 2),
        number_of_movies = n(),
        total_budget_usd = sum(budget_usd, na.rm = TRUE),
        total_revenue_usd = sum(revenue_usd, na.rm = TRUE),
        mean_gross_net_ratio = round(mean(gross_net_ratio, na.rm = TRUE), 2),
        .groups = "keep"
      )
```



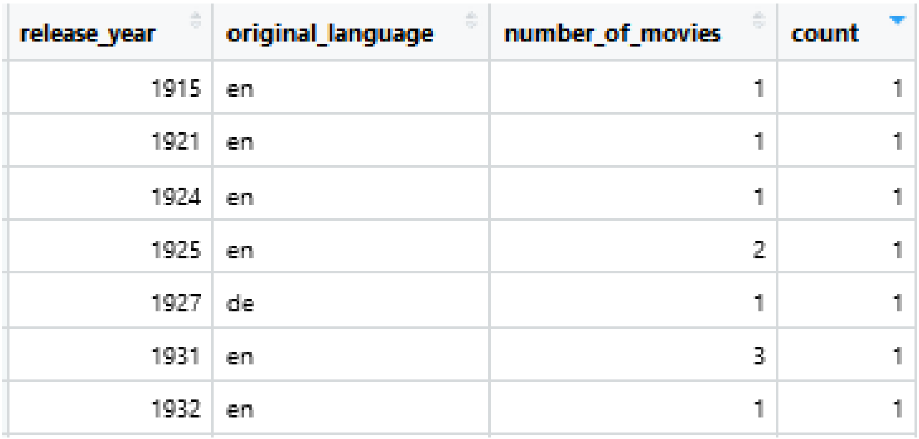
release_year	original_language	mean_vote_score	number_of_movies	total_budget_usd	total_revenue_usd	mean_gross_net_ratio
1972	cn	7.40	1	301028	196826000	652.85
1937	en	6.90	1	4764591	591964974	123.24
1915	en	6.40	1	375770	41334700	109.00
1942	en	7.08	4	9881766	908266511	89.88
2007	en	6.27	159	9769613743	26090588073	83.45
1964	it	7.60	1	503600	36511000	71.50
1972	en	7.01	9	68458988	1130939440	67.66
1977	en	6.50	20	341171235	4139322097	54.94
2012	zh	6.05	2	18511120	271302142	46.28

Movie dataset example - Group & Summarise

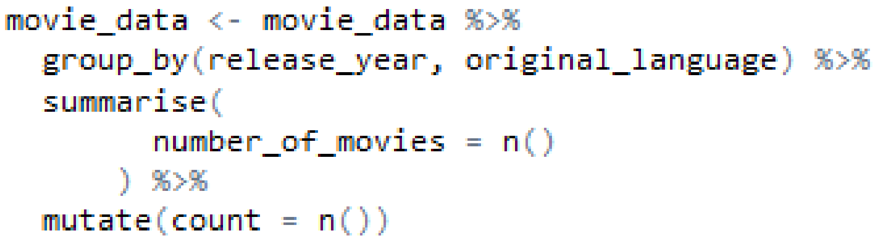
`.groups = “keep”` ensures that the `group_by` will remain after the `summarise`



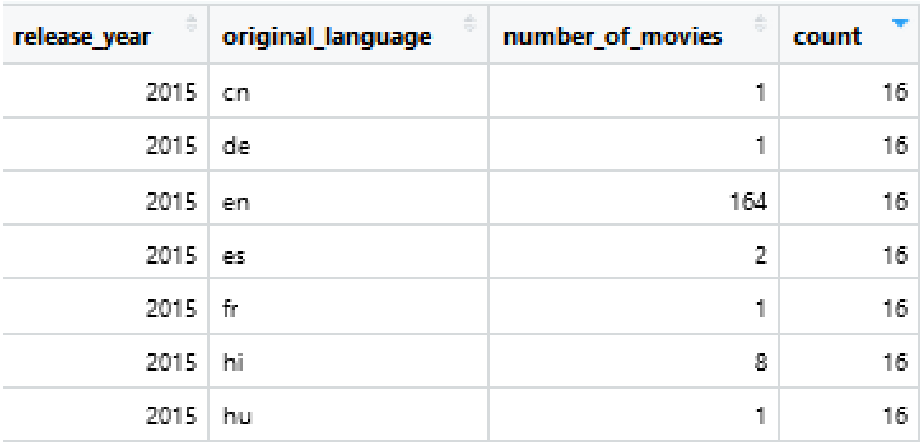
```
movie_data <- movie_data %>%
  group_by(release_year, original_language) %>%
  summarise(
        number_of_movies = n(),
        .groups = "keep"
      ) %>%
  mutate(count = n())
```



release_year	original_language	number_of_movies	count
1915	en	1	1
1921	en	1	1
1924	en	1	1
1925	en	2	1
1927	de	1	1
1931	en	3	1
1932	en	1	1

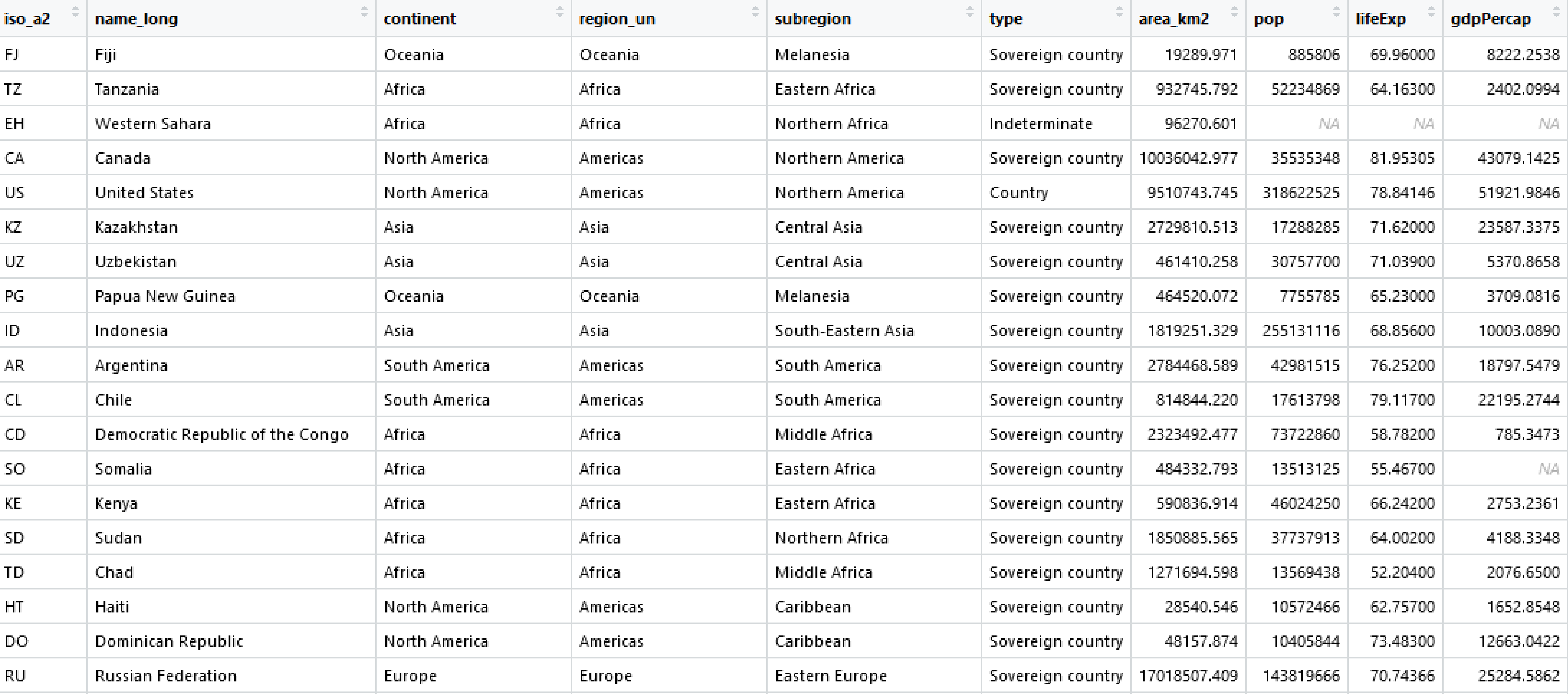


```
movie_data <- movie_data %>%
  group_by(release_year, original_language) %>%
  summarise(
        number_of_movies = n()
      ) %>%
  mutate(count = n())
```



release_year	original_language	number_of_movies	count
2015	cn	1	16
2015	de	1	16
2015	en	164	16
2015	es	2	16
2015	fr	1	16
2015	hi	8	16
2015	hu	1	16

About the exercise dataset



iso_a2	name_long	continent	region_un	subregion	type	area_km2	pop	lifeExp	gdpPercap
FJ	Fiji	Oceania	Oceania	Melanesia	Sovereign country	19289.971	885806	69.96000	8222.2538
TZ	Tanzania	Africa	Africa	Eastern Africa	Sovereign country	932745.792	52234869	64.16300	2402.0994
EH	Western Sahara	Africa	Africa	Northern Africa	Indeterminate	96270.601	NA	NA	NA
CA	Canada	North America	Americas	Northern America	Sovereign country	10036042.977	35535348	81.95305	43079.1425
US	United States	North America	Americas	Northern America	Country	9510743.745	318622525	78.84146	51921.9846
KZ	Kazakhstan	Asia	Asia	Central Asia	Sovereign country	2729810.513	17288285	71.62000	23587.3375
UZ	Uzbekistan	Asia	Asia	Central Asia	Sovereign country	461410.258	30757700	71.03900	5370.8658
PG	Papua New Guinea	Oceania	Oceania	Melanesia	Sovereign country	464520.072	7755785	65.23000	3709.0816
ID	Indonesia	Asia	Asia	South-Eastern Asia	Sovereign country	1819251.329	255131116	68.85600	10003.0890
AR	Argentina	South America	Americas	South America	Sovereign country	2784468.589	42981515	76.25200	18797.5479
CL	Chile	South America	Americas	South America	Sovereign country	814844.220	17613798	79.11700	22195.2744
CD	Democratic Republic of the Congo	Africa	Africa	Middle Africa	Sovereign country	2323492.477	73722860	58.78200	785.3473
SO	Somalia	Africa	Africa	Eastern Africa	Sovereign country	484332.793	13513125	55.46700	NA
KE	Kenya	Africa	Africa	Eastern Africa	Sovereign country	590836.914	46024250	66.24200	2753.2361
SD	Sudan	Africa	Africa	Northern Africa	Sovereign country	1850885.565	37737913	64.00200	4188.3348
TD	Chad	Africa	Africa	Middle Africa	Sovereign country	1271694.598	13569438	52.20400	2076.6500
HT	Haiti	North America	Americas	Caribbean	Sovereign country	28540.546	10572466	62.75700	1652.8548
DO	Dominican Republic	North America	Americas	Caribbean	Sovereign country	48157.874	10405844	73.48300	12663.0422
RU	Russian Federation	Europe	Europe	Eastern Europe	Sovereign country	17018507.409	143819666	70.74366	25284.5862

About the exercise dataset

- For the exercise we are going to be working with a world dataset containing the following columns:
  - `iso_a2` – jurisdiction code (string)
  - `name_long` – jurisdiction name (string)
  - `continent` – continent (string)
  - `region_un` – region (string)
  - `subregion` – subregion (string)
  - `type` – the type of jurisdiction (string)
  - `area_km2` – area in km squared (double)
  - `pop` – population (integer)
  - `lifeExp` – life expectancy (double)
  - `gdpPercap` – GDP per capita (double)

Manipulating data with Shiny

Using /stage2, inside a new global.R file or in a reactive inside server.R:

- Remove the iso_a2 column from the data
- Filter our data to keep only the records which have a value provided for the Population (pop) column.
- Create a new column called ‘Status’. This column should have the value “Complete” if all of Population, Life Expectancy and GDP Per Capita are provided. Otherwise, this should have the value “Incomplete”)
- Create a new column for Population Density per Sq. Km (pop / area_km2)
- Create a sliderInput and use this to filter the data based on either Area (Sq. Km) ***OR*** Population Density per Sq. Km
  - What should we use as our default start / end values?
- Create selectInputs for Continent and Status, use these to filter the data.
  - Hint: what choices should be included in these inputs, and where could we retrieve these from?

Exporting data

Getting data back out of the application

Some common functions:

- For CSV files: write.csv(data, “filepath”)
- For RDS files: saveRDS(data, “filepath”)

To link this up to R Shiny:

- downloadLink(outputId = “dl”, label = “download”) [*in ui.R*]
- downloadHandler(filename = function() {}, content = function() {}) [in server.R]

Exporting data

Add:

To *ui.R*: downloadLink(outputId = “download”, label = “Download”)

To *server.R*:

```
output$download <- downloadHandler(
          filename <- function() {
                    …
          },
          content = function(file) {
                    write.csv(data, file)
          })
```

Next time

- Case study: AIS Explorer
- Spatial visualisation with Leaflet

Challenge - using the /result folder as a template if needed:

- Add a fileInput to ui.R so that you can upload local files to your app.
- Use this fileInput with *world_data.csv,* create a reactive that uses read.csv to read the data *(Ex: read.csv(input$uploadFile$datapath))*
- Create a reactive to group and summarise this data to create a new data frame and table, where there is one record for each combination of Region, Subregion, Type, average Population Density and average Life Expectancy. Put this table into a new tabPanel in the UI
- Create a downloadHandler for this new table which includes the region / subregion / type in the file name
- Any other changes you can think of?