THE OHIO STATE UNIVERSITY
FISHER COLLEGE OF BUSINESS

BUSMGT 7331: Descriptive Analytics and Visualization
Week 1

Data Wrangling and Descriptive Statistics

Hyunwoo Park
Fisher College of Business
The Ohio State University

Course Overview

Before we begin

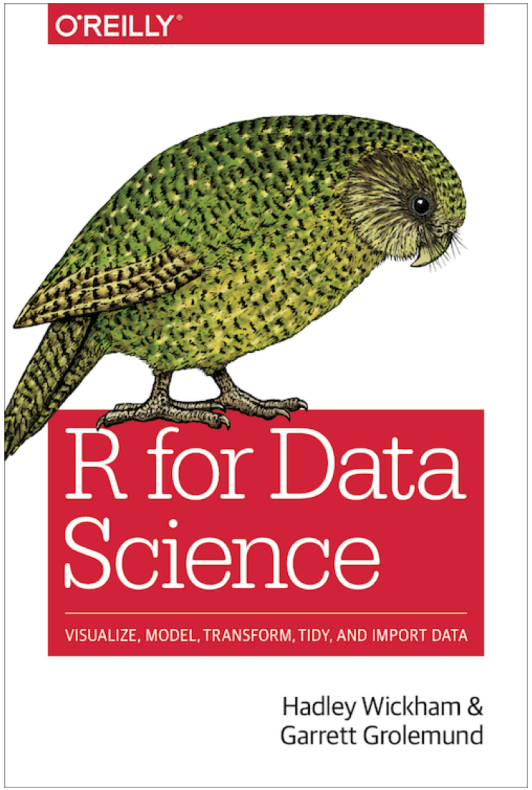
- Read the syllabus.
- Teaching philosophy
  - I will try to make this course relevant to you.
  - I should teach what I believe is actually good for you.
  - I believe in learning by doing, especially for coding and programming.
  - You should learn from my perspective and organization of thoughts, not just crystalized knowledge.
- About me
  - Assistant Professor in Management Sciences
    Fisher College of Business
    The Ohio State University
    Affiliated with Translational Data Analytics Institute
  - Email: park.2706@osu.edu
    Office: 632 Fisher Hall
    Office Hours: probably through WebEx and by appointment
  - For more information about me, visit http://hyunwoopark.com.

My assumptions on what you learned from prerequisites

- You know how to navigate Carmen.
- You have R Studio installed.
- You know how to open R Studio, load some data, and run R scripts.
- You have experience using tidyverse.
- You created some plots using ggplot2 with tidy data.
- You know how to compose a report with R Markdown.

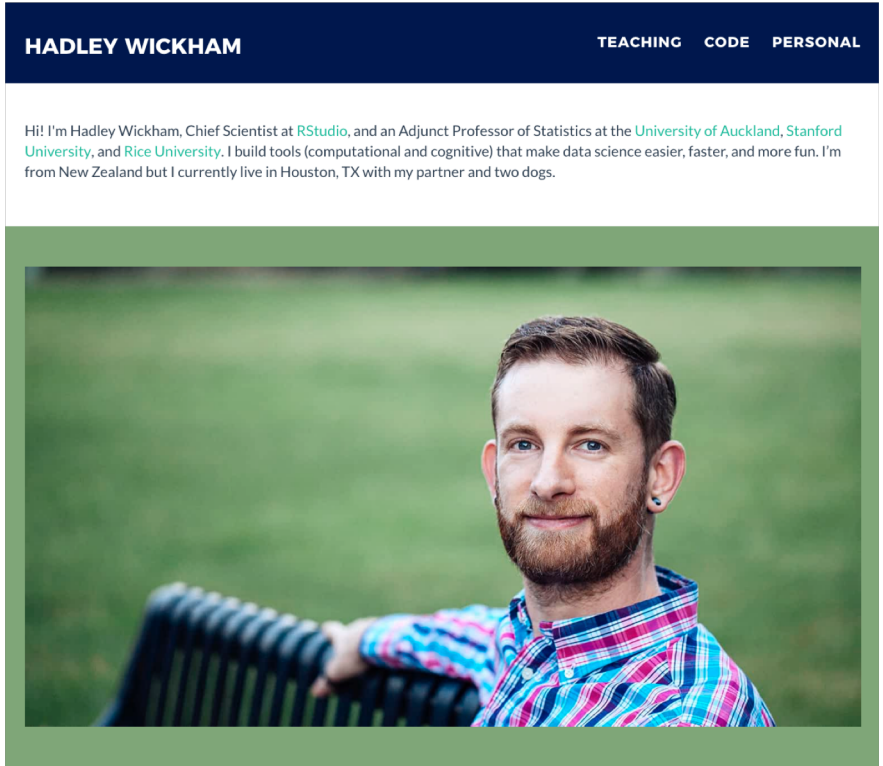
Textbook for first two weeks

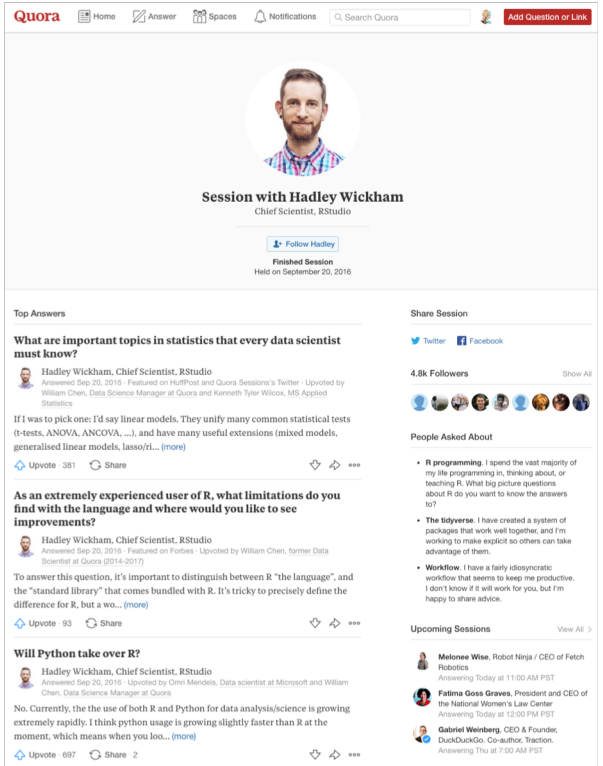
- R for Data Science
- Available free online: https://r4ds.had.co.nz/
- I will assign relevant chapters for reading.
- This book will be abbreviated as “R4DS” hereinafter.



Hadley Wickham

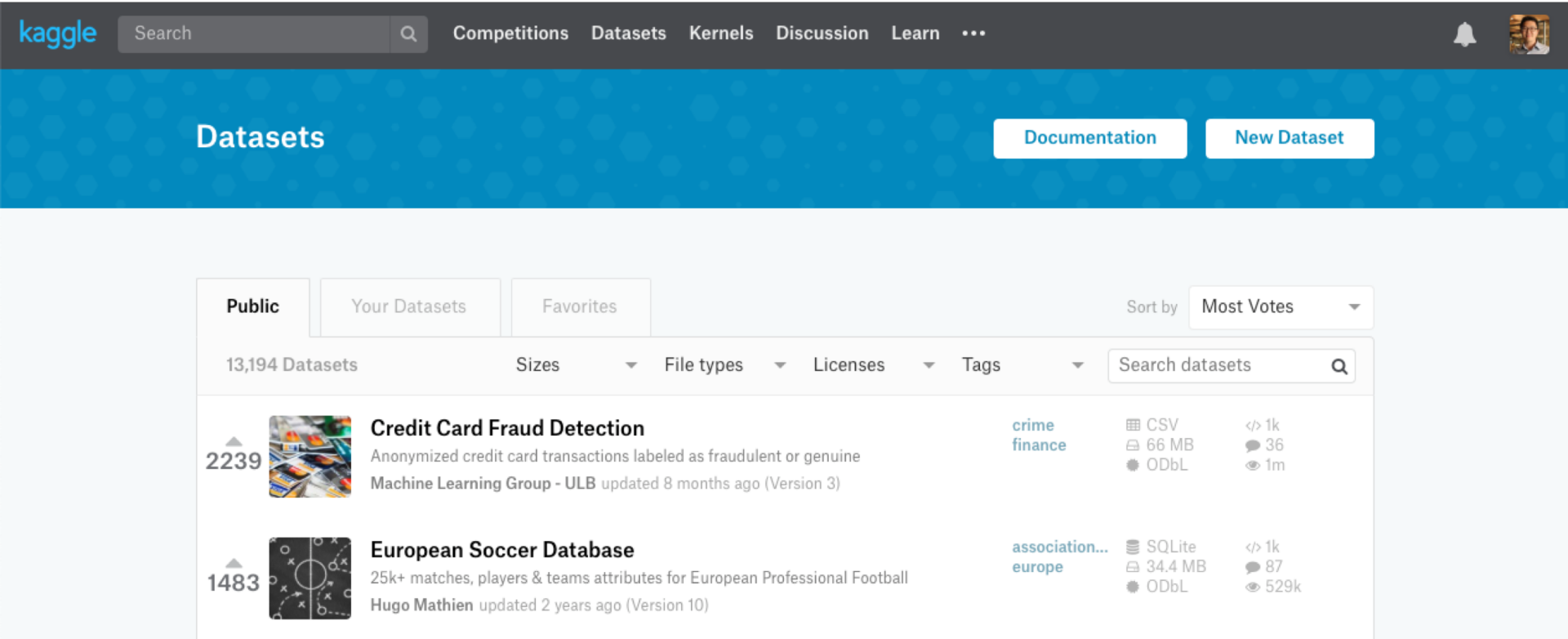
- https://www.quora.com/session/Hadley-Wickham/1





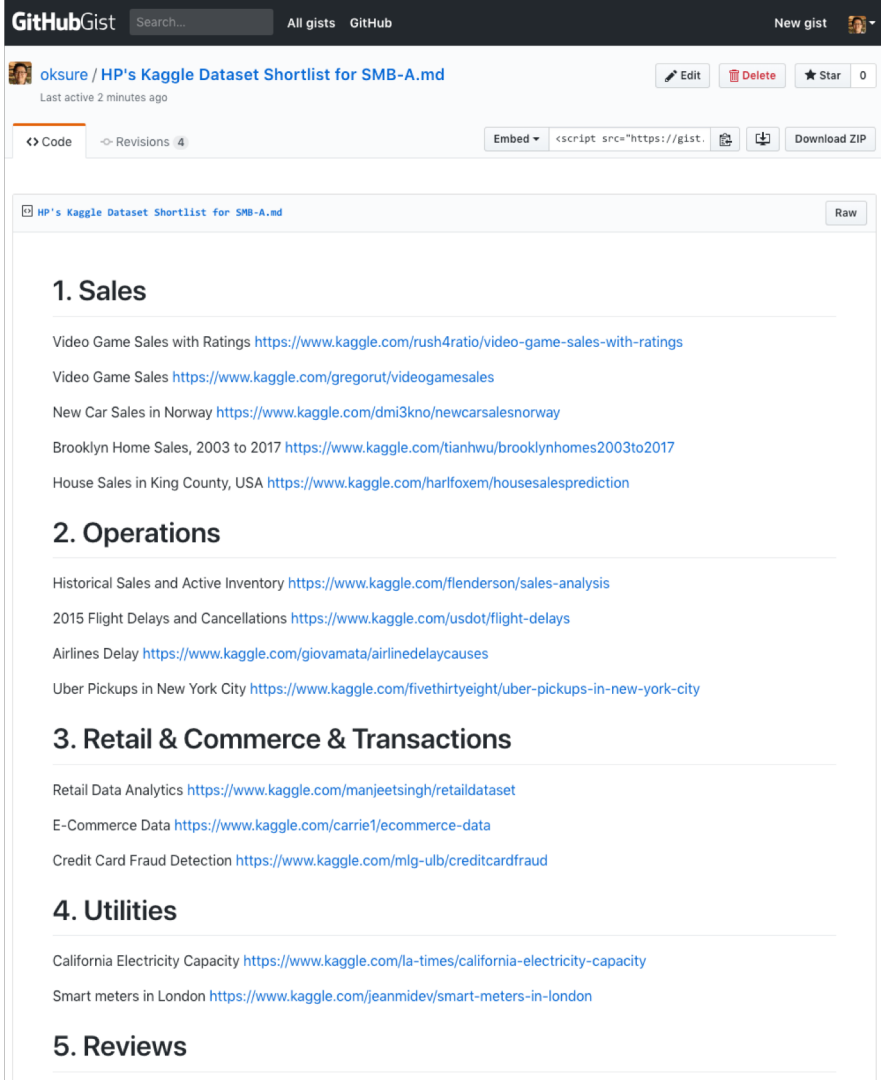
Datasets for this course

- Kaggle (https://www.kaggle.com/)



My shortlist datasets

- http://bit.ly/hp-kaggle-datasets
- I browsed public Kaggle datasets and handpicked a set of datasets that might be of your interest in business analytics.



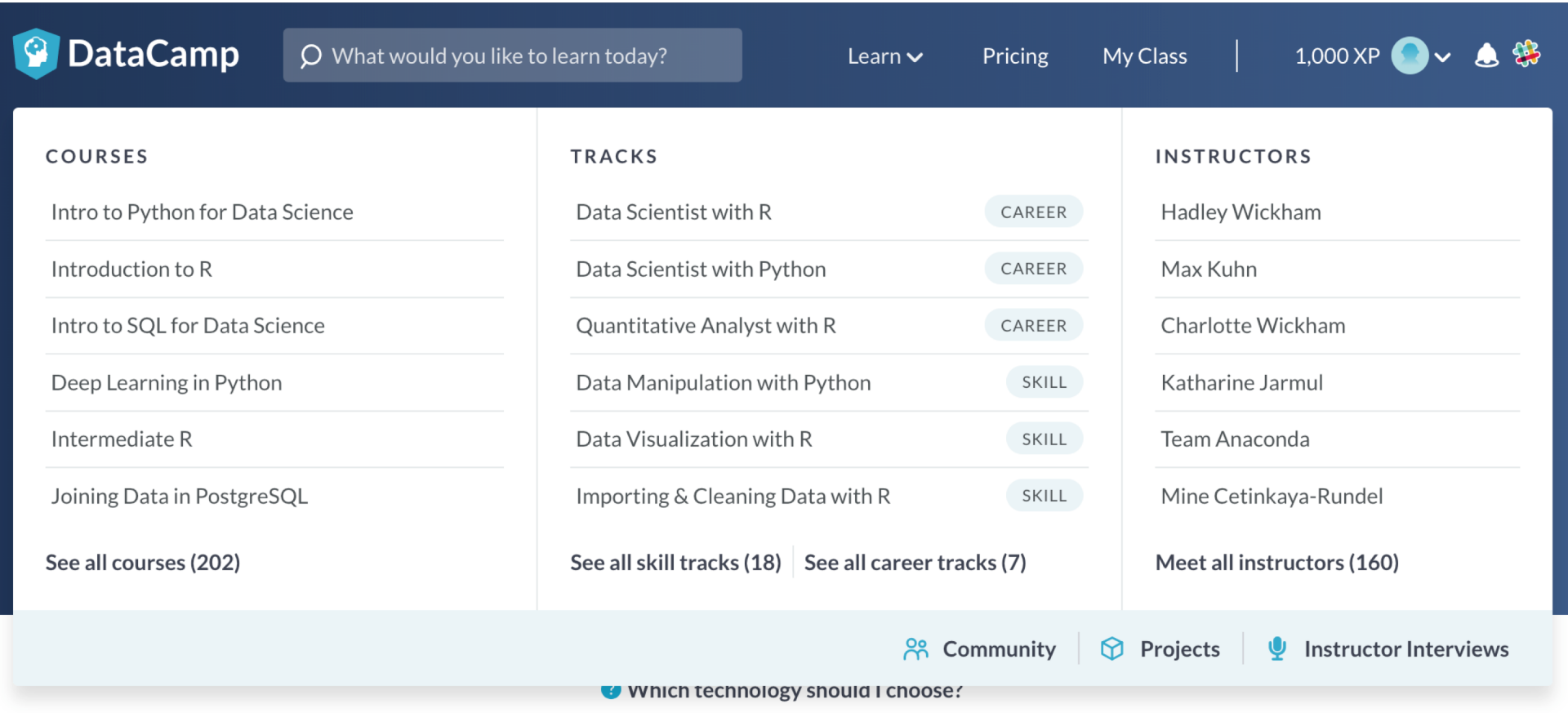
Motivation

- It’s good time to learn R.
- https://www.datacamp.com/community/tutorials/r-or-python-for-data-analysis
- https://www.kdnuggets.com/2017/09/python-vs-r-data-science-machine-learning.html

Course objectives in plain English

- Understand how to approach a dataset you don't know about.
- Compute some common statistics that you could reasonably expect other people would also know.
- Examine relationships among variables.
- Communicate these numbers effectively and efficiently using visualization.
- Compose visualizations and empower your boss and colleagues using interactivity.
- Gain experience in creating advanced visualizations and understanding non-rectangular data.

DataCamp

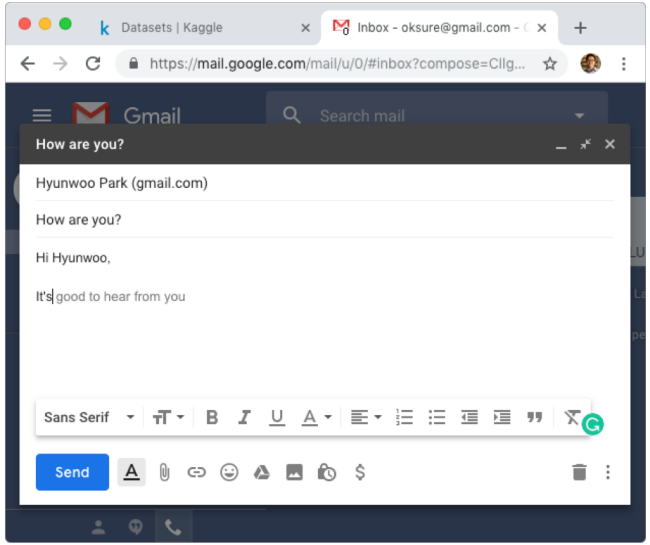

DataCamp
What would you like to learn today?
Learn
Pricing
My Class
1,000 XP
COURSES
Intro to Python for Data Science
Introduction to R
Intro to SQL for Data Science
Deep Learning in Python
Intermediate R
Joining Data in PostgreSQL
See all courses (202)
TRACKS
Data Scientist with R
CAREER
Data Scientist with Python
CAREER
Quantitative Analyst with R
CAREER
Data Manipulation with Python
SKILL
Data Visualization with R
SKILL
Importing & Cleaning Data with R
SKILL
See all skill tracks (18)
See all career tracks (7)
INSTRUCTORS
Hadley Wickham
Max Kuhn
Charlotte Wickham
Katharine Jarmul
Team Anaconda
Mine Cetinkaya-Rundel
Meet all instructors (160)
Community
Projects
Instructor Interviews
Which technology should I choose?

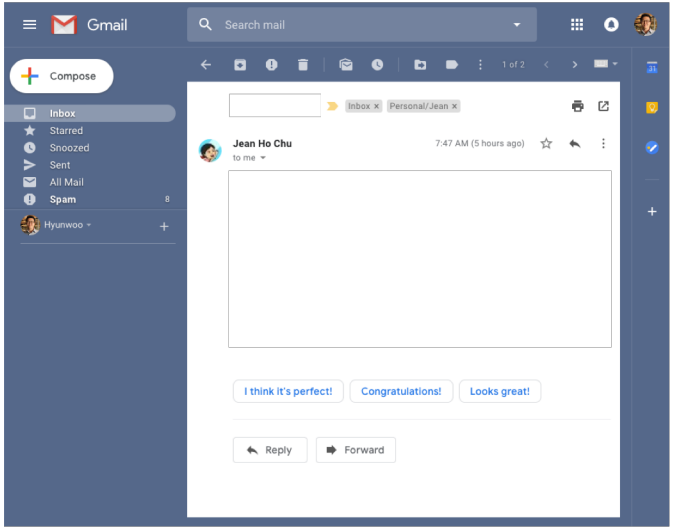
DataCamp tracks and courses for BM 7331

- Data Visualization with R
  - [1] Data Visualization with ggplot2 (Part 1)
  - [2] Data Visualization with ggplot2 (Part 2)
  - [3] Visualization Best Practices in R
- Text Mining with R
  - [1] String Manipulation in R with stringr
  - [3] Sentiment Analysis in R: The Tidy Way
  - [4] Sentiment Analysis in R
- Spatial Data with R
  - [1] Working with Geospatial Data in R
  - [4] Interactive Maps with leaflet in R
- Shiny Fundamentals with R
  - [3] Building Dashboards with shinydashboard
  - [4] Building Dashboards with flexdashboard
- Individual Courses
  - Categorical Data in Tidyverse
  - Working with Dates and Times in R
  - Correlation and Regression
  - Cluster Analysis in R

Types of analytics

- Descriptive Analytics
- Predictive Analytics
- Prescriptive Analytics





Why visualization?

Example from [Rahul Basole (Georgia Tech)]

- How many 7's are there?

6 1 0 1 5 6 3 8 3 5 3 8 3 2 3 1 8 1 9 1 0 2 1 6 3 3 8 1 6 3 2 9 3 1 8 9 5 0 5 1
4 9 8 9 4 6 5 5 9 5 9 8 5 1 9 3 0 6 1 8 4 0 0 1 4 9 8 3 1 5 9 4 5 3 3 9 3 1 0 2
4 6 1 5 6 3 0 3 6 9 3 7 9 4 8 2 4 8 3 3 4 4 8 6 9 8 0 0 6 0 5 2 1 2 3 9 8 4 1 5
3 3 5 1 5 4 0 1 9 3 3 1 5 7 3 1 9 8 1 5 3 6 9 2 4 2 1 6 5 8 2 5 7 1 5 0 5 6 4 8
9 0 5 9 3 9 4 1 2 2 3 5 2 9 5 9 9 2 3 2 3 6 9 4 2 3 9 0 2 5 3 4 0 8 4 3 5 8 9 8
4 8 1 6 2 6 3 2 3 3 3 3 4 1 9 9 2 2 5 6 4 3 3 2 2 1 9 6 6 4 0 3 9 0 1 2 0 0 2 9
6 3 0 3 2 3 6 0 3 1 2 6 6 3 5 8 2 3 3 3 5 8 0 8 9 9 4 1 2 7 8 3 3 5 6 8 3 4 6 3
4 9 9 0 6 4 3 4 4 5 5 3 0 0 5 3 0 3 5 7 0 6 1 8 0 1 0 0 6 1 2 2 9 8 8 6 6 2 3 6
9 8 1 1 3 5 6 3 8 5 9 4 9 4 2 3 3 1 3 2 6 1 3 6 2 6 8 0 9 3 5 9 8 1 0 4 9 2 9 1
1 0 5 2 2 4 2 0 9 0 3 8 0 3 6 3 3 2 5 4 9 1 4 1 1 4 1 5 8 5 5 3 8 2 3 6 2 2 3 9

Power of visualization

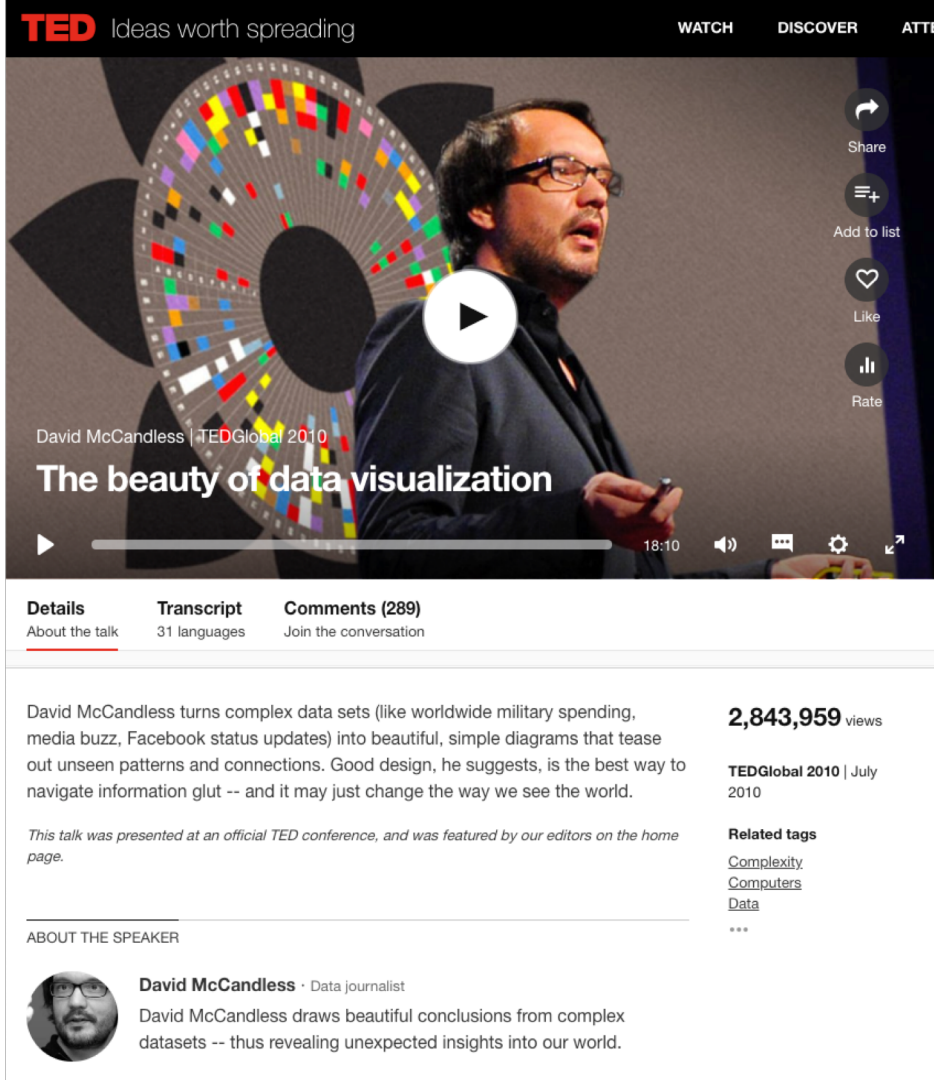
Example from [Rahul Basole (Georgia Tech)]

- How about now?

Power of human vision

- Unparalleled information processing capacity compared to other senses

- https://www.ted.com/talks/david_mccandless_the_beauty_of_data_visualization or https://youtu.be/pLqjQ55tz-U
- An article about his talk: https://www.theguardian.com/science/punctuated-equilibrium/2010/dec/30/2



Trivia about the trivia

- I created these random numbers using R.

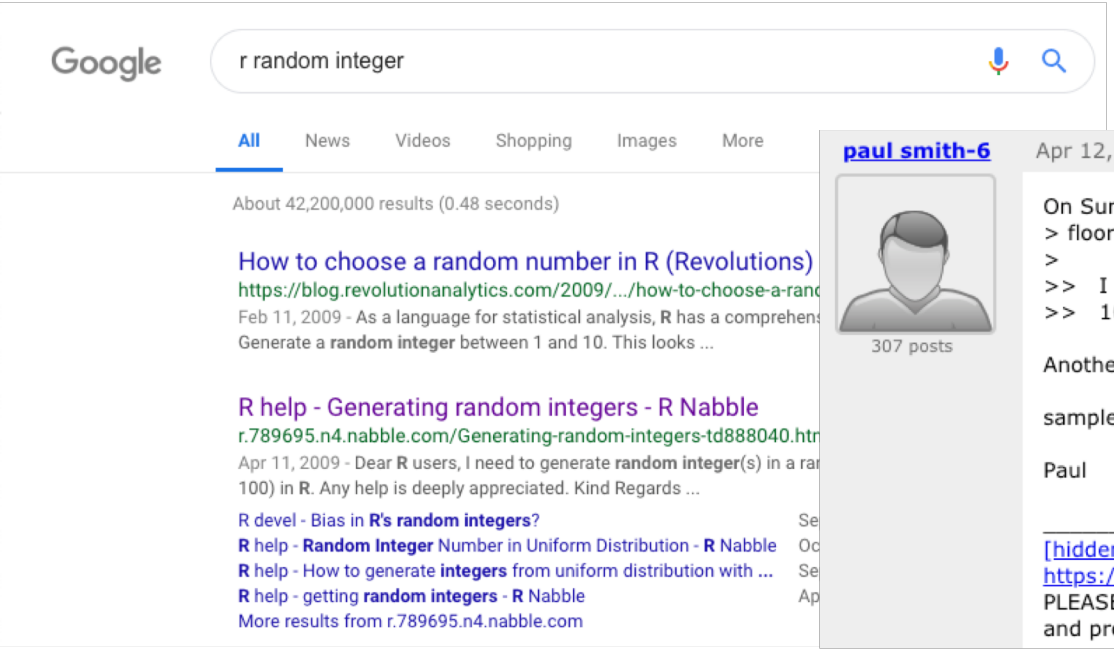
```
1   prmatrix(matrix(sample(0:9,400,replace=T),nrow=10),
      rowlab=rep("",10),collab=rep("",40))
```

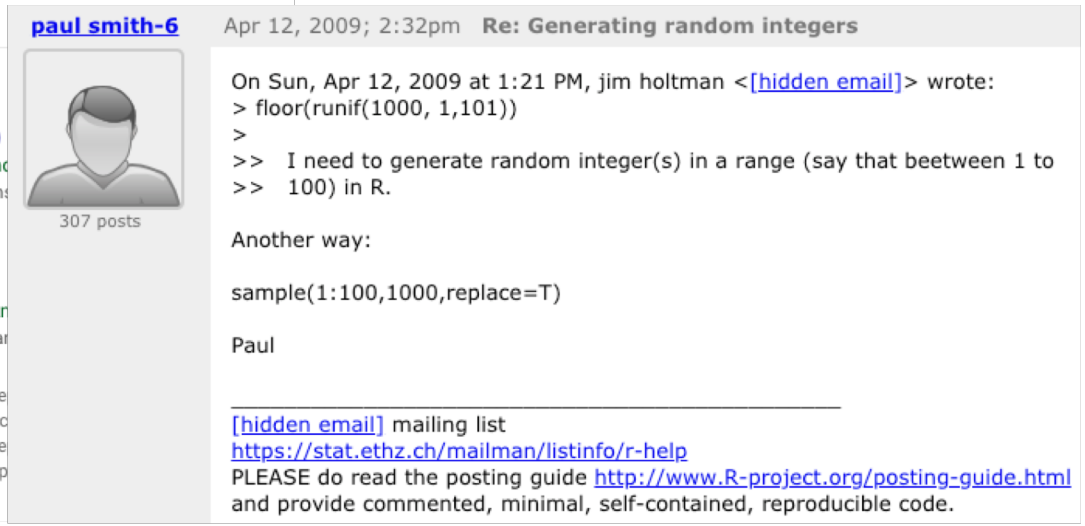
```
> prmatrix(matrix(sample(0:9,400,replace=T),nrow=10),rowlab=rep("",10),collab=rep("",40))

 6 1 0 1 5 6 3 8 7 5 3 8 3 2 3 1 8 1 9 1 0 2 1 6 3 3 8 1 6 3 2 9 7 1 8 9 5 0 5 1
 4 9 8 9 4 6 5 5 9 5 9 8 5 1 9 7 0 6 1 8 4 0 0 1 4 9 8 7 1 5 9 4 5 7 7 9 7 1 0 2
 4 6 1 5 6 7 0 3 6 9 3 7 9 4 8 2 4 8 3 7 4 4 8 6 9 8 0 0 6 0 5 2 1 2 3 9 8 4 1 5
 3 7 5 1 5 4 0 1 9 7 3 1 5 7 3 1 9 8 1 5 3 6 9 2 4 2 1 6 5 8 2 5 7 1 5 0 5 6 4 8
 9 0 5 9 3 9 4 1 2 2 3 5 2 9 5 9 9 2 3 2 3 6 9 4 2 7 9 0 2 5 3 4 0 8 4 3 5 8 9 8
 4 8 1 6 2 6 3 2 3 7 3 7 4 1 9 9 2 2 5 6 4 7 3 2 2 1 9 6 6 4 0 3 9 0 1 2 0 0 2 9
 6 3 0 3 2 3 6 0 7 1 2 6 6 7 5 8 2 7 7 7 5 8 0 8 9 9 4 1 2 7 8 3 7 5 6 8 7 4 6 3
 4 9 9 0 6 4 7 4 4 5 5 3 0 0 5 3 0 3 5 7 0 6 1 8 0 1 0 0 6 1 2 2 9 8 8 6 6 2 7 6
 9 8 1 1 7 5 6 3 8 5 9 4 9 4 2 7 7 1 7 2 6 1 3 6 2 6 8 0 9 3 5 9 8 1 0 4 9 2 9 1
 1 0 5 2 2 4 2 0 9 0 7 8 0 7 6 7 7 2 5 4 9 1 4 1 1 4 1 5 8 5 5 7 8 2 3 6 2 2 3 9
```

How did I figure this out?

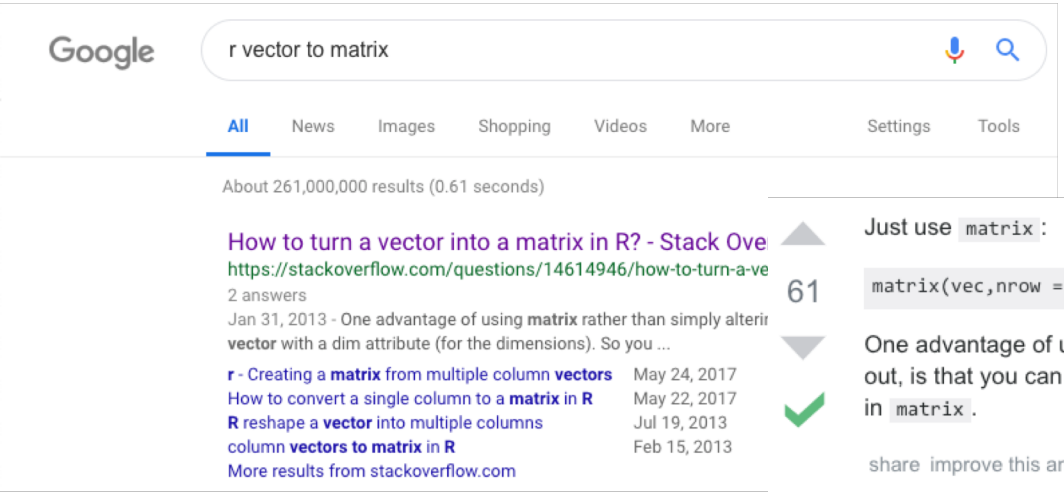
- I didn't know how to do it from the beginning. So I googled.
- http://r.789695.n4.nabble.com/Generating-random-integers-td888040.html

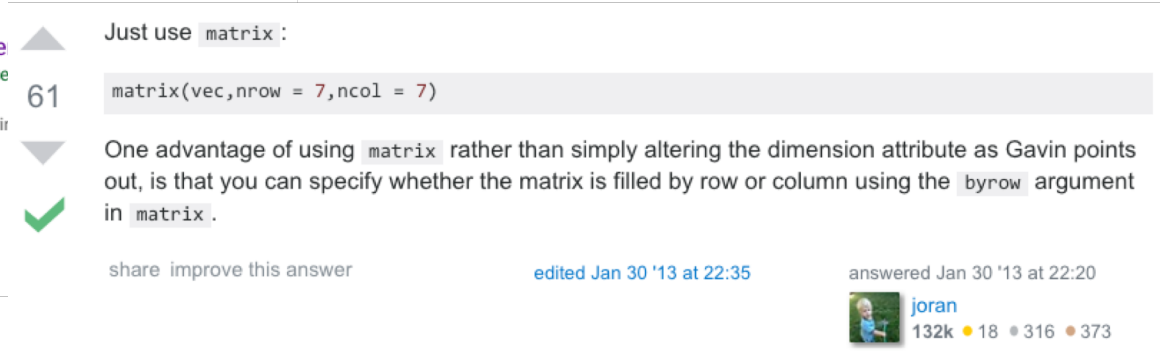




Not quite there yet

- What I wanted is 10x40 matrix of random integers between 0 and 9.
- https://stackoverflow.com/questions/14614946/how-to-turn-a-vector-into-a-matrix-in-r

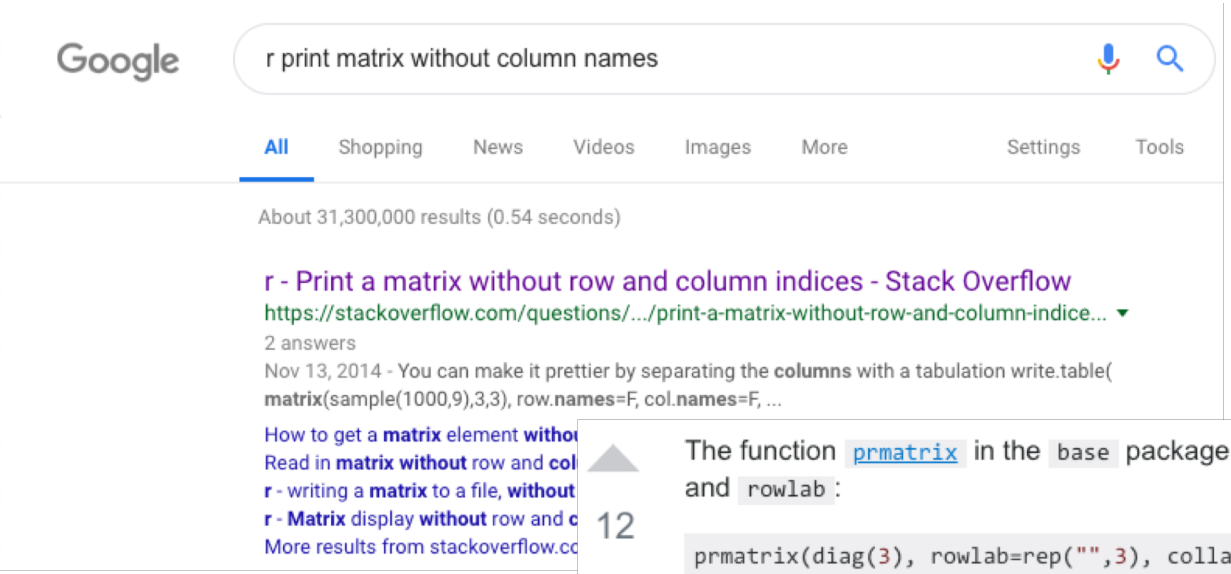




Urgh… I don't want these row and column labels.

```
> matrix(sample(0:9,400,replace=T),nrow=10)
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24] [,25]
 [1,]    2    8    4    8    7    4    0    3    0     1     5     7     0     7     6     0     8     3     9     8     4     1     9     3     1
 [2,]    8    7    6    3    3    5    7    6    6     2     7     7     4     0     0     2     8     7     4     6     2     0     4     0     0
 [3,]    1    9    5    6    5    8    8    7    6     6     0     9     5     2     0     4     4     1     7     2     3     5     2     2     6
 [4,]    4    0    2    0    4    2    5    9    2     0     0     9     9     7     1     0     8     9     0     6     4     9     5     5     7
 [5,]    2    9    8    1    5    8    7    0    4     9     3     1     0     4     8     5     2     7     0     6     8     7     4     1     5
 [6,]    7    3    2    8    5    1    6    1    2     5     7     5     0     2     5     2     7     7     7     4     5     0     4     9     8
 [7,]    7    5    7    8    9    4    9    0    1     0     0     6     5     6     6     0     5     3     5     1     2     7     4     1     3
 [8,]    4    5    3    7    9    8    9    4    0     2     0     6     7     6     1     3     9     1     8     1     0     0     7     2     4
 [9,]    2    0    6    7    0    9    0    0    0     7     1     4     3     7     2     8     0     1     9     7     0     6     3     9     7
[10,]    4    9    6    9    8    9    6    6    3     6     2     8     5     5     1     1     7     9     5     2     2     7     5     0     5
      [,26] [,27] [,28] [,29] [,30] [,31] [,32] [,33] [,34] [,35] [,36] [,37] [,38] [,39] [,40]
 [1,]     2     5     3     9     6     3     5     2     0     1     5     2     5     6     1
 [2,]     7     6     7     1     6     0     0     6     7     4     4     2     6     9     0
 [3,]     3     8     4     6     3     7     2     5     7     3     6     1     3     3     8
 [4,]     5     4     6     7     2     0     9     5     3     5     2     7     3     1     5
 [5,]     3     9     9     0     8     9     5     5     7     5     9     2     8     8     8
 [6,]     2     7     6     2     9     6     7     4     6     9     1     1     0     2     0
 [7,]     0     1     3     3     0     4     7     0     4     2     6     7     4     5     0
 [8,]     3     0     4     2     1     8     4     9     7     8     8     5     8     5     1
 [9,]     3     9     0     4     1     0     5     4     7     2     7     2     1     1     9
[10,]     0     4     7     8     0     0     3     0     3     3     9     1     3     7     0
```

Finally!



12

The function `prmatrix` in the `base` package could work for this, it can take the arguments `collab` and `rowlab`:

```
prmatrix(diag(3), rowlab=rep("",3), collab=rep("",3))

 1 0 0
 0 1 0
 0 0 1
```

share improve this answer

edited Nov 13 '14 at 11:14

Richie Cotton
78.3k ● 27 ● 184 ● 304

answered Nov 13 '14 at 10:33

user1981275
8,067 ● 5 ● 44 ● 73

Metacognition

- Meaning
  - Thinking about own thinking
  - Knowing about own knowing
  - Awareness of one's awareness
- Metacognitive knowledge
  - (1) Content knowledge (declarative knowledge): Knowing what you know and what you don't know
  - (2) Task knowledge (procedural knowledge): Knowing the procedure to find a solution for the problem
  - (3) Strategic knowledge (conditional knowledge): Knowing what to do to learn how to solve the problem
- Articulating what you do NOT know is paramount
  in accelerating self-learning in the era of Googling everything.

Key Takeaway

- Knowing what to search for is really really important.
- Google is your friend. Stack Overflow is usually a Google's go to friend.
- Use R help as well. Try out the following on your own.

```
?sample
0:9
sample(0:9,10,replace=T)
sample(0:9,10,replace=F)
matrix(0:9,nrow=2)
matrix(0:9,nrow=5)
```

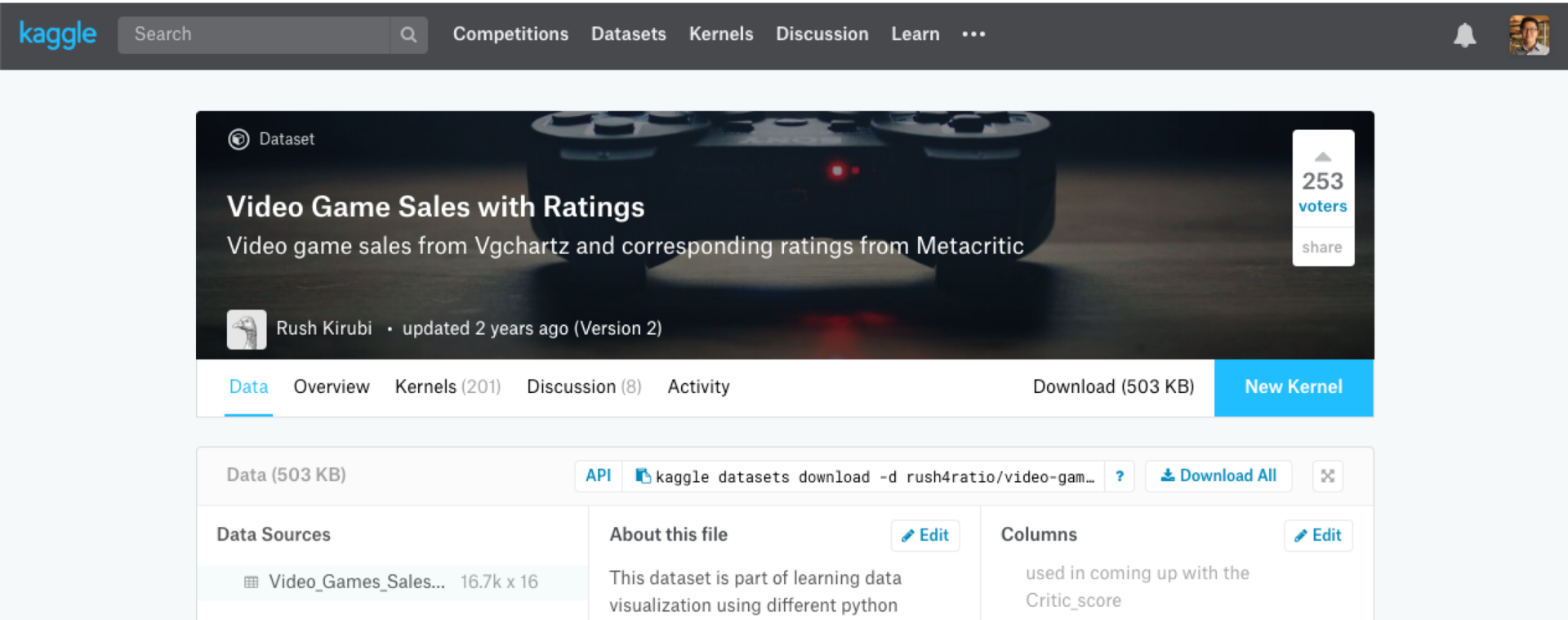
Data Wrangling

Reading

- R4DS Chapter 7. Tibbles with tibble
- R4DS Chapter 8. Data Import with readr
- R4DS Chapter 9. Tidy Data with tidyr
- R4DS Chapter 3. Data Transformation with dplyr

Let's grab a dataset.

- Video Game Sales with Ratings
  https://www.kaggle.com/rush4ratio/video-game-sales-with-ratings



A typical data science project & data wrangling

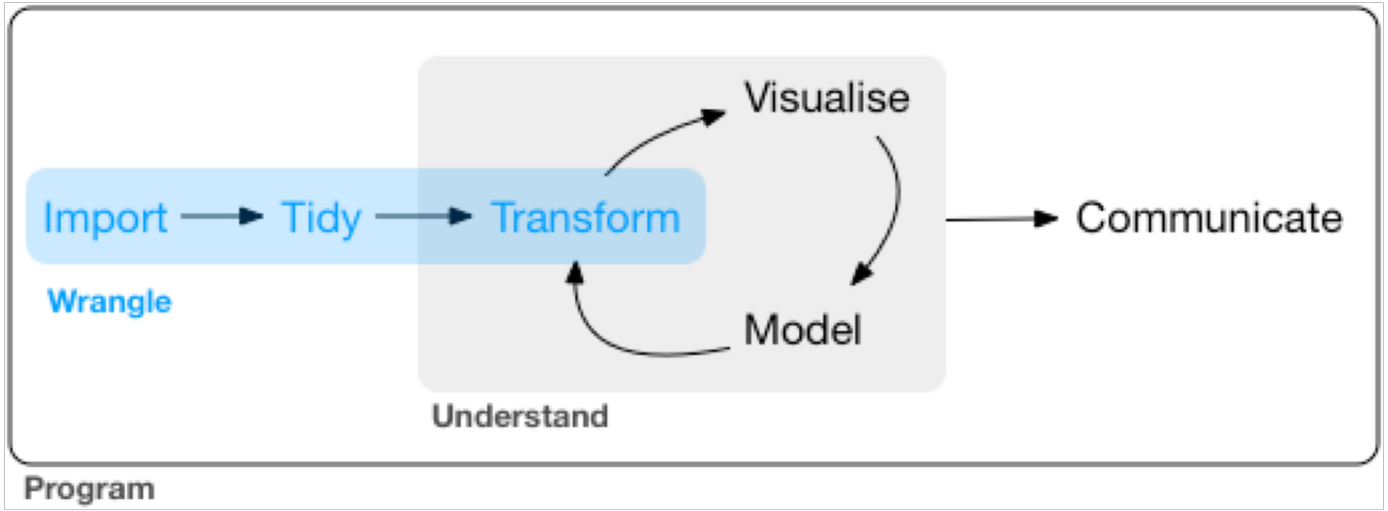


Figure from R4DS, p. 117 (https://r4ds.had.co.nz/wrangle-intro.html)

Let's import the data.

```
1  library(tidyverse)
2  df <- read_csv("data/Video_Games_Sales_as_at_22_Dec_2016.csv")
3  df
4  is_tibble(df)
```

```
> df <- read_csv("data/Video_Games_Sales_as_at_22_Dec_2016.csv")
Parsed with column specification:
cols(
  Name = col_character(),
  Platform = col_character(),
  Year_of_Release = col_character(),
  Genre = col_character(),
  Publisher = col_character(),
  NA_Sales = col_double(),
  EU_Sales = col_double(),
```

```
> is_tibble(df)
[1] TRUE
```

```
> df
# A tibble: 16,719 x 16
   Name          Platform Year_of_Release Genre   Publisher NA_Sales EU_Sales JP_Sales Other_Sales Global_Sales Critic_Score Critic_Count User_Score User_Count Developer Rating
   <chr>         <chr>    <chr>          <chr>   <chr>       <dbl>    <dbl>    <dbl>       <dbl>        <dbl>        <int>        <int> <chr>         <int> <chr>     <chr>
 1 Wii Sports    Wii      2006           Sports  Nintendo     41.4     29.0     3.77        8.45         82.5           76           51 8              322 Nintendo  E
 2 Super Mario … NES      1985           Platfo… Nintendo     29.1      3.58    6.81        0.77         40.2           NA           NA NA              NA NA        NA
 3 Mario Kart W… Wii      2008           Racing  Nintendo     15.7     12.8     3.79        3.29         35.5           82           73 8.3            709 Nintendo  E
 4 Wii Sports R… Wii      2009           Sports  Nintendo     15.6     10.9     3.28        2.95         32.8           80           73 8              192 Nintendo  E
 5 Pokemon Red/… GB       1996           Role-P… Nintendo     11.3      8.89   10.2         1            31.4           NA           NA NA              NA NA        NA
 6 Tetris        GB       1989           Puzzle  Nintendo     23.2      2.26    4.22        0.580        30.3           NA           NA NA              NA NA        NA
 7 New Super Ma… DS       2006           Platfo… Nintendo     11.3      9.14    6.5         2.88         29.8           89           65 8.5            431 Nintendo  E
 8 Wii Play      Wii      2006           Misc    Nintendo     14.0      9.18    2.93        2.84         28.9           58           41 6.6            129 Nintendo  E
 9 New Super Ma… Wii      2009           Platfo… Nintendo     14.4      6.94    4.7         2.24         28.3           87           80 8.4            594 Nintendo  E
10 Duck Hunt     NES      1984           Shooter Nintendo     26.9      0.63    0.28        0.47         28.3           NA           NA NA              NA NA        NA
# ... with 16,709 more rows
```

What is tibble?

- Base R’s data frame has a few shortcomings that make people overlook early mistakes that leave them confused.
- Basically, data frame does some unasked things that turn out to be quite confusing.
- Tibble is more explicit and straightforward.
- Read more about the advantage of tibble over data.frame here. https://tibble.tidyverse.org/

What is tidy data?

- It depends on the data context.
- Variables are easier to be linked; observations are harder.
- Further explanation on tidy data:
  Wickham, H. (2014). Tidy data. Journal of Statistical Software, 59.
  http://vita.had.co.nz/papers/tidy-data.pdf

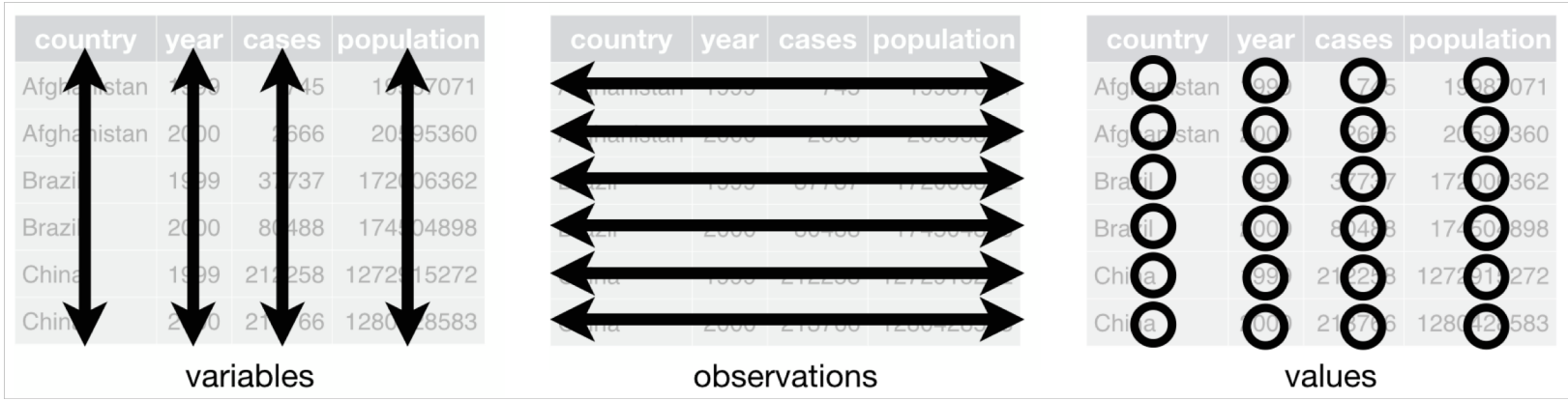


Figure from R4DS, p. 149 (https://r4ds.had.co.nz/tidy-data.html)

More on tidy data

- Why are these not tidy?

religion	<$10k	$10-20k	$20-30k	$30-40k	$40-50k	$50-75k
Agnostic	27	34	60	81	76	137
Atheist	12	27	37	52	35	70
Buddhist	27	21	30	34	33	58
Catholic	418	617	732	670	638	1116
Don't know/refused	15	14	15	11	10	35
Evangelical Prot	575	869	1064	982	881	1486
Hindu	1	9	7	9	11	34
Historically Black Prot	228	244	236	238	197	223
Jehovah's Witness	20	27	24	24	21	30
Jewish	19	19	25	25	30	95

Table 4: The first ten rows of data on income and religion from the Pew Forum. Three columns, `$75-100k`, `$100-150k` and `>150k`, have been omitted

year	artist	track	time	date.entered	wk1	wk2	wk3
2000	2 Pac	Baby Don't Cry	4:22	2000-02-26	87	82	72
2000	2Ge+her	The Hardest Part Of ...	3:15	2000-09-02	91	87	92
2000	3 Doors Down	Kryptonite	3:53	2000-04-08	81	70	68
2000	98^0	Give Me Just One Nig...	3:24	2000-08-19	51	39	34
2000	A*Teens	Dancing Queen	3:44	2000-07-08	97	97	96
2000	Aaliyah	I Don't Wanna	4:15	2000-01-29	84	62	51
2000	Aaliyah	Try Again	4:03	2000-03-18	59	53	38
2000	Adams, Yolanda	Open My Heart	5:30	2000-08-26	76	76	74

Table 7: The first eight Billboard top hits for 2000. Other columns not shown are `wk4`, `wk5`, ..., `wk75`.

Tables from Wickham (2014) (http://vita.had.co.nz/papers/tidy-data.pdf)

More on tidy data

- Are they now?

religion	income	freq
Agnostic	<$10k	27
Agnostic	$10-20k	34
Agnostic	$20-30k	60
Agnostic	$30-40k	81
Agnostic	$40-50k	76
Agnostic	$50-75k	137
Agnostic	$75-100k	122
Agnostic	$100-150k	109
Agnostic	>150k	84
Agnostic	Don't know/refused	96

Table 6: The first ten rows of the tidied Pew survey dataset on income and religion. The `column` has been renamed to `income`, and `value` to `freq`.

year	artist	time	track	date	week	rank
2000	2 Pac	4:22	Baby Don't Cry	2000-02-26	1	87
2000	2 Pac	4:22	Baby Don't Cry	2000-03-04	2	82
2000	2 Pac	4:22	Baby Don't Cry	2000-03-11	3	72
2000	2 Pac	4:22	Baby Don't Cry	2000-03-18	4	77
2000	2 Pac	4:22	Baby Don't Cry	2000-03-25	5	87
2000	2 Pac	4:22	Baby Don't Cry	2000-04-01	6	94
2000	2 Pac	4:22	Baby Don't Cry	2000-04-08	7	99
2000	2Ge+her	3:15	The Hardest Part Of ...	2000-09-02	1	91
2000	2Ge+her	3:15	The Hardest Part Of ...	2000-09-09	2	87
2000	2Ge+her	3:15	The Hardest Part Of ...	2000-09-16	3	92
2000	3 Doors Down	3:53	Kryptonite	2000-04-08	1	81
2000	3 Doors Down	3:53	Kryptonite	2000-04-15	2	70
2000	3 Doors Down	3:53	Kryptonite	2000-04-22	3	68
2000	3 Doors Down	3:53	Kryptonite	2000-04-29	4	67
2000	3 Doors Down	3:53	Kryptonite	2000-05-06	5	66

Table 8: First fifteen rows of the tidied billboard dataset. The `date` column does not appear in the original table, but can be computed from `date.entered` and `week`.

Tables from Wickham (2014) (http://vita.had.co.nz/papers/tidy-data.pdf)

Is our dataset tidy?

- Yes.
- How could it not be a tidy data?
- If there were a “Sales” column and a “Region” column, it wouldn’t be tidy.

```
# A tibble: 83,595 x 3
   Name                                Region       Sales
   <chr>                               <chr>        <dbl>
 1 ¡Shin Chan Flipa en colores!        EU_Sales      0
 2 ¡Shin Chan Flipa en colores!        Global_Sales  0.14
 3 ¡Shin Chan Flipa en colores!        JP_Sales      0.14
 4 ¡Shin Chan Flipa en colores!        NA_Sales      0
 5 ¡Shin Chan Flipa en colores!        Other_Sales   0
 6 .hack: Sekai no Mukou ni + Versus EU_Sales      0
 7 .hack: Sekai no Mukou ni + Versus Global_Sales  0.03
 8 .hack: Sekai no Mukou ni + Versus JP_Sales      0.03
 9 .hack: Sekai no Mukou ni + Versus NA_Sales      0
10 .hack: Sekai no Mukou ni + Versus Other_Sales   0
# ... with 83,585 more rows
```

Some tools to tidy a non-tidy dataset

- Long vs wide format: `spread, gather`
- `spread`: long to wide
  `gather`: wide to long

```
> df
# A tibble: 16,719 x 16
```

```
> df1
# A tibble: 83,595 x 14
```

```
> df2
# A tibble: 16,719 x 17
```

```
1  df1 <- df %>% mutate(UniqueKey = row_number()) %>% gather("NA_Sales",
   "EU_Sales", "JP_Sales", "Other_Sales", "Global_Sales", key="Region",
   value="Sales")
2
3  df1
4  df1 %>% arrange(Name, Region) %>% select(Name, Region, Sales)
5
6  df2 <- df1 %>% spread(Region, Sales)
7  df2
```

Inspect the data.

- Number of observations? Number of variables?
- What types of variables does it have?
- `glimpse(df)`

- Something wrong?
  - `Year_of_Release` and `User_Score` are parsed as `<chr>` not `<int>` or `<dbl>`.

```
> glimpse(df)
Observations: 16,719
Variables: 16
$ Name            <chr> "Wii Sports", "Super Mario Bros.", "Mario Kart Wii", "Wii Sports Resor...
$ Platform        <chr> "Wii", "NES", "Wii", "Wii", "GB", "GB", "DS", "Wii", "Wii", "NES", "DS...
$ Year_of_Release <chr> "2006", "1985", "2008", "2009", "1996", "1989", "2006", "2006", "2009"...
$ Genre           <chr> "Sports", "Platform", "Racing", "Sports", "Role-Playing", "Puzzle", "P...
$ Publisher       <chr> "Nintendo", "Nintendo", "Nintendo", "Nintendo", "Nintendo", "Nintendo"...
$ NA_Sales        <dbl> 41.36, 29.08, 15.68, 15.61, 11.27, 23.20, 11.28, 13.96, 14.44, 26.93, ...
$ EU_Sales        <dbl> 28.96, 3.58, 12.76, 10.93, 8.89, 2.26, 9.14, 9.18, 6.94, 0.63, 10.95, ...
$ JP_Sales        <dbl> 3.77, 6.81, 3.79, 3.28, 10.22, 4.22, 6.50, 2.93, 4.70, 0.28, 1.93, 4.1...
$ Other_Sales     <dbl> 8.45, 0.77, 3.29, 2.95, 1.00, 0.58, 2.88, 2.84, 2.24, 0.47, 2.74, 1.90...
$ Global_Sales    <dbl> 82.53, 40.24, 35.52, 32.77, 31.37, 30.26, 29.80, 28.92, 28.32, 28.31, ...
$ Critic_Score    <int> 76, NA, 82, 80, NA, NA, 89, 58, 87, NA, NA, 91, NA, 80, 61, 80, 97, 95...
$ Critic_Count    <int> 51, NA, 73, 73, NA, NA, 65, 41, 80, NA, NA, 64, NA, 63, 45, 33, 50, 80...
$ User_Score      <chr> "8", NA, "8.3", "8", NA, NA, "8.5", "6.6", "8.4", NA, NA, "8.6", NA, "...
$ User_Count      <int> 322, NA, 709, 192, NA, NA, 431, 129, 594, NA, NA, 464, NA, 146, 106, 5...
$ Developer       <chr> "Nintendo", NA, "Nintendo", "Nintendo", NA, NA, "Nintendo", "Nintendo"...
$ Rating          <chr> "E", NA, "E", "E", NA, NA, "E", "E", "E", NA, NA, "E", NA, "E", "E", "...
```

Overriding default parsing behavior during import.

```r
# Check what's wrong
df %>% type_convert(cols(Year_of_Release=col_integer()))
warnings()
df %>% type_convert(cols(User_Score=col_double()))
warnings()

# 1. fix it after read_csv
df <- df %>% type_convert(
  col_types = cols(
    Year_of_Release = col_integer(),
    User_Score=col_double()
    ), na = c("N/A", "tbd"))

# 2. fix it when reading the file
df <- read_csv("data/Video_Games_Sales_as_at_22_Dec_2016.csv", na = c("N/A",
"tbd"))
```

```
> df <- read_csv("data/Video_Games_
Parsed with column specification:
cols(
  Name = col_character(),
  Platform = col_character(),
  Year_of_Release = col_integer(),
  Genre = col_character(),
  Publisher = col_character(),
  NA_Sales = col_double(),
  EU_Sales = col_double(),
  JP_Sales = col_double(),
  Other_Sales = col_double(),
  Global_Sales = col_double(),
  Critic_Score = col_integer(),
  Critic_Count = col_integer(),
  User_Score = col_double(),
  User_Count = col_integer(),
  Developer = col_character(),
  Rating = col_character()
)
```

5 verbs of data transformation in tidyverse

- Column operations: `select, mutate`
- Row operations: `filter, arrange`
- Summarize: `summarize`

```
> df
# A tibble: 16,719 x 16
   Name  Platform Year_of_Release Genre Publisher NA_Sales EU_Sales JP_Sales Other_Sales
   <chr> <chr>    <chr>           <chr> <chr>        <dbl>    <dbl>    <dbl>       <dbl>
 1 Wii … Wii      2006            Spor… Nintendo      41.4     29.0     3.77        8.45
 2 Supe… NES      1985            Plat… Nintendo      29.1      3.58    6.81        0.77
 3 Mari… Wii      2008            Raci… Nintendo      15.7     12.8     3.79        3.29
 4 Wii … Wii      2009            Spor… Nintendo      15.6     10.9     3.28        2.95
 5 Poke… GB       1996            Role… Nintendo      11.3      8.89   10.2         1
 6 Tetr… GB       1989            Puzz… Nintendo      23.2      2.26    4.22        0.580
 7 New … DS       2006            Plat… Nintendo      11.3      9.14    6.5         2.88
 8 Wii … Wii      2006            Misc  Nintendo      14.0      9.18    2.93        2.84
 9 New … Wii      2009            Plat… Nintendo      14.4      6.94    4.7         2.24
10 Duck… NES      1984            Shoo… Nintendo      26.9      0.63    0.28        0.47
# ... with 16,709 more rows, and 7 more variables: Global_Sales <dbl>, Critic_Score <int>,
#   Critic_Count <int>, User_Score <chr>, User_Count <int>, Developer <chr>, Rating <chr>
```

- You select and mutate “variables”, filter and arrange “observations”.
- `mutate` = add or alter columns
  `arrange` = sort

Select

- You select “columns” or “variables” NOT “rows” or “observations”.
- Let’s select these 5 columns:
  `Name, Platform, Year_of_Release, Genre, Global_Sales`.

```
1   df %>% select(Name, Platform, Year_of_Release, Genre, Global_Sales)
```

```
> df %>% select(Name, Platform, Year_of_Release, Genre, Global_Sales)
# A tibble: 16,719 x 5
   Name                      Platform Year_of_Release Genre        Global_Sales
   <chr>                     <chr>              <int> <chr>              <dbl>
 1 Wii Sports                Wii                 2006 Sports              82.5
 2 Super Mario Bros.         NES                 1985 Platform            40.2
 3 Mario Kart Wii            Wii                 2008 Racing              35.5
 4 Wii Sports Resort         Wii                 2009 Sports              32.8
 5 Pokemon Red/Pokemon Blue  GB                  1996 Role-Playing        31.4
 6 Tetris                    GB                  1989 Puzzle              30.3
 7 New Super Mario Bros.     DS                  2006 Platform            29.8
 8 Wii Play                  Wii                 2006 Misc                28.9
 9 New Super Mario Bros. Wii Wii                 2009 Platform            28.3
10 Duck Hunt                 NES                 1984 Shooter             28.3
# ... with 16,709 more rows
```

Mutate

- You add a new column as a combination (or operation) of other columns.
  You can also alter a current column by creating a new column with the same name.
- Let's create a new column called:
  `Total_Sales = NA_Sales + EU_Sales + JP_Sales + Other_Sales`.

```
1   df %>% mutate(Total_Sales = NA_Sales + EU_Sales + JP_Sales + Other_Sales)
```

```
> df %>% mutate(Total_Sales = NA_Sales + EU_Sales + JP_Sales + Other_Sales)
# A tibble: 16,719 x 17
   Name  Platform Year_of_Release Genre Publisher NA_Sales EU_Sales JP_Sales Other_Sales
   <chr> <chr>              <int> <chr> <chr>        <dbl>    <dbl>    <dbl>       <dbl>
 1 Wii … Wii                2006 Spor… Nintendo      41.4    29.0      3.77       8.45
 2 Supe… NES                1985 Plat… Nintendo      29.1     3.58     6.81       0.77
 3 Mari… Wii                2008 Raci… Nintendo      15.7    12.8      3.79       3.29
 4 Wii … Wii                2009 Spor… Nintendo      15.6    10.9      3.28       2.95
 5 Poke… GB                 1996 Role… Nintendo      11.3     8.89    10.2        1
 6 Tetr… GB                 1989 Puzz… Nintendo      23.2     2.26     4.22       0.580
 7 New … DS                 2006 Plat… Nintendo      11.3     9.14     6.5        2.88
 8 Wii … Wii                2006 Misc  Nintendo      14.0     9.18     2.93       2.84
 9 New … Wii                2009 Plat… Nintendo      14.4     6.94     4.7        2.24
10 Duck… NES                1984 Shoo… Nintendo      26.9     0.63     0.28       0.47
# ... with 16,709 more rows, and 8 more variables: Global_Sales <dbl>, Critic_Score <int>,
#   Critic_Count <int>, User_Score <dbl>, User_Count <int>, Developer <chr>, Rating <chr>,
#   Total_Sales <dbl>
```

Filter

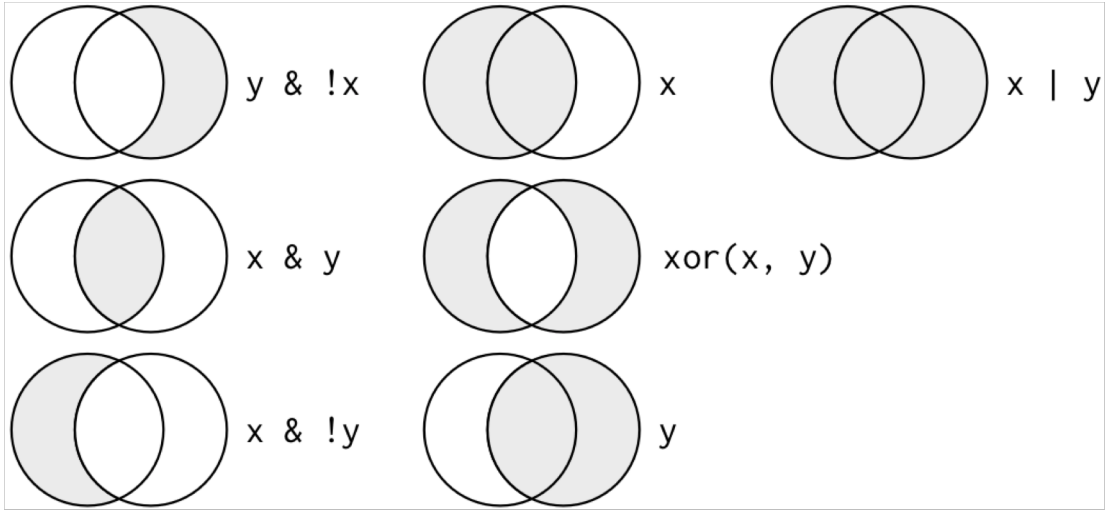
- You can keep a subset of observations by filtering out others.
- Let's collect videos games released on PS4 or Xbox One.

```
1  # See first what unique (distinct) values exist in Platform variable.
2  unique(df$Platform)
3
4  df %>% filter(Platform=="PS4" | Platform=="XOne")
```

```
> df %>% filter(Platform=="PS4" | Platform=="XOne")
# A tibble: 640 x 16
   Name  Platform Year_of_Release Genre Publisher NA_Sales EU_Sales JP_Sales Other_Sales
   <chr> <chr>              <int> <chr> <chr>        <dbl>    <dbl>    <dbl>       <dbl>
 1 Call… PS4                 2015 Shoo… Activisi…     6.03     5.86     0.36        2.38
 2 Gran… PS4                 2014 Acti… Take-Two…     3.96     6.31     0.38        1.97
 3 FIFA… PS4                 2015 Spor… Electron…     1.12     6.12     0.06        1.28
 4 Star… PS4                 2015 Shoo… Electron…     2.99     3.49     0.22        1.28
 5 Call… PS4                 2014 Shoo… Activisi…     2.81     3.48     0.14        1.23
 6 FIFA… PS4                 2016 Spor… Electron…     0.66     5.75     0.08        1.11
 7 Call… XOne                2015 Shoo… Activisi…     4.59     2.11     0.01        0.68
 8 Fall… PS4                 2015 Role… Bethesda…     2.53     3.27     0.24        1.13
 9 FIFA… PS4                 2014 Spor… Electron…     0.8      4.33     0.05        0.9
10 Dest… PS4                 2014 Shoo… Activisi…     2.49     2.07     0.16        0.92
# ... with 630 more rows, and 7 more variables: Global_Sales <dbl>, Critic_Score <int>,
#   Critic_Count <int>, User_Score <dbl>, User_Count <int>, Developer <chr>, Rating <chr>
```

Logical operators review

- List of logical operators in order of precedence
  - `( )` parenthesis
  - `!` negation
  - `& &&` and
  - `| ||` or
- These are the same.
  - `x | !y & z`
  - `(x | ((!y) & z))`
- It's different.
  - `(x | !y) & z`



- https://stat.ethz.ch/R-manual/R-devel/library/base/html/Syntax.html

Figure from R4DS, p. 47 (https://r4ds.had.co.nz/transform.html)

Arrange

- You sort rows (or observations) with one or more criteria.
- Let's sort the data according to the following criteria in order:
  - (1) in descending order of `Year_of_Release`
  - (2) in ascending order of `Platform`
  - (3) in ascending order of `Name`

```
1  df %>% arrange(-Year_of_Release, Platform, Name)
```

```
> df %>% arrange(-Year_of_Release, Platform, Name)
# A tibble: 16,719 x 16
   Name  Platform Year_of_Release Genre Publisher NA_Sales EU_Sales JP_Sales Other_Sales
   <chr> <chr>              <int> <chr> <chr>        <dbl>    <dbl>    <dbl>       <dbl>
 1 Imag… DS                  2020 Simu… Ubisoft       0.27        0     0           0.02
 2 Phan… PS4                 2017 Role… Sega          0           0     0.04        0
 3 Brot… PSV                 2017 Acti… Idea Fac…     0           0     0.01        0
 4 Phan… PSV                 2017 Role… Sega          0           0     0.01        0
 5 12-S… 3DS                 2016 Adve… Happinet      0           0     0.05        0
 6 3DS … 3DS                 2016 Misc  Sega          0.03        0     0           0
 7 Ace … 3DS                 2016 Adve… Capcom        0           0     0.27        0
 8 Aika… 3DS                 2016 Acti… Namco Ba…     0           0     0.01        0
 9 Ansa… 3DS                 2016 Acti… Namco Ba…     0           0     0.06        0
10 Azur… 3DS                 2016 Acti… Yacht Cl…     0.01        0     0           0
# ... with 16,709 more rows, and 7 more variables: Global_Sales <dbl>, Critic_Score <int>,
#   Critic_Count <int>, User_Score <dbl>, User_Count <int>, Developer <chr>, Rating <chr>
```

Let's chain them with pipe operator (%>%).

- Let's do this altogether at once.
  - (1) Create a column called `NAEU_Sales = NA_Sales + EU_Sales`.
  - (2) Select columns: `Name, Platform, Year_of_Release, NAEU_Sales`.
  - (3) Keep only observations with `NAEU_Sales` greater than 5 and released on Wii.
  - (4) Sort in descending order of `NAEU_Sales`.

```
1  df %>%
2    mutate(NAEU_Sales = NA_Sales + EU_Sales) %>%
3    select(Name, Platform, Year_of_Release, NAEU_Sales) %>%
4    filter(NAEU_Sales>5 & Platform=="Wii") %>%
5    arrange(-NAEU_Sales)
```

```
> df %>%
+   mutate(NAEU_Sales = NA_Sales + EU_Sales) %>%
+   select(Name, Platform, Year_of_Release, NAEU_Sales) %>%
+   filter(NAEU_Sales>5 & Platform=="Wii") %>%
+   arrange(-NAEU_Sales)
# A tibble: 20 x 4
   Name                      Platform Year_of_Release NAEU_Sales
   <chr>                     <chr>              <int>      <dbl>
 1 Wii Sports                Wii                 2006       70.3
 2 Mario Kart Wii            Wii                 2008       28.4
 3 Wii Sports Resort         Wii                 2009       26.5
 4 Wii Play                  Wii                 2006       23.1
 5 New Super Mario Bros. Wii Wii                 2009       21.4
 6 Wii Fit Plus              Wii                 2009       17.5
 7 Wii Fit                   Wii                 2007       17.0
 8 Super Mario Galaxy        Wii                 2007        9.41
 9 Super Smash Bros. Brawl   Wii                 2008        9.17
10 Just Dance 3              Wii                 2011        9.06
```

Revisiting the piping (%>%) syntax

- Pipe operator basically passes left-hand side object to right-hand side function as a first argument.

- What if we didn't have pipe operator and needed to achieve the same?

```
df2 <- mutate(df, NAEU_Sales = NA_Sales + EU_Sales)
df2 <- select(df2, Name, Platform, Year_of_Release, NAEU_Sales)
df2 <- filter(df2, NAEU_Sales>5 & Platform=="Wii")
df2 <- arrange(df2, -NAEU_Sales)
df2
```

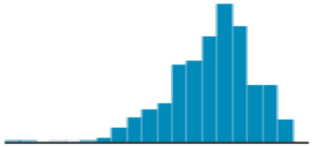
- For more on pipe, read R4DS Chapter 14. Pipes with magrittr.

Descriptive Statistics

Reading
- R4DS Chapter 5. Exploratory Data Analysis

A brief look at descriptive statistics and visualization

Video_Games_Sales_as_at_22_Dec_2016.csv (503.05 KB) — 16 of 16 columns

	Name Name of the game	Platform Console on which the game is running	Year_of_Release Year of the game released	Genre Game's category	Publisher Publisher	NA_S Game sa America units)
	11562 unique values	PS2 13% DS 13% Other (29) 74%	 1.98k 2.02k	Action 20% Sports 14% Other (10) 66%	Electronic Arts 8% Activision 6% Other (580) 86%	 0
1	Wii Sports	Wii	2006	Sports	Nintendo	
2	Super Mario Bros.	NES	1985	Platform	Nintendo	
3	Mario Kart Wii	Wii	2008	Racing	Nintendo	
4	Wii Sports Resort	Wii	2009	Sports	Nintendo	
5	Pokemon Red/Pokemon Blue	GB	1996	Role-Playing	Nintendo	
6	Tetris	GB	1989	Puzzle	Nintendo	

Types of variables

- Categorical variables (or qualitative variable)
    - Platform: PS4, XOne, …
    - Genre: Action, Sports, …
- Numerical variables (or quantitative variable)
    - Discrete variables
        - Year_of_Release: 2006, 1985, 2008, 2009, …
    - Continuous variables
        - Global_Sales: 82.53, 40.24, 35.52, 32.77, …
- Understanding a single individual variable is to understand how it “varies” over different observations or measurements.
- Figuring out how observations are “distributed” along each of these variables is a good starting point.

Figuring out the distribution

- In essence, it is about counting observations that fall into a certain range.
- For categorical variable (and discrete variable sometimes), it's straightforward. And it's called tabulation.
- For continuous variable (and discrete variable sometimes), it requires another step before tabulation: binning.

Variable type	How to figure out the distribution?	Visualization
Categorical	Tabulation	Bar / Column Chart
Discrete	Both	Both
Continuous	Binning + Tabulation	Histogram, Density Plot, Boxplot

Tabulation for categorical variable

- You just need to count the number of observations for each category.
- Let's count with `count` function.
- Since categories don't have intrinsic ordering,
  it's usually useful to sort them by count in descending order.

```
1   df %>% count(Platform) %>% arrange(-n)
```

```
> df %>% count(Platform) %>% arrange(-n)
# A tibble: 31 x 2
   Platform     n
   <chr>    <int>
 1 PS2       2161
 2 DS        2152
 3 PS3       1331
 4 Wii       1320
 5 X360      1262
 6 PSP       1209
 7 PS        1197
 8 PC         974
 9 XB         824
10 GBA        822
# ... with 21 more rows
```

Binning + tabulation for continuous variable

- For a continuous variable, you first need to discretize the variable into ranges.
- Three binning methods
  - `cut_width`: makes groups of the given `width`
  - `cut_interval`: makes `n` groups with equal range
  - `cut_number`: makes `n` groups with (approximately) equal numbers of observations

```
1  df %>% count(cut_width(Global_Sales, 10))
2  df %>% count(cut_interval(Global_Sales, 10))
3  df %>% count(cut_number(Global_Sales, 10))
```

```
> df %>% count(cut_width(Global_Sales, 10))
# A tibble: 6 x 2
  `cut_width(Global_Sales, 10)`     n
  <fct>                         <int>
1 [-5,5]                        16512
2 (5,15]                          179
3 (15,25]                          18
4 (25,35]                           7
5 (35,45]                           2
6 (75,85]                           1
```

```
> df %>% count(cut_interval(Global_Sales, 10))
# A tibble: 6 x 2
  `cut_interval(Global_Sales, 10)`     n
  <fct>                             <int>
1 [0.01,8.26]                       16638
2 (8.26,16.5]                          58
3 (16.5,24.8]                          13
4 (24.8,33]                             7
5 (33,41.3]                             2
6 (74.3,82.5]                           1
```

```
> df %>% count(cut_number(Global_Sales, 10))
# A tibble: 10 x 2
   `cut_number(Global_Sales, 10)`     n
   <fct>                           <int>
 1 [0.01,0.02]                      1725
 2 (0.02,0.05]                      2134
 3 (0.05,0.08]                      1594
 4 (0.08,0.11]                      1271
 5 (0.11,0.17]                      1793
 6 (0.17,0.25]                      1620
 7 (0.25,0.38]                      1624
 8 (0.38,0.6]                       1636
 9 (0.6,1.2]                        1651
10 (1.2,82.5]                       1671
```

(Binning +) tabulation for discrete variable

- For a discrete variable, you can tabulate with or without binning.

```
1  df %>% count(Year_of_Release) %>% arrange(-n)
2  df %>% count(cut_width(Year_of_Release, 10))
```

```
> df %>% count(Year_of_Release) %>% arrange(-n)
# A tibble: 40 x 2
   Year_of_Release     n
             <int> <int>
 1            2008  1427
 2            2009  1426
 3            2010  1255
 4            2007  1197
 5            2011  1136
 6            2006  1006
 7            2005   939
 8            2002   829
 9            2003   775
10            2004   762
# ... with 30 more rows
```

```
> df %>% count(cut_width(Year_of_Release, 10))
# A tibble: 6 x 2
  `cut_width(Year_of_Release, 10)`     n
  <fct>                            <int>
1 [1975,1985]                        136
2 (1985,1995]                        571
3 (1995,2005]                       5406
4 (2005,2015]                       9831
5 (2015,2025]                        506
6 NA                                 269
```

Summary statistics for numerical variables

- Measure of location
  - Single representative numbers: mean, median
- Measure of spread
  - range, inter-quartile range, standard deviation
- Measure of rank
  - Five number summary
    - Minimum
    - First quartile
    - Median (= second quartile)
    - Third quartile
    - Maximum

Summarize

- The fifth verb in tidyverse is `summarize`, which can be used to compute summary statistics.

```
df %>% summarize(mean = mean(Year_of_Release, na.rm = T))
df %>% summarize(
  n = sum(!is.na(Year_of_Release)),
  mean = mean(Year_of_Release, na.rm = T),
  sd = sd(Year_of_Release, na.rm = T),
  min = min(Year_of_Release, na.rm = T),
  max = max(Year_of_Release, na.rm = T)
)
```

- Problems
  - It is redundant to type the variable name repeatedly, which makes this code error prone.
  - It gets even worse when you want to compute these summary stats for multiple variables.

Compute multiple summary stats for multiple variables

- Let's solve the problem in a `tidyverse` (`tidyr`, `dplyr`) way.
- `summarize_at` and `summarize_all` functions are useful here.

```
df %>%
  summarize_at(
    c("Year_of_Release", "Global_Sales"),
    funs(n = sum(!is.na(.)), mean, sd, min, max),
    na.rm = T
  ) %>%
  gather(stat, val) %>%
  separate(stat, c("var", "stat"), sep = "_(?!.*_)") %>%
  spread(stat, val) %>%
  select(var, n, mean, sd, min, max) %>%
  arrange(factor(var, levels=colnames(df)))
```

- Try to achieve the same output with `summarize_all` instead of `summarize_at`.
- To see what the regular expression, _(?!.*_), means, visit https://regexr.com/44irf.

Export and restore data

- You can export the summary statistics table into a csv file to load in Excel.

```
sum_stats %>% write_csv("summary_stats.csv", na="")
sum_stats %>% write_excel_csv("summary_stats_excel.csv", na="")
```

- One downside of csv file is that you lose metadata such as data types for variables.
- RDS file saves a tibble as is and allows you to restore the data in the same format.

```
df %>% write_rds("temp_data.rds", "gz")
df2 <- read_rds("temp_data.rds")
```

You can do the same thing in multiple ways in tidyverse.

- Let's tabulate a categorical variable with `summarize` and `group_by`.

```
1  df %>% group_by(Platform) %>% summarize(n = n()) %>% arrange(-n)
```

```
> df %>% group_by(Platform) %>% summarize(n = n()) %>% arrange(-n)
# A tibble: 31 x 2
   Platform     n
   <chr>    <int>
 1 PS2       2161
 2 DS        2152
 3 PS3       1331
 4 Wii       1320
 5 X360      1262
 6 PSP       1209
 7 PS        1197
 8 PC         974
 9 XB         824
10 GBA        822
# ... with 21 more rows
```

```
> df %>% group_by(Platform) %>% summarize(n = n()) %>% mutate(freq = n/sum(n)*100) %>% arrange(-n)
# A tibble: 31 x 3
   Platform     n  freq
   <chr>    <int> <dbl>
 1 PS2       2161 12.9
 2 DS        2152 12.9
 3 PS3       1331  7.96
 4 Wii       1320  7.90
 5 X360      1262  7.55
 6 PSP       1209  7.23
 7 PS        1197  7.16
 8 PC         974  5.83
 9 XB         824  4.93
10 GBA        822  4.92
# ... with 21 more rows
```

- This combination of `summarize` and `group_by` will prove very useful when we summarize the relationship between two variables.

Common discrete distributions

Distribution	R function	Example
Bernoulli	`rbernoulli`	Head or tail in a coin toss
Rectangular (or uniform)		Number of 1's in n dice rolls
Binomial	`rbinom`	Number of heads in n coin tosses
Geometric	`rgeom`	Number of failures until the first success
Negative binomial	`rnbinom`	Number of failures until nth success
Poisson	`rpois`	Number of events if occurrences are independent from each other
Zipf		Number of occurrences of words in texts

Common continuous distributions

Distribution	R function	Example
Uniform	`runif`	Random number within a range
Normal	`rnorm`	Height of people in a population
Exponential	`rexp`	Length of time between independent events
Lognormal	`rlnorm`	Length of comments, system repair times, income distribution
Pareto		Wealth distribution, city size, size of meteorites, 80/20 rule

Create a new tibble with random variables

```r
N <- 1000
tibble(
  seq = 1:N,
  rbernoulli = rbernoulli(n=N, p=.1),
  runifint = sample(x=1:6, size=N, replace=T),
  rbinom = rbinom(n=N, size=10, prob=.1),
  rgeom = rgeom(n=N, prob=.1),
  rnbinom = rnbinom(n=N, size=10, prob=.1),
  rpois = rpois(n=N, lambda=10),
  runif = runif(n=N, min=10, max=20),
  rnorm = rnorm(n=N, mean=10, sd=2),
  rexp = rexp(n=N, rate=.1),
  rlnorm = rlnorm(n=N, meanlog=2, sdlog=2)
  )
```

- Try to compute the summary statistics table about this randomly generated dataset.

Basic Plots

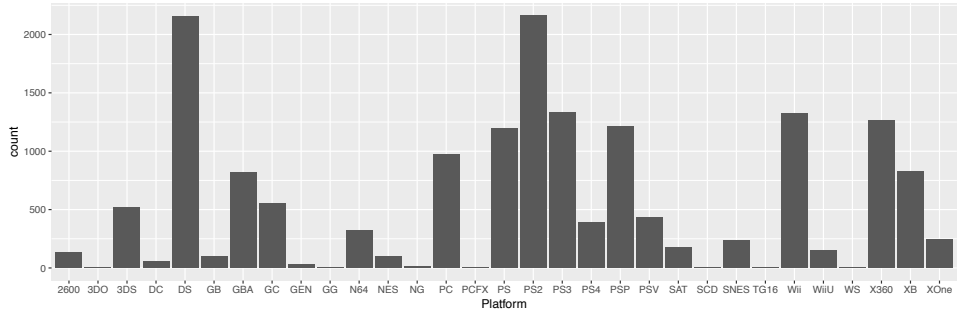
Reading

- R4DS Chapter 1. Data Visualization with ggplot2

Bar chart

- Let’s start with visualizing the distribution of a categorical (or discrete) variable.
- Bar chart is a go-to solution.

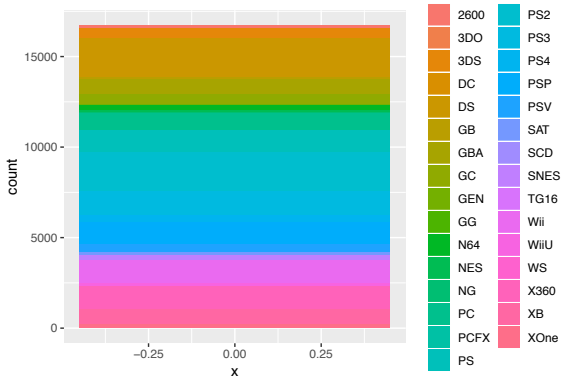
```
1  ggplot(df) + geom_bar(aes(x=Platform))
2  ggsave("img/platform.pdf", width = 12, height = 4)
```

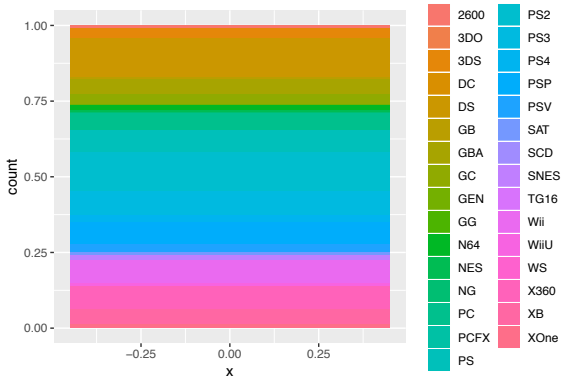


Stacked bar chart

- We can stack up bars into a single bar.
- This looks dumb, but it will be useful when we visualize more than one variable.

```
1  ggplot(df) + geom_bar(aes(x=0, fill=Platform))
2  ggplot(df) + geom_bar(aes(x=0, fill=Platform), position="fill")
```

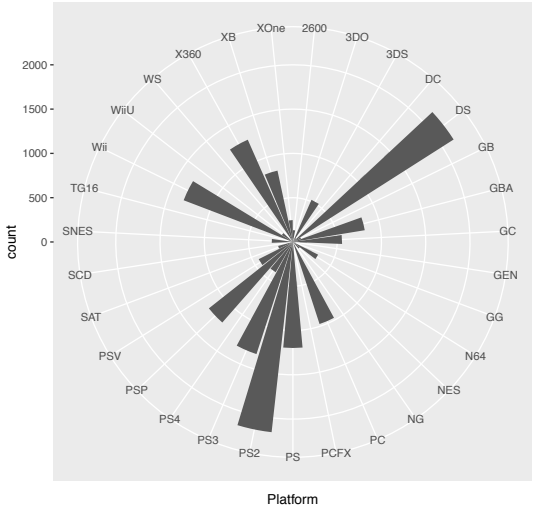


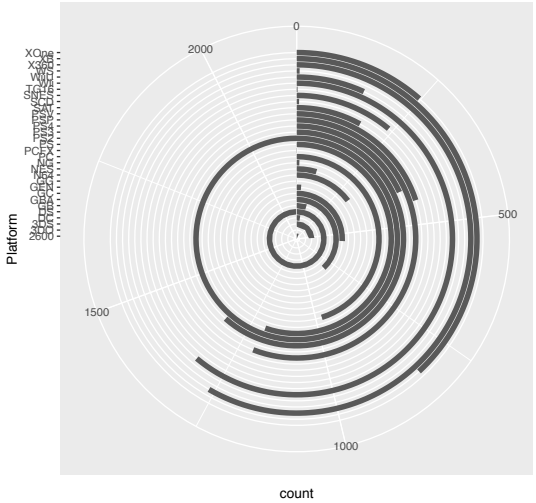


Radial charts

- How is bar chart related to pie chart? Both portray categorical distribution.

```
1  ggplot(df) + geom_bar(aes(x=Platform)) + coord_polar()
2  ggplot(df) + geom_bar(aes(x=Platform)) + coord_polar(theta = "y")
```

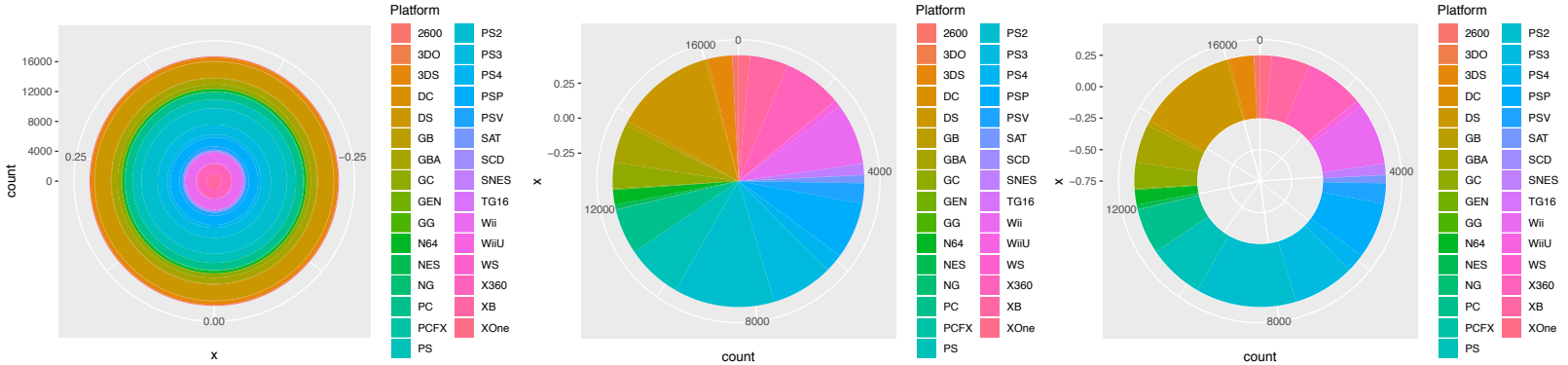




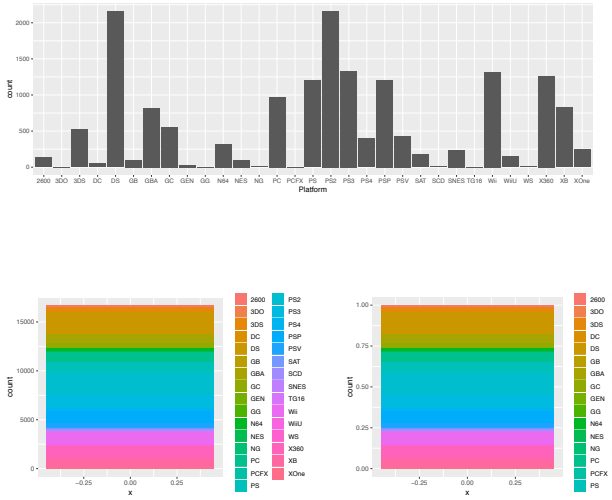
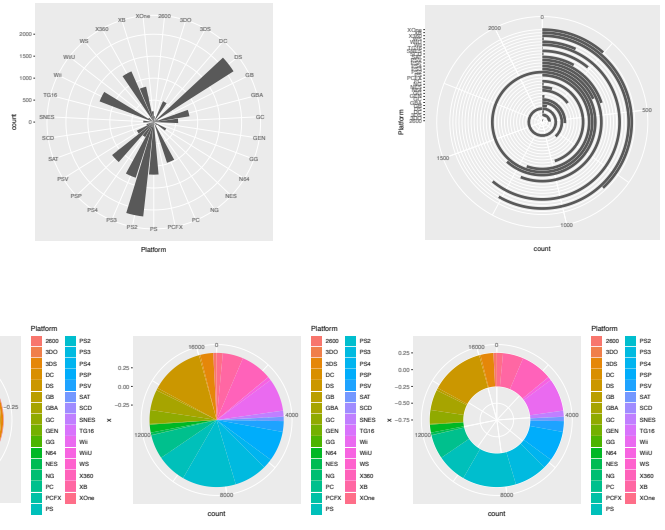
Bullseye chart, pie chart, donut chart

- The pie chart family is basically a stacked bar chart in polar coordinate.

```
ggplot(df) + geom_bar(aes(x=0, fill=Platform)) + coord_polar()
ggplot(df) + geom_bar(aes(x=0, fill=Platform)) + coord_polar("y")
ggplot(df) + geom_bar(aes(x=0, fill=Platform), width=.5) + coord_polar("y") +
xlim(c(-.75,.25))
```



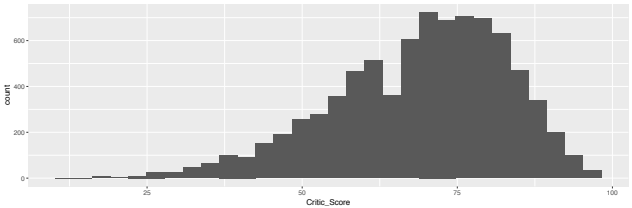
Recap on visualizing a categorical (or discrete) variable

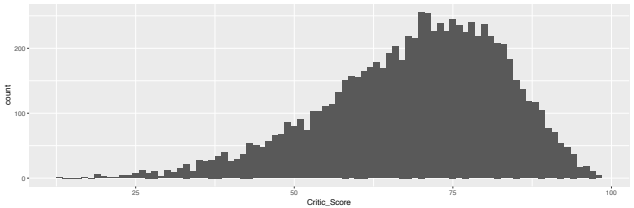
Cartesian coordinate	Polar coordinate
	

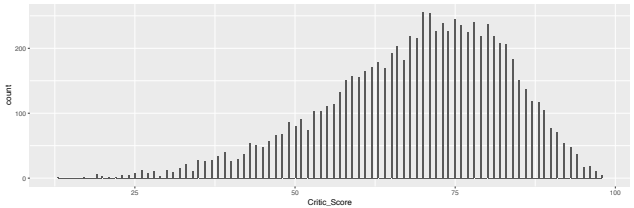
Histogram

- Let's move on to visualizing the distribution of a numerical (both continuous and discrete) variable.
- Histogram visualizes the distribution of a single numerical variable.
  It's equivalent to bar chart for categorical variable.

```
1  ggplot(df) + geom_histogram(aes(x=Critic_Score))
2  ggplot(df) + geom_histogram(aes(x=Critic_Score), binwidth = 1)
3  ggplot(df) + geom_histogram(aes(x=Critic_Score), bins = 500)
```



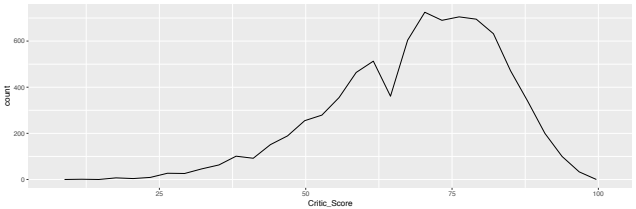


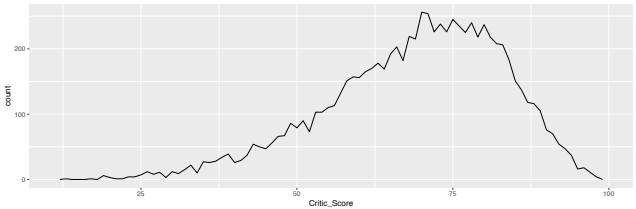


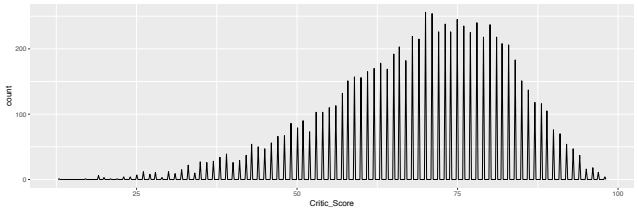
Line plot (frequency polyline plots)

- How about connecting bars in histogram so that we can save some ink?

```
1  ggplot(df) + geom_freqpoly(aes(x=Critic_Score))
2  ggplot(df) + geom_freqpoly(aes(x=Critic_Score), binwidth = 1)
3  ggplot(df) + geom_freqpoly(aes(x=Critic_Score), bins = 500)
```



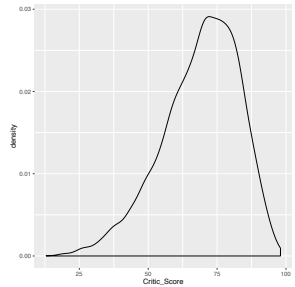


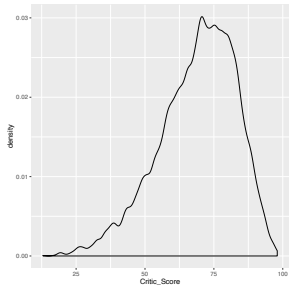


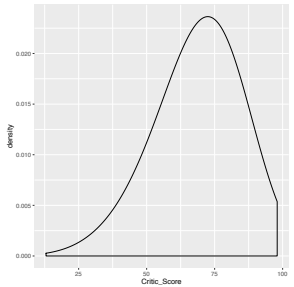
Density plot

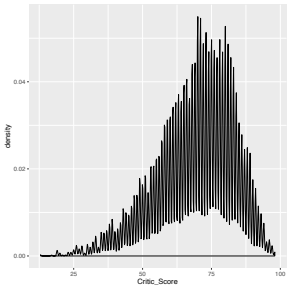
- Density plot smooths out and normalizes a histogram with moving window of width called “bandwidth”.

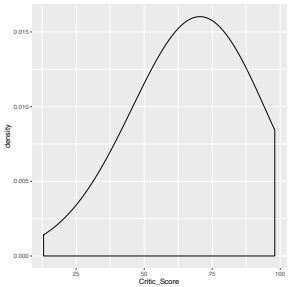
```
1  ggplot(df) + geom_density(aes(x=Critic_Score))
2  ggplot(df) + geom_density(aes(x=Critic_Score), bw=1)
3  ggplot(df) + geom_density(aes(x=Critic_Score), bw=10)
4  ggplot(df) + geom_density(aes(x=Critic_Score), adjust=.1)
5  ggplot(df) + geom_density(aes(x=Critic_Score), adjust=10)
```







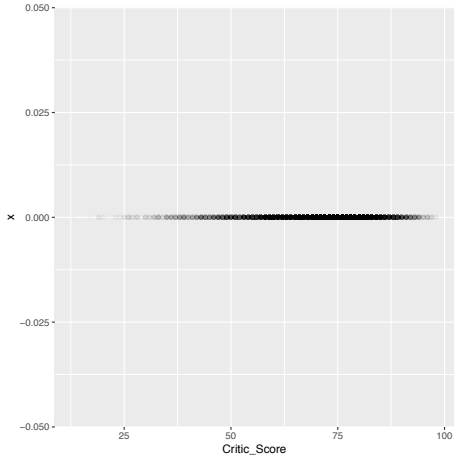


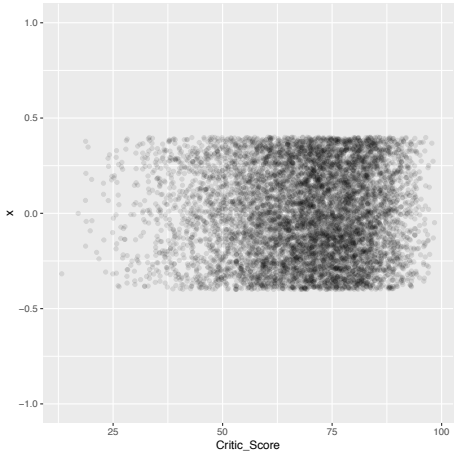


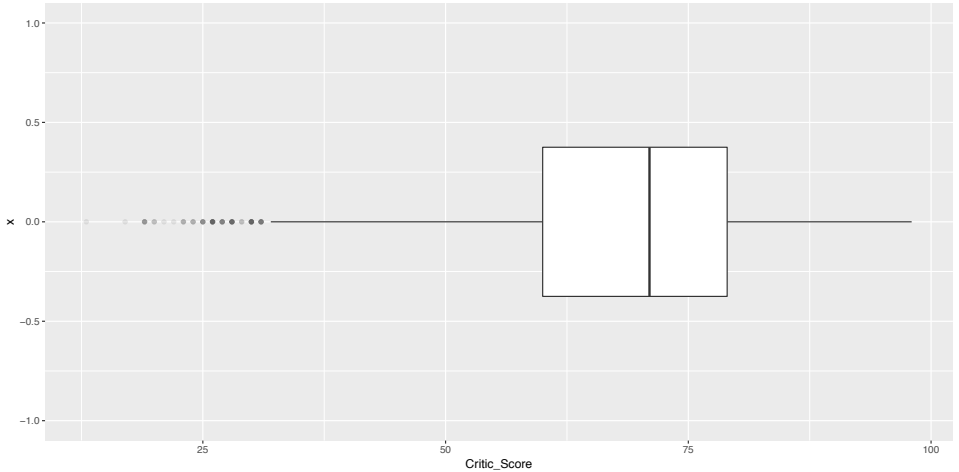
Box plot

- Box plot provides a visual five-number summary of a numerical variable.

```
1  ggplot(df) + geom_point(aes(x=0, y=Critic_Score), alpha=.05) + coord_flip()
2  ggplot(df) + geom_jitter(aes(x=0, y=Critic_Score), alpha=.2) + xlim(c(-1,1))
   + coord_flip()
3  ggplot(df) + geom_boxplot(aes(x=0, y=Critic_Score), outlier.alpha=.1) +
   xlim(c(-1,1)) + coord_flip()
```



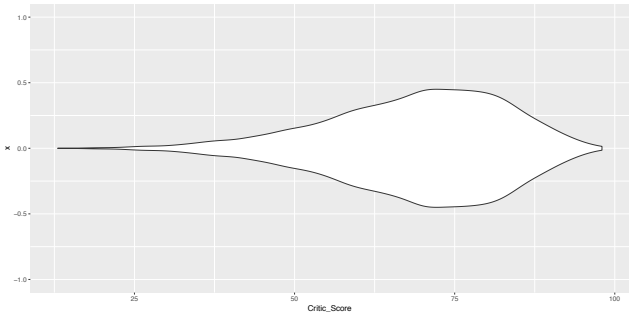


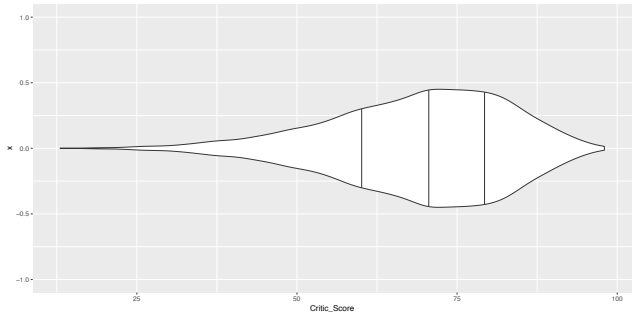


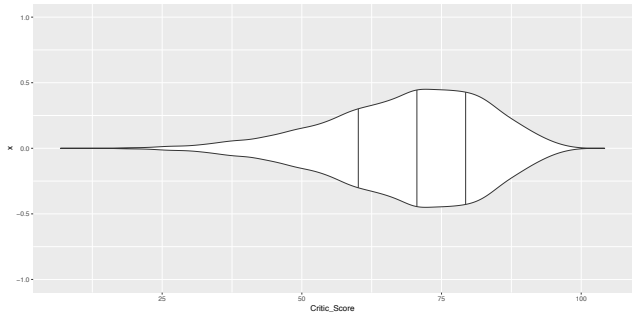
Violin plot

- Violin plot is a combination of density plot and box plot.

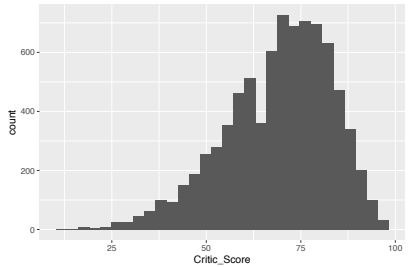
```
1  ggplot(df) + geom_violin(aes(x=0, y=Critic_Score)) + xlim(c(-1,1)) +
   coord_flip()
2  ggplot(df) + geom_violin(aes(x=0, y=Critic_Score),
   draw_quantiles=c(.25,.5,.75)) + xlim(c(-1,1)) + coord_flip()
3  ggplot(df) + geom_violin(aes(x=0, y=Critic_Score),
   draw_quantiles=c(.25,.5,.75), trim=F) + xlim(c(-1,1)) + coord_flip()
```

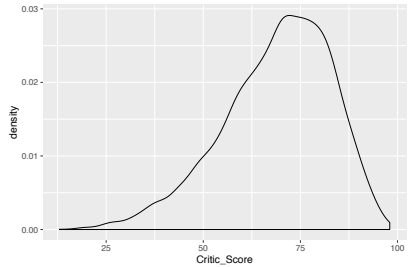


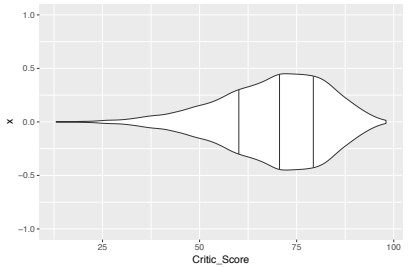


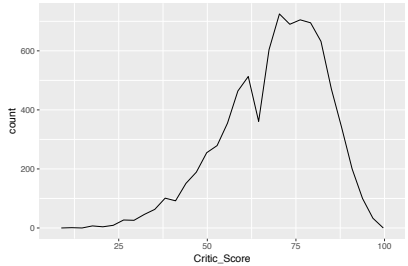


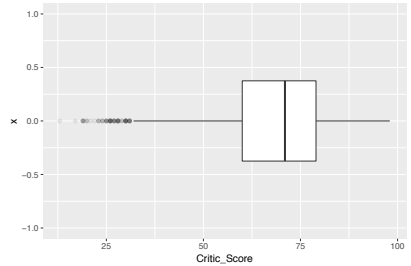
Recap on visualizing a continuous (or discrete) variable











What to look for in a distribution visualization?

- Location
- Shape
  - Dispersion
  - Skewness
  - Kurtosis
  - Modality
- Outlier

Characterizing the shape of a histogram

- Symmetric and unimodal
- Skewed right
- Skewed left
- Bimodal
- Multimodal
- Symmetric

Figure from Wikipedia, (https://en.wikipedia.org/wiki/Histogram)

Grammar of graphics ("gg" in ggplot2)

- By Leland Wilkinson in 1999
- The thesis is that:
  a systematic mapping between data and visuals can be developed and articulated.
- It must sound so natural these days given your experience with ggplot2 already.
- Many statistical graphing systems do share and implicitly incorporate this "grammar" to varying degrees, but they do not have this "grammar" thought through in their implementation.
- Nevertheless, this stream of thought had profound influences on recently developed modern graphing systems.

Data-ink ratio

- By Edward Tufte in 1983
- A guru of data visualization striving to develop guiding principles for creating “good” data visualization in a systematic way.
- Data-ink ratio is one of such guiding principles.

> *“A large share of ink on a graphic should present data-information, the ink changing as the data change. Data-ink is the non-erasable core of a graphic, the non-redundant ink arranged in response to variation in the numbers represented.”*
>
> Tufte, 1983

- A few short reads:
  - InfoVis Wiki on data-ink ratio
  - A Medium post on the data-ink ratio

Layered grammar of graphics

- Hadley Wickham's paper: http://vita.had.co.nz/papers/layered-grammar.pdf
- Three essential layers of a graphic
  - Data (of course)
  - Aesthetics (mapping variables to visual elements)
  - Geometries (
- Four optional layers of a graphic
  - Facets
  - Statistics
  - Coordinates
  - Themes

Implementation of the philosophy in ggplot2

- The barebone structure of ggplot2 syntax
  - ```
    ggplot(data = <DATA>) +
      <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
    ```
- More complete ggplot2 syntax
  - ```
    ggplot(data = <DATA>) +
      <GEOM_FUNCTION>(
        mapping = aes(<MAPPINGS>),
        stat = <STAT>,
        position = <POSITION>) +
      <SCALE_FUNCTION> +
      <COORDINATE_FUNCTION> +
      <FACET_FUNCTION> +
      <THEME_FUNCTION>
    ```

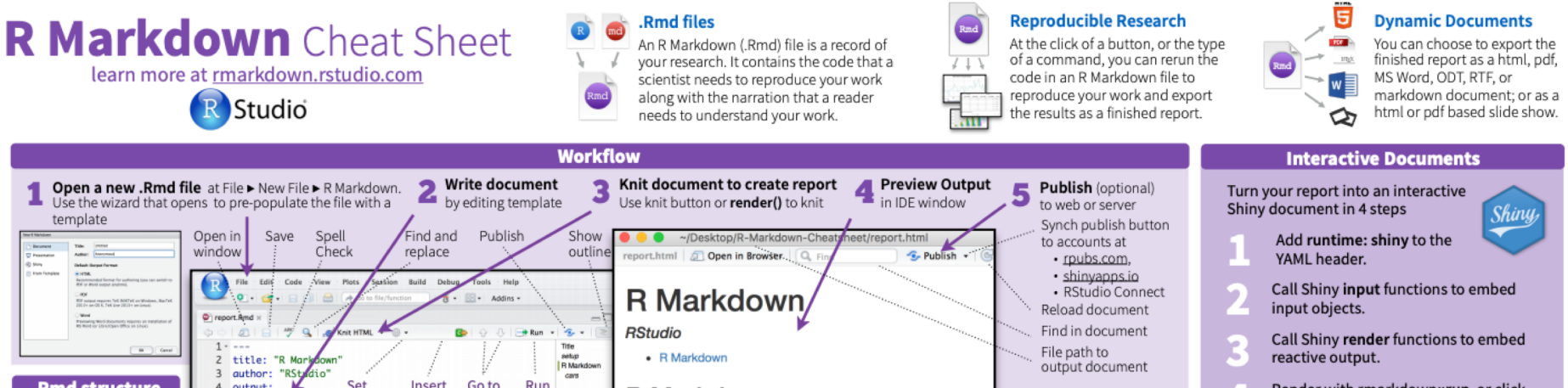
Workflow

Reading

- R4DS Chapter 21. R Markdown
- R4DS Chapter 2. Workflow: Basics
- R4DS Chapter 4. Workflow: Scripts
- R4DS Chapter 6. Workflow: Projects

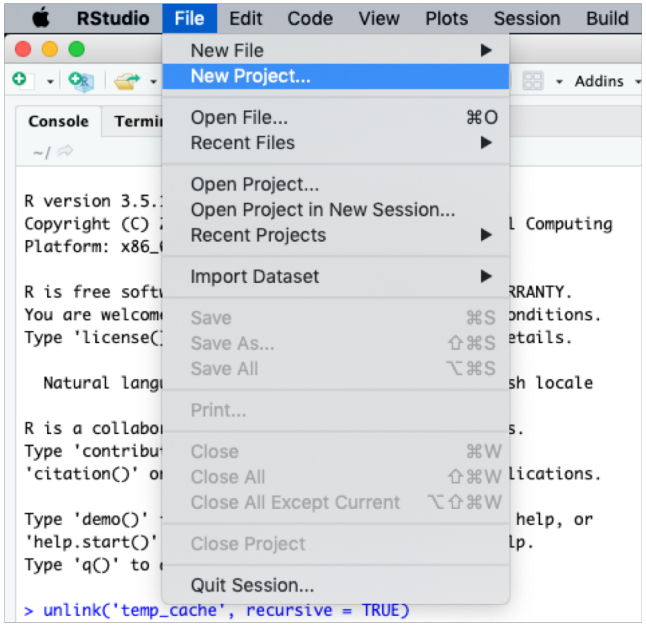
R Markdown

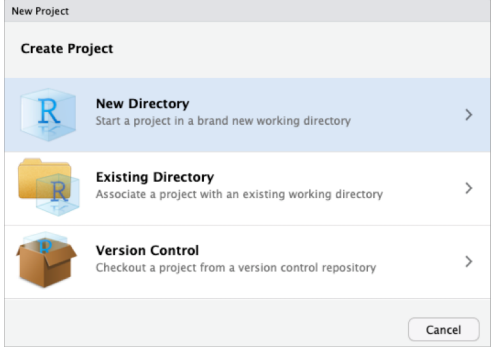
- I believe you already know how to write a report in R Markdown.
- If you need a refresher, read Chapter 21. R Markdown from R4DS.
- Or, visit https://rmarkdown.rstudio.com/lesson-1.html
- Cheat Sheet: https://www.rstudio.com/wp-content/uploads/2016/03/rmarkdown-cheatsheet-2.0.pdf

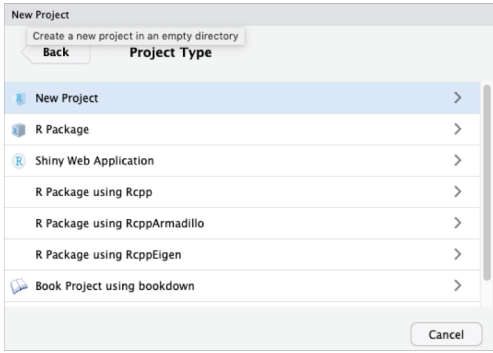


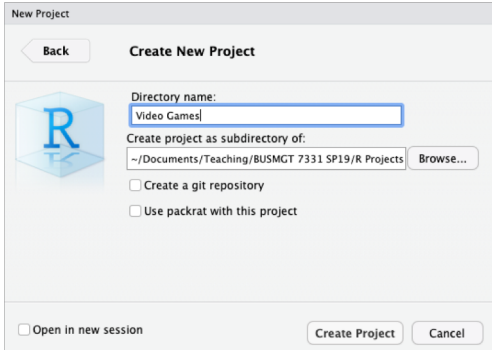
Project-based workflow

- Better way of organizing and keeping track of files (data, code, document) for a project
- Create a new project whenever you start in a new data analysis context.
- Start with a R Script file seems okay, but I also find it helpful to start a R Notebook.









R Notebook vs R Markdown

- R Notebook file is a R Markdown.
- The only difference is Notebook has the
  `output: html_notebook`
  line in the YAML header.
- When knitting an R Markdown document, the entire document is compiled.
  In an R Notebook, you can execute chunks independently,
  which makes the coding process more interactive.
- Read the following document for more information about R Notebook.
  https://bookdown.org/yihui/rmarkdown/notebook.html

Weekly Recap

Things we covered this week

- Course overview
  - Types of analytics
  - Why visualization?
  - Metacognition
- Data wrangling
  - Import
  - Tidy (gather, spread)
  - Transform (select, mutate, filter, arrange)
- Descriptive statistics
  - Tabulation for categorical & discrete variable
  - Binning + Tabulation for continuous & discrete variable
  - Summary statistics
  - Common discrete and continuous distributions
- Basic plots
  - For categorical and discrete: bar chart, stacked bar chart, radial chart, pie chart
  - For continuous and discrete: histogram, line plot, density plot, box plot, violin plot
  - Characterizing distributions
  - (Layered) grammar of graphics
- Workflow
  - R Markdown
  - Project-based workflow
  - R Notebook

Things to do this week

- 4 Concept Checker Quizzes (Due: Thursday, January 10, 11:59pm)
  - Week 1.1. Course Overview
  - Week 1.2. Data Wrangling
  - Week 1.3. Descriptive Statistics
  - Week 1.4. Basic Plots
- 1 Weekly Problem (Due: Friday, January 11, 11:59pm)
- 1 Bi-weekly Assignment (Due: Saturday, January 12, 11:59pm)
- 3 DataCamp Courses (Due: Friday, January 18, 11:59pm)
  - Data Visualization with R
    - [1] Data Visualization with ggplot2 (Part 1)
    - [2] Data Visualization with ggplot2 (Part 2)
    - [3] Visualization Best Practices in R