# Lecture 2:
# The Richness of Data

Prof. Martin Shepperd



Image courtesy of Jonathan Weicher

# Lecture Poll

Please use your mobile or computer to go to:

**pollev.com/mshepperd**

We will need it in a few minutes. Thanks!

# Lecture protocol

1. I will start at **1105 prompt**; please be ready
2. Be aware, lectures will be recorded
3. Feel free to ask **questions** as we go along or ...
4. ... ask during in a question gap (2-3 per lecture)
5. Be **considerate** of others (fellow students and me) and don't chat.
6. Thanks!

# Agenda

1. The idea of rich data
2. Data sources
3. Complex data structures in R
4. Files in R
5. Databases and R
6. Very large data sets
7. Week 2 goals

# 1. The idea of rich data

Multiple dimensions:
- **V**olume
- **V**ariety
- **V**elocity
- **V**eracity

# Velocity of data

## Wisdom of crowds



Source: Kozalov and Valkov, https://

# Veracity of data

## Wisdom of crowds

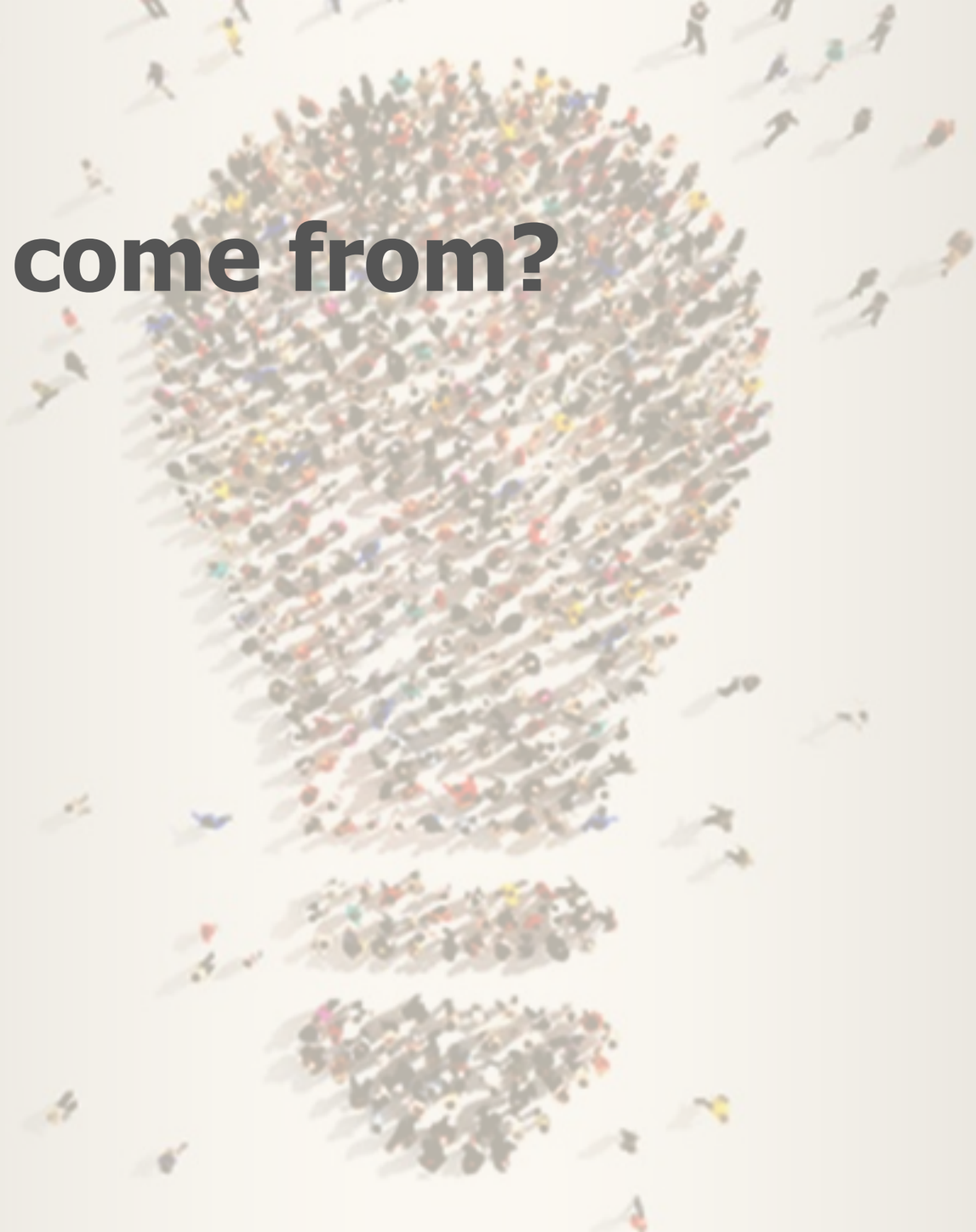


Source: Kozalov and Valkov, https://

# 2. Data sources

# Where does data come from?

## Wisdom of crowds





Source: Kozalov and Valkov, https://

# 3. Complex data structures in R

3.1 Data types recap
3.2 Data frames

# 3.1 Data types (of variables)

— A variable is a named **container** for data; it can be set, modified or referenced.

— R determines the data type from what you assign/force.

```
> n <- 10L          # The L (for Long) makes an integer type
> is.numeric(n)
[1] TRUE
> is.integer(n)
[1] TRUE
> n <- 10.1         # Coerce from integer to numeric
> is.integer(n)
 [1] FALSE
```

# Data types matter

```
> # Make a logical vector x
> # T is short for TRUE, F for FALSE
> x <- c(T, T, T, F, F)
> min(x)
```

## Wisdom of crowds: What do you think happens?

Sometimes we can get unxpected results! So be careful with types.

# Some operations are polymorphic [0]

```
> # NB there's no built in mode function to Base R
> library(modeest)
> # Or you can specify the library as
> # modeest::mlv(x)
> mlv(x)
 [1] TRUE
```

[0] Polymorphism describes the situation where you can access objects of different types through the same function and its implementation will depend upon the target type.

# Data and scales

Classical measurement theory [1].

1. **Nominal**, e.g., country or sex
2. **Ordinal** e.g., rank or Likert scale
3. **Interval** e.g., centigrade
4. **Ratio** e.g., length or degrees Kelvin
5. (Absolute e.g., counts)

[1] Stevens, S. S. (1946). On the theory of scales of measurement. Science, 103, 677-680.

# Data classification and data type

1. Categorical
   — Nominal (restricted) - **factor** e.g., sex
   — Nominal (unrestricted) - **character** e.g., tweet
   — Binary - **logical**, special case of nominal
2. Numeric
   — Ordinal - assume(?) equal intervals e.g., Likert scale
   — Interval+ratio - treat as **numeric** type
   — Absolute - non-negative integer (**L** in R!)
3. Audio, image, video, ...

# Repeating groups[2]

— vector (repeating group of the same data type)
— more generally ...

Table of more complex data structures in R

| Dimensions | Homogeneous type | Heterogeneous types |
|---|---|---|
| 1 | vector | list |
| 2 | matrix | data frame |
| n | array | n.a. |

[2] See the Modern Data book Chapter 2.4.

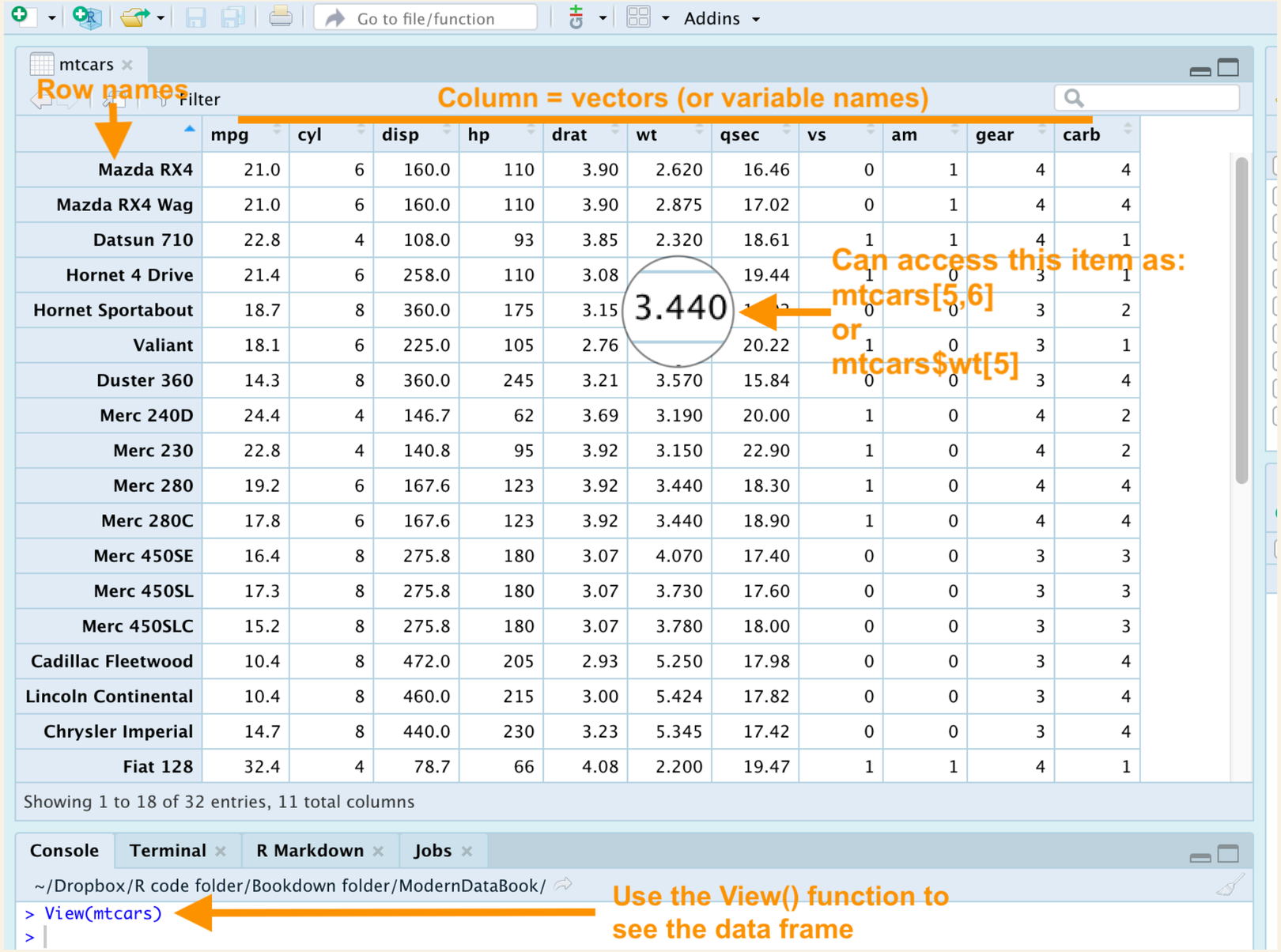
# 3.2 Data frames

— The data frame is one of the most useful data structures in R

— Analgous to a spreadsheet but you can't directly edit

— Easy to `View()` and manipulate

— Efficient as each column is a vector so simple code and fast processing




| | mpg | cyl | disp | hp | drat | wt | qsec | vs | am | gear | carb |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Mazda RX4 | 21.0 | 6 | 160.0 | 110 | 3.90 | 2.620 | 16.46 | 0 | 1 | 4 | 4 |
| Mazda RX4 Wag | 21.0 | 6 | 160.0 | 110 | 3.90 | 2.875 | 17.02 | 0 | 1 | 4 | 4 |
| Datsun 710 | 22.8 | 4 | 108.0 | 93 | 3.85 | 2.320 | 18.61 | 1 | 1 | 4 | 1 |
| Hornet 4 Drive | 21.4 | 6 | 258.0 | 110 | 3.08 | | 19.44 | 1 | 0 | 3 | 1 |
| Hornet Sportabout | 18.7 | 8 | 360.0 | 175 | 3.15 | 3.440 | | 0 | 0 | 3 | 2 |
| Valiant | 18.1 | 6 | 225.0 | 105 | 2.76 | | 20.22 | 1 | 0 | 3 | 1 |
| Duster 360 | 14.3 | 8 | 360.0 | 245 | 3.21 | 3.570 | 15.84 | 0 | 0 | 3 | 4 |
| Merc 240D | 24.4 | 4 | 146.7 | 62 | 3.69 | 3.190 | 20.00 | 1 | 0 | 4 | 2 |
| Merc 230 | 22.8 | 4 | 140.8 | 95 | 3.92 | 3.150 | 22.90 | 1 | 0 | 4 | 2 |
| Merc 280 | 19.2 | 6 | 167.6 | 123 | 3.92 | 3.440 | 18.30 | 1 | 0 | 4 | 4 |
| Merc 280C | 17.8 | 6 | 167.6 | 123 | 3.92 | 3.440 | 18.90 | 1 | 0 | 4 | 4 |
| Merc 450SE | 16.4 | 8 | 275.8 | 180 | 3.07 | 4.070 | 17.40 | 0 | 0 | 3 | 3 |
| Merc 450SL | 17.3 | 8 | 275.8 | 180 | 3.07 | 3.730 | 17.60 | 0 | 0 | 3 | 3 |
| Merc 450SLC | 15.2 | 8 | 275.8 | 180 | 3.07 | 3.780 | 18.00 | 0 | 0 | 3 | 3 |
| Cadillac Fleetwood | 10.4 | 8 | 472.0 | 205 | 2.93 | 5.250 | 17.98 | 0 | 0 | 3 | 4 |
| Lincoln Continental | 10.4 | 8 | 460.0 | 215 | 3.00 | 5.424 | 17.82 | 0 | 0 | 3 | 4 |
| Chrysler Imperial | 14.7 | 8 | 440.0 | 230 | 3.23 | 5.345 | 17.42 | 0 | 0 | 3 | 4 |
| Fiat 128 | 32.4 | 4 | 78.7 | 66 | 4.08 | 2.200 | 19.47 | 1 | 1 | 4 | 1 |

# Data frames and data type

— Very easy to import files into data frames

— Think about the variables (columns) e.g., use `head()` or `str()` to examine them

— Be careful about numeric values/levels for factors **(why?)**

# 3.3 Text

The "golden age" of text processing or natural language processing (NLP). [3]


[3] A worked example in R or for a very detailed book "Text Mining with R: A Tidy Approach" by David Robinson and Julia Silge (2017).

# 4. Files in R

— Very often, the data we need to analyse is stored in external files.

— Frequently as comma-separated variable (CSV) files

— Many built in functions to help e.g., `read.csv()`

— Great deal of flexibility

```r
# R code to read a remote file (on GitHub) into a data frame pubsDataFrame
# Because the path url is quite long I've first copied it to a character string fname
# to improve readability

fname <- "https://raw.githubusercontent.com/mjshepperd/CS5702-Data/master/pubs.csv"
pubsDataFrame <- read.csv(fname, header = TRUE, stringsAsFactors = FALSE)

head(pubsDataFrame)
```

```
249 head(pubsDataFrame)
250 ```
```

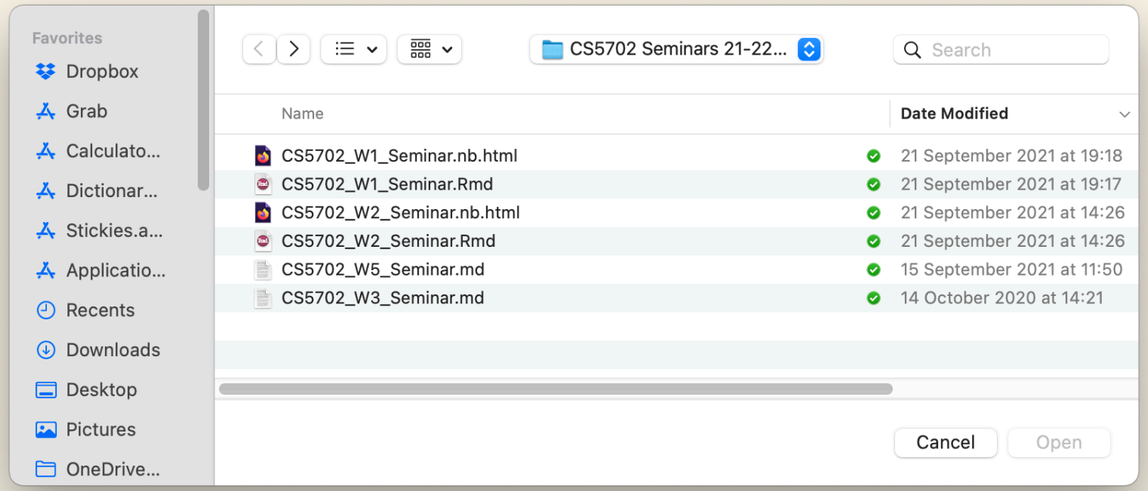
| | pubName <chr> | open <lgl> | town <chr> | weeklySales <int> | foodSales <int> |
|---|---|---|---|---|---|
| 1 | The Dead Albatross | TRUE | Uxbridge | 2735 | 1209 |
| 2 | The Island Queen | TRUE | Islington | 3644 | 0 |
| 3 | Johnnys Bar | FALSE | Vladivostok | 0 | 0 |
| 4 | Red Lion | TRUE | Habrough | 3263 | *NA* |
| 5 | The Crown | FALSE | Haccombe | 0 | 0 |
| 6 | Royal Oak | FALSE | Haceby | 0 | 0 |

6 rows

```
251
```

# A useful file reading tip

```
# The user is prompted
# to locate the file
myDF <- read.csv(file.choose())
```



# Writing files

— It's also useful to output files

— Analogous to the `read.csv()` function is the `write.csv()`

— Don't forget the **append** argument (False means if the file already exists it will be overwritten)

```
write.csv(myDF, "mydata.csv", sep=",", append = FALSE)
```

# 5. Databases and R

— Sometimes data sets are stored as databases, e.g., relational database such as SQLite, MySQL or Oracle

— Typically collections of tables that need to be organised, into a single rectangular data frame

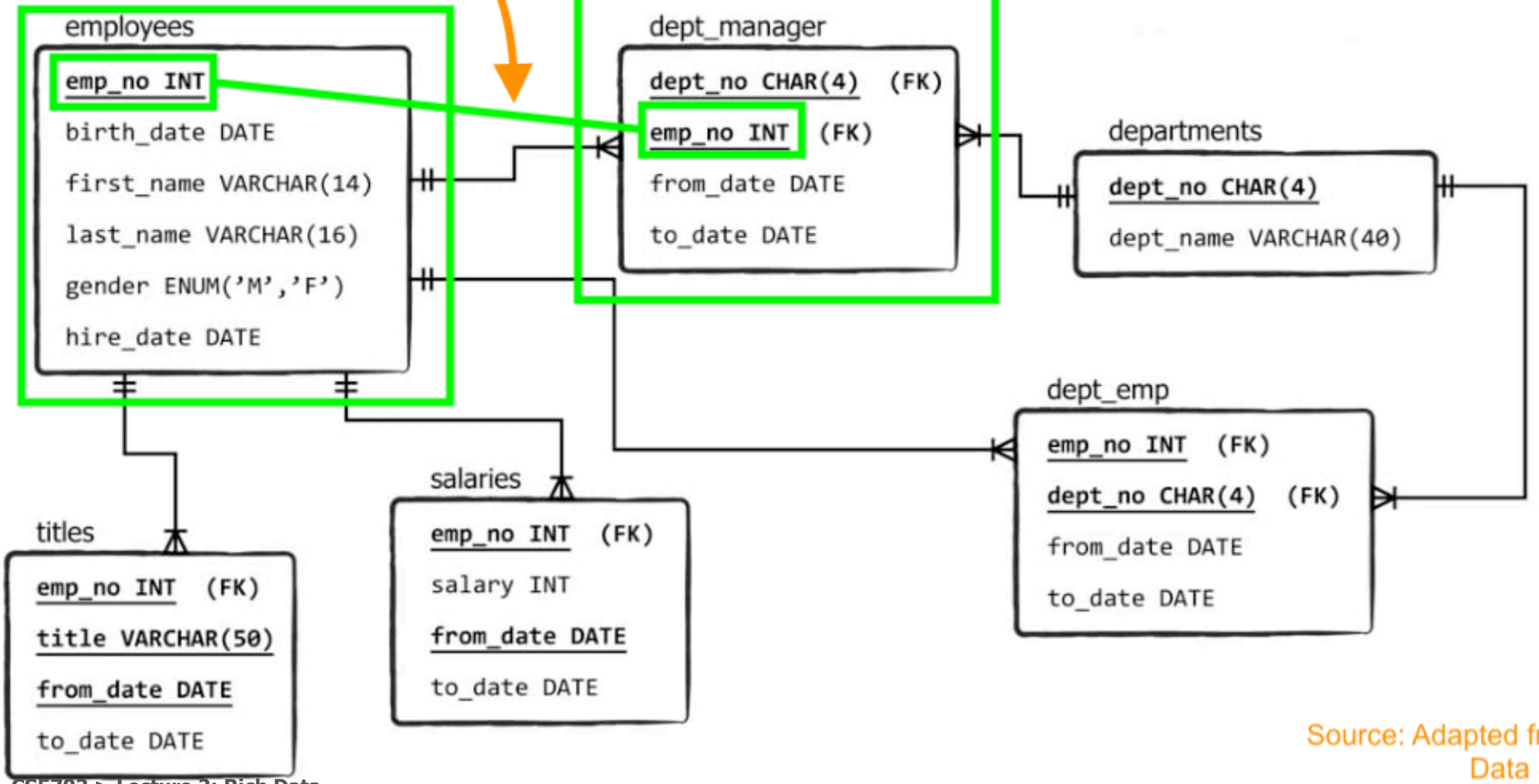
— helps manage very large volumes of data

— offers the possibility of only accessing data as and when it is needed

— R offers multiple approaches including embedding sql within your R code

# Database: employees

Joining database tables via Foreign Keys




Source: Adapted from 365 Data Science

# 6. Very large data sets

— over last 10 years growing interest in so-called "Big Data"

— now think in zettabytes rather than exabytes

— but challenges lie in data complexity as well as volume

# R and large data set restrictions

1. R can only process data that fits into your computer's memory, ~4-16Gb
2. but you will be manipulating your data, so a good rule of thumb: **twice the amount of RAM** that the data occupies
3. R reads entire data set into RAM all at once: time to pull a very large data set into memory can be far slower than executing the analysis
4. There is a two 2 billion vector index limit

# Big data solutions

1. specialised packages (actually implemented in C++) e.g., {bigmemory} that enable access the data set without reading the entire set into R.
2. parallelisation using some cloud facility or use a hadoop distributed system via the {RHadoop} package.

Nevertheless, our focus in Modern Data is on 'small', in-memory datasets.

# 7. Week 2 goals

By the end of this week you should:
[ ] have an appreciation of the richness of data e.g. list different characteristics and types
[ ] be familiar with some of the major sources of free and publicly available data
[ ] use R to create and manipulate vector and data frame variables
[ ] use R to import and export csv files
[ ] complete the Week 2 worksheet and quiz

# Further Reading

1. For more details on R basics, file handling and databases see the Modern Data, chapter two
2. Alternatively see chapters 1, 2 and 4 from Kabacoff (2015) R in Action: Data Analysis and Graphics with R
3. Grolemund and Hadley (2018) R for Data Science, Chapter 13 for more details of working with relational databases such as MySQL and sqlite in R

# Questions