Notes and Materials for Data Computing

Daniel Kaplan

2016-10-13

Contents

This book contains class notes, activities, and projects used in the Data Computing class. Often, there are more activities and projects than we actually used.

The *Data Computing* textbook has 18 chapters. These notes are divided into 8 parts, each of which corresponds to multiple chapters the textbook. This reflects the division of the course itself into eight components.

This is very much a work in progress. It's not yet complete and what's here may have many errors. Please do point out the problems so that these notes evolve in a positive way.

Chapter 1

(PART) Getting Organized

The materials from Week 1 will go here.

1. Getting to RStudio
2. Connecting to GitHub
3. RMarkdown
    - Writing a simple RMarkdown document

One-page cheat sheet.

Chapter 2

Files and Documents

2.1 The "file path"

The "file path"" is the set of successive folders that bring you to the file.

There is a standard format for file paths. An example: `/Users/kaplan/Downloads/0021_001.pdf`. Here the filename is `0021_001.pdf`, the filename extension is `.pdf`, and the file itself is in the Downloads folder contained in the kaplan folder, which is in turn contained in the Users folder. The starting / means "on this computer".

2.2 Using a file browser to find the path

Many people are used to finding files interactively with the mouse and may not be aware of the path to the directory holding the file.

The R `file.choose()` — which should be used only in the console, not in an Rmd file — brings up an interactive file browser. You can select a file with the browser. The returned value will be a quoted character string with the path and file name.

`file.choose()` then select a file. The output is: `[1] "/Users/kaplan/Downloads/0021_001.pdf"`

For example I have a `.csv` file on my Desktop called `names.csv`. If I want to load it into R I might first find the path:

```
file.choose()
```

```
## [1] "/Users/Adam/Desktop/names.csv"
```

Then to load it I type:

```
my_names <- read.file("/Users/Adam/Desktop/names.csv")
```

Some common filename extensions for the sort of web resources you will be using:

- `.png` for pictures
- `.jpg` or `.jpeg` for photographs
- `.csv` or `.Rdata` for data files
- `.Rmd` for the human editable text of a document (called R Markdown)
- `.html` for web pages themselves

Credit: Adam Lucas

Chapter 3

Embeding Files within Rmd -> HTML files

When you produce an HTML document with R/Markdown, the commands can only be run by cutting-and-pasting them from the HTML into R. Often, you'd prefer to provide your reader with a copy of the original Rmd file itself.

The `DataComputing` package provides a way to do this easily. Include the following chunk in your document:

````
```{r results = "asis"}
DataComputing::includeSourceDocuments()
```
````

Notes_and_Other_Materials_for_Data_Computing.Rmd

This will produce a link to the Rmd document itself, like this.

Clicking on the link will (in most browsers) extract the Rmd file and install it in your Downloads directory.

You can embed multiple files in a document, which you may find convenient for CSV files, etc. When embedding a file other than the source Rmd file, give as an argument to `includeSourceDocuments()` the quoted character string containing the path and filename for your Rmd file. You can construct this easily by using `file.choose()` in the console, and copying the result as the argument of `includeSourceDocuments()` function. This will embed the named file into your HTML, so that the file will be available directly through the HTML file. You need to view the HTML in your browser to be able to download the included file.

Chapter 4

(PART) Data Infrastructure

Topics

1) The structure of tabular data
    - cases and variables
    - numerical and categorical variables
    - tidy data
2) R Commands
3) Files and documents

Chapter 5

Case Study: Highway Fatalities

On August 29, 2016, the White House issued a data-science "call to action."[1](https://www.transportation.gov/fastlane/2015-traffic-fatalities-data-has-just-been-released-call-action-download-and-analyze).

> Today, the U.S. Department of Transportation is releasing an open data set that contains detailed, anonymized information about each of these tragic incidents. As the new data being released show, and as DOT reported earlier this summer, 2015 showed a marked increase in traffic fatalities nationwide.
>
> To be precise, 7.2% more people died in traffic-related accidents in 2015 than in 2014. This unfortunate data point breaks a recent historical trend of fewer deaths occurring per year.
>
> Under the leadership of Transportation Secretary Anthony Foxx, we're doing two things differently this year.
>
> One: We're publishing the data through NHTSA's Fatality Analysis Reporting System (FARS) three months earlier than last year.
>
> Two: We're directly soliciting your help to better understand what these data are telling us. Whether you're a non-profit, a tech company, or just a curious citizen wanting to contribute to the conversation in your local community, we want you to jump in and help us understand what the data are telling us.
>
> Some key questions worth exploring:
>
> How might improving economic conditions around the country change how Americans are getting around? What models can we develop to identify communities that might be at a higher risk for fatal crashes? How might climate change increase the risk of fatal crashes in a community? How might we use studies of attitudes toward speeding, distracted driving, and seat belt use to better target marketing and behavioral change campaigns? How might we monitor public health indicators and behavior risk indicators to target communities that might have a high prevalence of behaviors linked with fatal crashes (drinking, drug use/addiction, etc.)?
>
> DOT is aggressively seeking ways to improve safety on the roads. From our work with the auto industry to improve vehicle safety, to new solutions to behavioral challenges like drunk, drugged, distracted and drowsy driving, we know we need to find novel solutions to old challenges.
>
> We're also looking to accelerate technologies that may make driving safer, including connected and highly automated vehicles.
>
> But we need your help, too! Data Science is a team sport.

[1]The call was cross-posted by the US Department of Transportation here

> We are calling on data scientists, public health experts, students and researchers—even if you have never thought about road safety before—to dive in to these data and help answer these important questions, especially on tough issues like pedestrian and bicyclist fatalities.
>
> Start by downloading and playing with the data. Then share your insights and let us know what you find by sending us a note at opendata@dot.gov.

5.1 The accident data

The link to the data in the call to action is ftp://ftp.nhtsa.dot.gov/fars/2015/.[2]

1. Go to that site. Is it immediately clear what's going on?
2. What can you figure out by browsing the site.
   - look in "parent" directories
   - try substituting 2015 for other similar sorts of values in the URL.

A blog by Lucas Puente conveniently gives simple instructions for downloading the data. He writes:

> Simply visit ftp://ftp.nhtsa.dot.gov/fars/2015/National/ and download the `FARS2015NationalDBF.zip` file, unzip it, and load into R.

After unzipping, there is a directory `FARS2015NationalDBF` taking up 874.9 MB of disk for 27 items. I put it in my `Downloads` directory.

Lucas provides commands to read the data into R.

```
library(foreign)
accidents <- read.dbf("FARS2015NationalDBF/accident.dbf")
```

To make sense of these instructions, it helps to know some things:

- What is `library(foreign)` doing?
  - What does the `library()` part of the command tell you?
  - What is `foreign`. How would get get some instructions or documentation for `foreign` to help you understand why this is appropriate.
- What is `read.dbf()`?
- What does `"FARS2015NationalDBF/accident.dbf"` tell you about the data file and where it's located.

Aside: I would rather you wrote:

```
filename <- "~/Downloads/FARS2015NationalDBF/accident.dbf"
Accidents <- foreign::read.dbf(filename)
```

What's different about the file name I'm using? Why?

Lucas's blog leads you through the steps of making a map of accident locations:

What does this map tell you?

With the `Accidents` data table read into R, it's easy to look at it and perhaps construct summaries.

```
##   STATE ST_CASE VE_TOTAL VE_FORMS PVH_INVL PEDS PERNOTMVIT PERMVIT
## 1     1   10001        1        1        0    0          0       1
## 2     1   10002        1        1        0    0          0       1
## 3     1   10003        1        1        0    0          0       2
## 4     1   10004        1        1        0    0          0       1
```

[2] Such an address is called a URL.

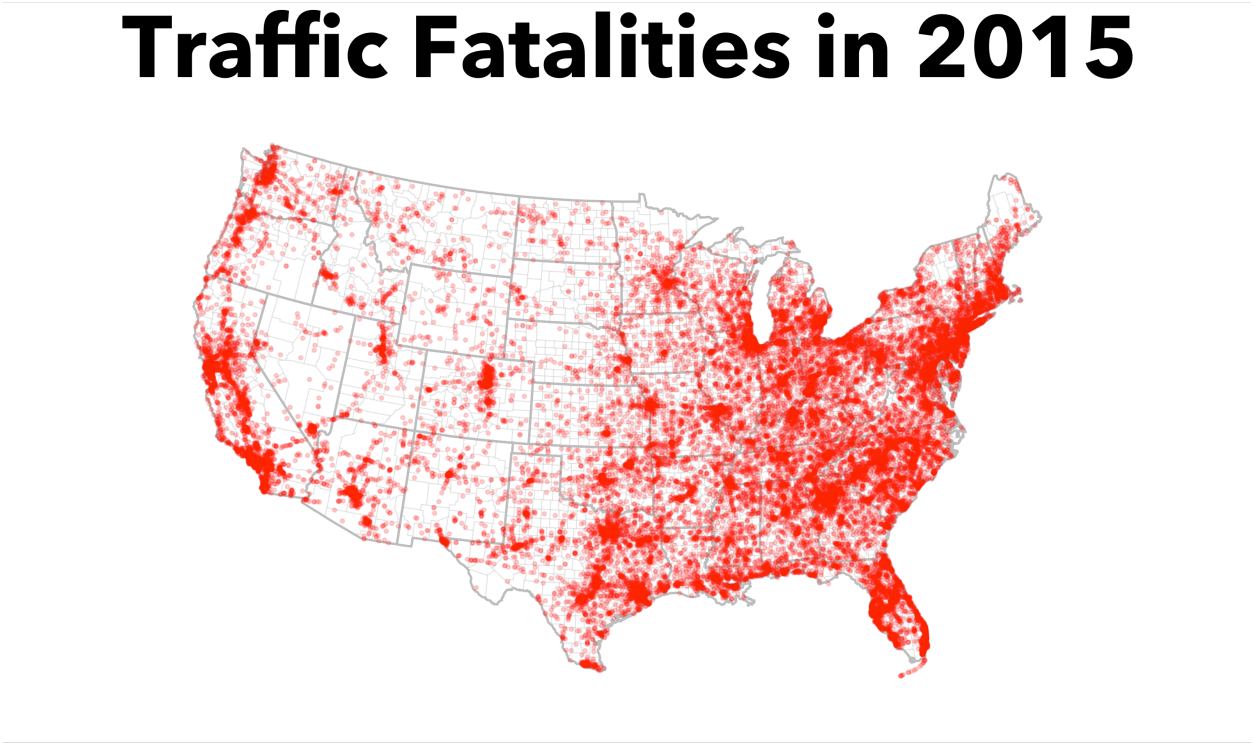


Figure 5.1: Lucas's map

Some simple things:

```
nrow(Accidents)
```

```
## [1] 32166
```

```
names(Accidents)
```

```
##  [1] "STATE"      "ST_CASE"    "VE_TOTAL"   "VE_FORMS"   "PVH_INVL"
##  [6] "PEDS"       "PERNOTMVIT" "PERMVIT"    "PERSONS"    "COUNTY"
## [11] "CITY"       "DAY"        "MONTH"      "YEAR"       "DAY_WEEK"
## [16] "HOUR"       "MINUTE"     "NHS"        "RUR_URB"    "FUNC_SYS"
## [21] "RD_OWNER"   "ROUTE"      "TWAY_ID"    "TWAY_ID2"   "MILEPT"
## [26] "LATITUDE"   "LONGITUD"   "SP_JUR"     "HARM_EV"    "MAN_COLL"
## [31] "RELJCT1"    "RELJCT2"    "TYP_INT"    "WRK_ZONE"   "REL_ROAD"
## [36] "LGT_COND"   "WEATHER1"   "WEATHER2"   "WEATHER"    "SCH_BUS"
## [41] "RAIL"       "NOT_HOUR"   "NOT_MIN"    "ARR_HOUR"   "ARR_MIN"
## [46] "HOSP_HR"    "HOSP_MN"    "CF1"        "CF2"        "CF3"
## [51] "FATALS"     "DRUNK_DR"
```

How to figure out what each variable means? Browse around the FARS server to see if you can find something that might help.

I found a codebook here. (Just in case the FARS website changes, here's a copy downloaded on 9/7/2016.)

Let's figure out what `CF1` means. How about `WEATHER2` and `REL_LOAD`?

5.2 Other tables

But there are other data files in the database.

```
dir("~/Downloads/FARS2015NationalDBF/")
```

```
##  [1] "ACC_AUX.dbf"   "accident.dbf"  "cevent.dbf"    "Damage.dbf"
##  [5] "Distract.dbf"  "DrImpair.dbf"  "Factor.dbf"    "Maneuver.dbf"
##  [9] "MIACC.dbf"     "MIDRVACC.dbf"  "MIPER.dbf"     "nmcrash.dbf"
## [13] "NMImpair.dbf"  "NMPrior.dbf"   "parkwork.dbf"  "PBType.dbf"
## [17] "PER_AUX.dbf"   "person.dbf"    "SafetyEq.dbf"  "VEH_AUX.dbf"
## [21] "vehicle.dbf"   "VEvent.dbf"    "VINDecode.dbf" "Violatn.dbf"
## [25] "Vision.dbf"    "VSOE.dbf"
```

Let's look at some of them:

```
head(Vision)
```

```
##   STATE ST_CASE VEH_NO MVISOBSC
## 1     1   10001      1        0
## 2     1   10002      1        0
## 3     1   10003      1        0
## 4     1   10004      1        0
## 5     1   10005      1        2
## 6     1   10005      2        0
```

```
head(Distract)
```

```
##   STATE ST_CASE VEH_NO MDRDSTRD
## 1     1   10001      1       99
## 2     1   10002      1        0
## 3     1   10003      1        0
## 4     1   10004      1       99
```

```
## 5     1   10005      1       99
## 6     1   10005      2        0
```

```
head(DrImpair)
```

```
##   STATE ST_CASE VEH_NO DRIMPAIR
## 1     1   10001      1        0
## 2     1   10002      1        0
## 3     1   10003      1        9
## 4     1   10004      1        9
## 5     1   10005      1       98
## 6     1   10005      2        0
```

What's the connection between these tables and the `Accidents` table? Say, how would we be able to see which weather conditions distracted driving accidents tend to occur in?

Chapter 6

Case Study: Taxicabs and the sharing economy

A team of mathematicians and engineers has calculated that if taxi riders were willing to share a cab, New York City could reduce the current fleet of 13,500 taxis up to 40 percent. Link to news story and an interactive site with the data.

Chapter 7

Case Study: Medicare spending

Newspaper article here

Data available here.

DTK notes

Chapter 8

Untidy data: School enrollments

The US Census Bureau collects data on many aspects of the population. Data on school enrollments is available here. We're going to look at one of the data tables they make available:

> Table 2: Single Grade of Enrollment and High School Graduation Status for People 3 Years Old and Over, by Sex, Age (Single Years for 3 to 24 Years), Race, and Hispanic Origin: October 2014
> XLS or CSV format.

Download one of these files and open it in appropriate software. Or you can view the data on Google Drive here.

1. How many people are represented in this data table?
2. The table is in some ways a graphical visualization of features of school enrollment and age. (Unfocus your eyes and you will see a visual pattern.) What patterns do you see?
3. The table indicates that 74.4% of the people in the table are "not enrolled" in school. Figure out how to calculate this from the numbers in the table. (Hint: You need only look at line 9.)
4. These data are "untidy" in a technical sense. Identify the ways that they are untidy.
5. Some columns contain information that can be calculated from other columns. Look at the data for 4-year olds and identify those that are calculated from other columns. Figure out the minimal set of columns from which the others could be calculated.
6. Imagine that this table was created from a much bigger table in which each case is an individual person in the US.
   - How many cases would there be in that table?
   - What variables would you need so that you could calculate any entry in the "Table 2" provided by the Census Bureau?

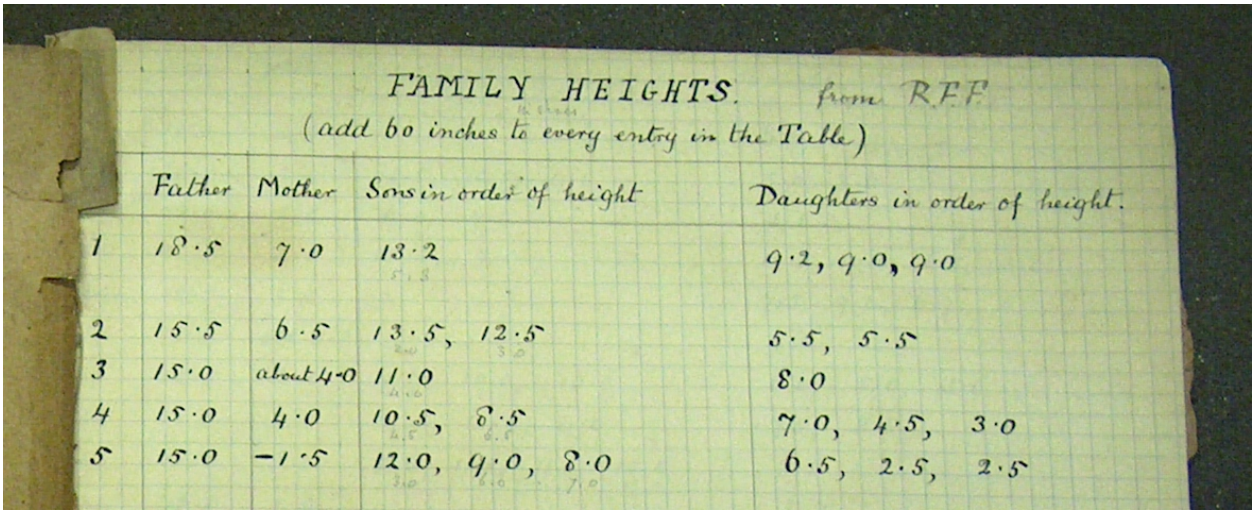
Chapter 9

Untidy data: Galton's measurements of height

In the 1880s, Francis Galton started to make a mathematical theory of evolution. But the basic biology of heritability was not known: nothing about "genes" or DNA, etc.

In order to create a theory, Galton needed a way to measure how traits are inherited from parents. To this end, he visited families in London and measured the heights of the parents and their (adult) children.

Here's part of a page from his lab notebook.



FAMILY HEIGHTS. from R.F.F.

(add 60 inches to every entry in the Table)

	Father	Mother	Sons in order of height	Daughters in order of height.
1	18·5	7·0	13·2	9·2, 9·0, 9·0
2	15·5	6·5	13·5, 12·5	5·5, 5·5
3	15·0	about 4·0	11·0	8·0
4	15·0	4·0	10·5, 8·5	7·0, 4·5, 3·0
5	15·0	−1·5	12·0, 9·0, 8·0	6·5, 2·5, 2·5

Figure 9.1: A page from Francis Galton's notebook.

Divide into groups of 2 or 3 and translate the notebook data into a tidy form.

- Think about what would be an appropriate "case" for storing this data.
- What variables should there be?
- Open a spreadsheet and fill in a couple of rows of the tidy table that you envision.

Here are a few that have already been made:Group-1, Group-2, Group-3, Group-4, Group-5, Group-6

Chapter 10

Untidy data: Family structure of military personnel

This spreadsheet contains is a presentation of data about family structure in the US Armed Forces.

1. How many military personnel (not their children) are represented in this data table?
2. What is a case in this data table?
3. These data are "untidy" in a technical sense. Identify the ways that they are untidy.
4. Some rows contain information that could be calculated from other rows. Identify these.
5. Some "tabs" contain information that could be calculated from the other tabs. Identify these.
6. Some columns contain information that can be calculated from other columns. Figure out the minimal set of columns from which the others could be calculated.
7. Imagine that this table was created from a much bigger table in which each case is an individual person.
    - How many cases would there be in that table?
    - What variables would you need so that you could calculate any entry in the table linked to above. Divide into groups and fill in a few rows of the table you created. Here are a few that have already been made:Group-1, Group-2, Group-3, Group-4, Group-5, Group-6

Chapter 11

Untidy data: Minneapolis Voting

The spreadsheet here contains data on the Minneapolis 2013 election by ward and precinct.

1. What is the case here?
2. How are the data not tidy?
3. What might these data look like in tidy form?
4. The data table `DataComputing::Minneapolis2013` lists the choices on individual ballots. What is the case?
5. The cases in `DataComputing::Minneapolis2013` can be aggregated to produce *some* of the variables in the spreadsheet.
    a. Which variables in the spreadsheet *cannot* be recreated from an aggregation of the ballot data? (Background on the voting law: to vote, a person must be registered in advance or do so at the polling place. Votes can be made at the polling place or, for a voter who is away, by mail as an absentee. Some ballots are not legible or otherwise violate voting rules; these are called "spoiled.")
    b. Imagine a data table like `DataComputing::Minneapolis2013` that could be aggregated to produce the variables in the spreadsheet. What would the cases be in that table?

Chapter 12

Tidy Data

12.1 Data has all sorts of forms

12.1.1 Signals

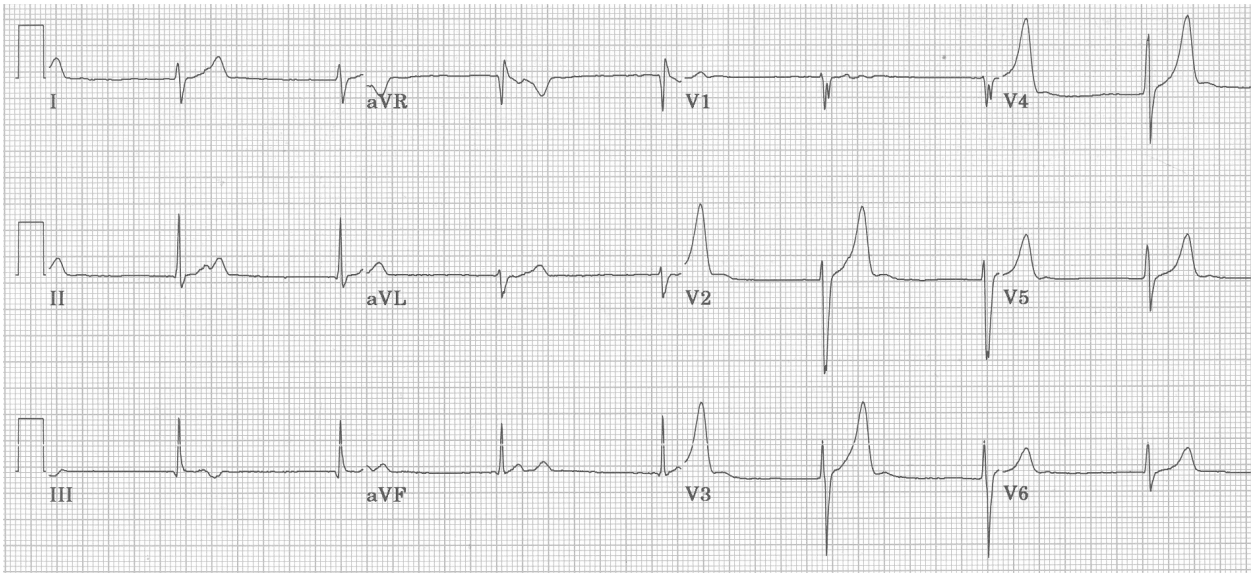


Figure 12.1:

12.1.2 Photographs

12.1.3 Video

Follow this link!

12.1.4 Text, e.g. What am I doing? on OKCupid

> currently working as an international agent for a freight forwarding company. import, export, domestic you know the works

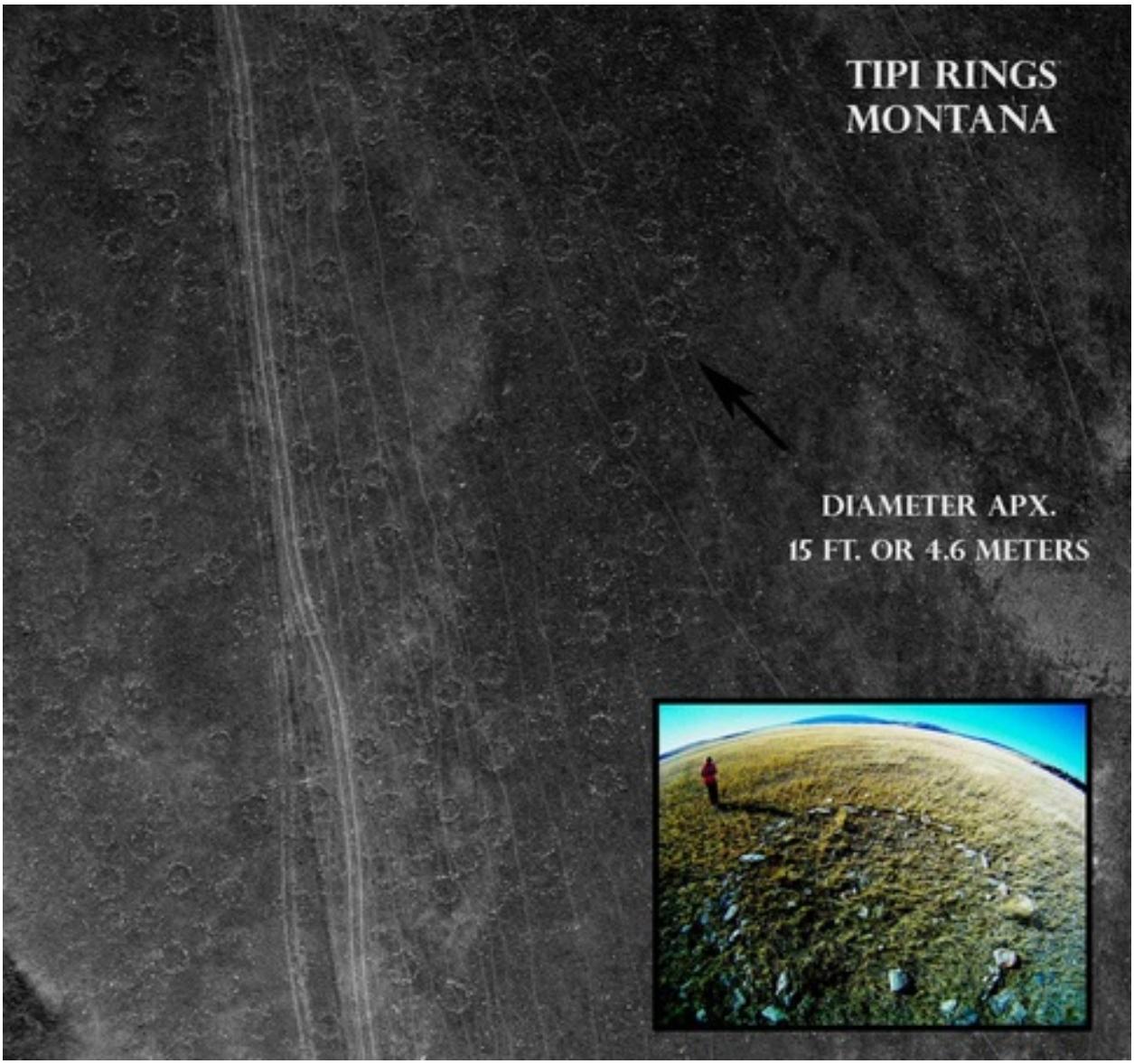


Figure 12.2:

online classes and trying to better myself in my free time. perhaps a hours worth of a good book or a video game on a lazy sunday."

dedicating everyday to being an unbelievable badass.

i make nerdy software for musicians, artists, and experimenters to indulge in their own weirdness, but i like to spend time away from the computer when working on my artwork (which is typically more concerned with group dynamics and communication, than with visual form, objects, or technology). i also record and deejay dance, noise, pop, and experimental music (most of which electronic or at least studio based). besides these relatively ego driven activities, i've been enjoying things like meditation and tai chi to try and gently flirt with ego death."

reading things written by old dead people

work work work work + play

building awesome stuff. figuring out what's important. having adventures. looking for treasure. digging up buried treasure

12.1.5 Sequences

Figure 12.3:

12.2 Data Tables

We're going to use just one very simple format: the *data table*.

name	sex	count	year
Rotha	F	7	1907
Julian	M	535	1948
Christina	M	22	1967
Song	F	11	1994
Wayman	M	9	1997

12.3 Conversion from images, videos, etc. to data table

12.3.1 OK Cupid

Sentiment extraction

12.3.2 Tipi rings in Montana

Assessment on family size based on tipi ring diameter

Population size by adding up the rings

12.3.3 Animal tracking

12.4 Cases and Variables

12.4.1 Anatomy of a data table

- A row is always a **case**
- A column is always a **variable**

12.5 What's a variable?

A quantity or category that may vary from case to case.

Two main types:

1. Quantitative: a number
2. Categorical: one of a set of discrete possibilities

12.6 Not in Tidy Data

- No units
- No footnotes

Instead, this information should go into a codebook.

Values and Cases need to be commensurate

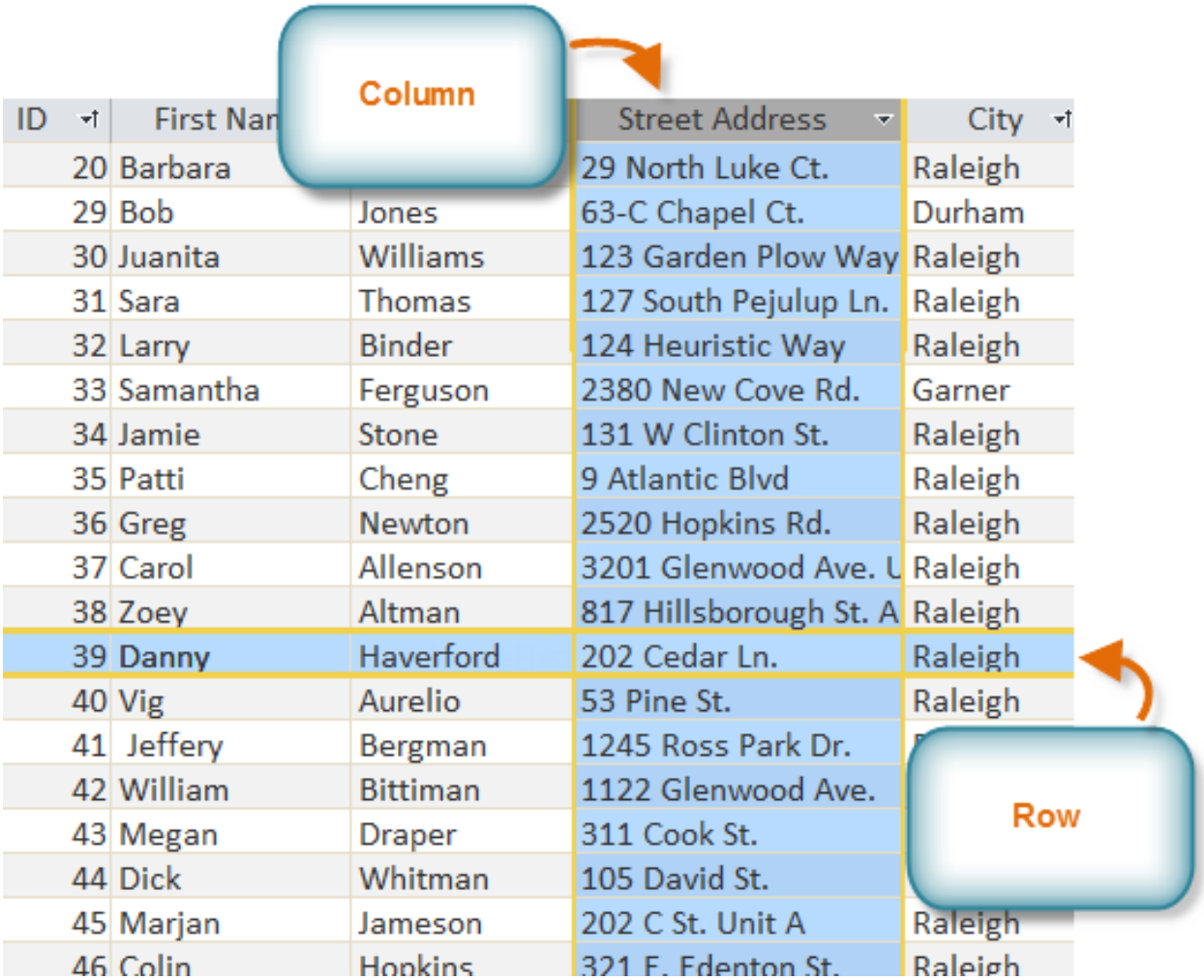
- Same kind of thing for each case, e.g. don't mix miles and km.
- Within a variable, only the same kind of value for each case.

12.7 Cases

The **object** from which the variables were measured.

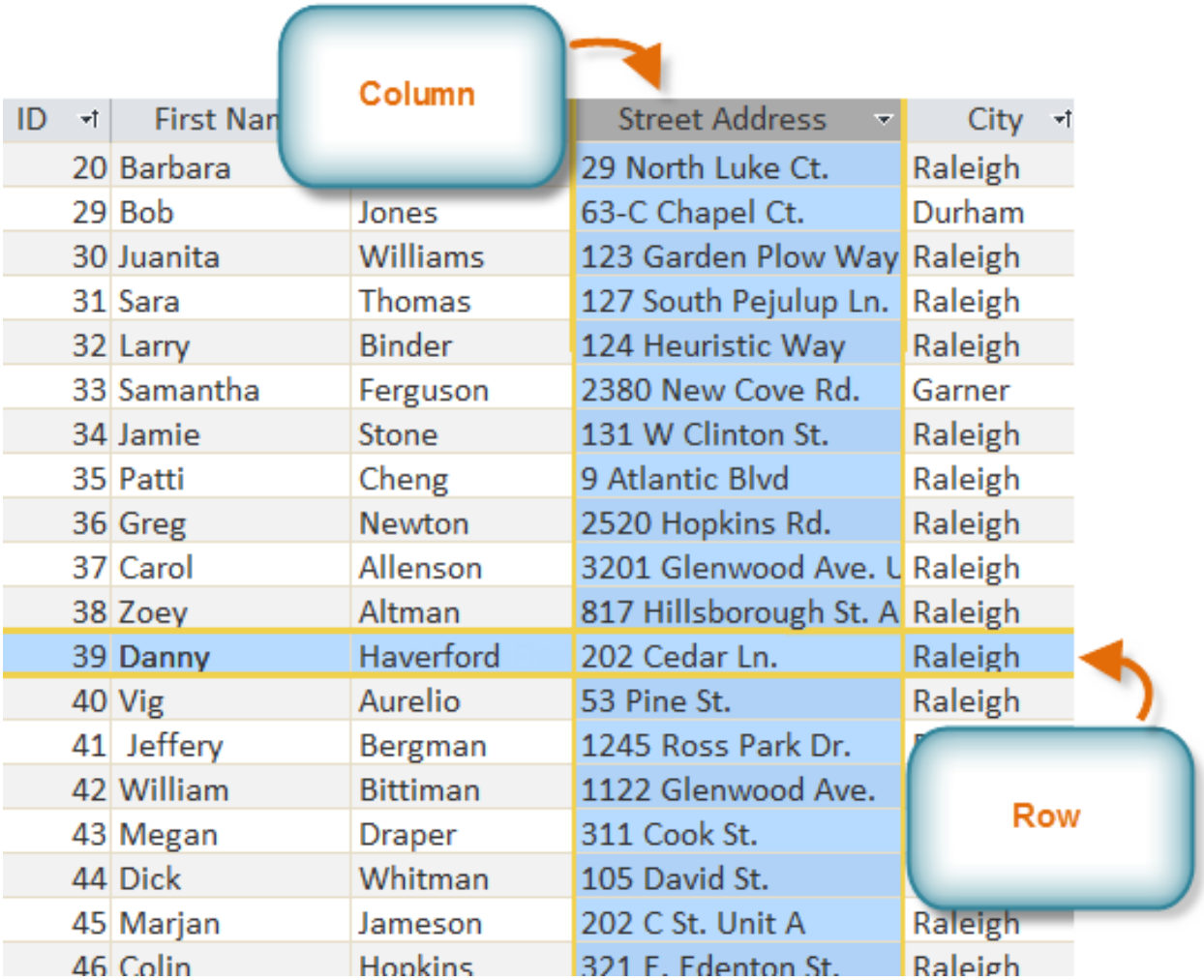
Examples:

- A person, a country, an earthquake, a bike rental
- A person on a date
- A country in a year
- An earthquake and its aftershocks



ID	First Nar		Street Address	City
20	Barbara		29 North Luke Ct.	Raleigh
29	Bob	Jones	63-C Chapel Ct.	Durham
30	Juanita	Williams	123 Garden Plow Way	Raleigh
31	Sara	Thomas	127 South Pejulup Ln.	Raleigh
32	Larry	Binder	124 Heuristic Way	Raleigh
33	Samantha	Ferguson	2380 New Cove Rd.	Garner
34	Jamie	Stone	131 W Clinton St.	Raleigh
35	Patti	Cheng	9 Atlantic Blvd	Raleigh
36	Greg	Newton	2520 Hopkins Rd.	Raleigh
37	Carol	Allenson	3201 Glenwood Ave. U	Raleigh
38	Zoey	Altman	817 Hillsborough St. A	Raleigh
39	Danny	Haverford	202 Cedar Ln.	Raleigh
40	Vig	Aurelio	53 Pine St.	Raleigh
41	Jeffery	Bergman	1245 Ross Park Dr.	
42	William	Bittiman	1122 Glenwood Ave.	
43	Megan	Draper	311 Cook St.	
44	Dick	Whitman	105 David St.	
45	Marjan	Jameson	202 C St. Unit A	Raleigh
46	Colin	Hopkins	321 E. Edenton St.	Raleigh

Figure 12.4:



ID	First Nar		Street Address	City
20	Barbara		29 North Luke Ct.	Raleigh
29	Bob	Jones	63-C Chapel Ct.	Durham
30	Juanita	Williams	123 Garden Plow Way	Raleigh
31	Sara	Thomas	127 South Pejulup Ln.	Raleigh
32	Larry	Binder	124 Heuristic Way	Raleigh
33	Samantha	Ferguson	2380 New Cove Rd.	Garner
34	Jamie	Stone	131 W Clinton St.	Raleigh
35	Patti	Cheng	9 Atlantic Blvd	Raleigh
36	Greg	Newton	2520 Hopkins Rd.	Raleigh
37	Carol	Allenson	3201 Glenwood Ave. U	Raleigh
38	Zoey	Altman	817 Hillsborough St. A	Raleigh
39	Danny	Haverford	202 Cedar Ln.	Raleigh
40	Vig	Aurelio	53 Pine St.	Raleigh
41	Jeffery	Bergman	1245 Ross Park Dr.	
42	William	Bittiman	1122 Glenwood Ave.	
43	Megan	Draper	311 Cook St.	
44	Dick	Whitman	105 David St.	
45	Marjan	Jameson	202 C St. Unit A	Raleigh
46	Colin	Hopkins	321 E. Edenton St.	Raleigh

Figure 12.5:

12.8 Basic Knowledge

1. What is each variable about.
2. What is the kind of object that defines a case

12.9 Tidy Data

Every value for each variable is the same kind of thing as all the other values for that variable.

Every case is the same kind of thing as all the other cases.

12.10 Workflow: Creating a chain of evidence

It's important to be able to state definitely where your data came from.

Part of this is not to edit your data. Once you have a table, don't change anything in it.

Instead, do your data-transformations in R so that you have a complete statement of how the data you collected are related to your analysis.

12.11 Summary

- Data Table: Rectangular format: cases (rows) and variables (columns)
- Separate analysis from data storage.
- Use a codebook to describe your cases and variables in detail
- Keep your data **tidy**

Chapter 13

R Programming: Parts of Speech

13.1 Command chains

Your commands will be written as chains.

- Each link in the chain will be a data verb and its arguments.
  - The very first link is usually a data frame.
- Links are connected by the chaining symbol `%>%`
- Often, but not always, you will save the output of the chain in a named object.
  - This is done with the *assignment operator*, `<-`

```
Name_of_result <-
  Starting_data_frame  %>%
  first verb (arguments for details) %>%
  next verb (and its arguments) %>%
   ... and so on, up through ...
  last verb (and its arguments)
```

13.2 An example command chain

```
Princes <-
  BabyNames %>%
  filter(grepl("Prince", name)) %>%
  group_by(year) %>%
  summarise(total = sum(count))
```

- A good idea to put each link on its own line
- Note that `%>%` is at the end of each line.
  - Except ... `Princes <-` is assignment
  - Except ... The last line has no `%>%`.

13.3 Syntax and semantics

There are two distinct aspects involved in reading or writing a command chain.

1. Syntax: the grammar of the command
2. Semantics: the meaning of the command

The focus today is on syntax.

13.4 Part of Speech

From the dictionarty

> part of speech noun
>
> parts of speech a category to which a word is assigned in accordance with its syntactic functions. In English the main parts of speech are noun, pronoun, adjective, determiner, verb, adverb, preposition, conjunction, and interjection.

13.5 Parts of Speech in R

1. Data frames
2. Functions
3. Arguments
4. Variables
5. Constants
6. Assignment
7. Formulas (we won't use these until the end)

13.6 Data frames

- A data frame comprises one or more variables.
- Naming convention: data frames are given names that start with a CAPITAL LETTER, e.g., `RegisteredVoters`.
- A data frame will always be the input at the start of a command chain.
- If assignment is used to save the result, the object created is usually a data frame.

13.7 Functions

- Functions are objects that transform an input into an output.
- Functions are always followed by parentheses, that is, an opening `(` and, eventually, a closing `)`.
- Each link in a command chain starts with a function.
  - More specifically, the function is a *data verb* that takes a data frame as input and produces another data frame as output.
  - There are other kinds of functions, e.g. summary (or reduction) functions and transformation functions.

13.8 Arguments

The things that go *inside* a function's parentheses are called *arguments*.

- Arguments describe the details of what a function is to do.
- If there are multiple arguments, they are always separated by commas.

- Many functions take *named arguments* which look like a name followed by an `=` sign, e.g.

```
summarise(total = sum(count))
```

You can also consider the data frame passed along by `%>%` as an argument to the following function.

13.9 Variables

Variables are the components of data frames.

- When they are used, they *always* appear in function arguments, that is, between the function's parentheses.
- A good convention is for variables to have names that start with a lower-case letter. The convention is *not* universally followed.
- Variables will *never* be followed by `(`.

13.10 Constants

Constants are single values, most commonly a number or a character string.

- Character strings will always be in quotation marks,
  `"like this."`

- Numerals are the written form of numbers, for instance.
  `-42 1984 3.14159`

13.11 Discussion Problem

Consider this command chain:

```
Princes <-
  BabyNames %>%
  filter(grepl("Prince", name)) %>%
  group_by(year) %>%
  summarise(total = sum(count))
```

Just from the syntax, you should be able to tell which of the five different kinds of object each of these things is: `Princes`, `BabyNames`, `filter`, `grepl`, `"Prince"`, `name`, `group_by`, `year`, `summarise`, `total`, `sum`, `count`.

Explain your reasoning.

Chapter 14

(PART) Data Summaries and Graphics

Chapter 15

Data vs Information

What's the difference. Although they are often used synonomously, in this course ...

> Data is given; information is taken.

15.1 The word "data":

Dictionary etymology: 1640s, plural of datum, from Latin *datum* "(thing) given," neuter past participle of *dare* "to give".

Historically: Data (Greek: Δ , Dedomena) is a work by Euclid. It deals with the nature and implications of "given" information in geometrical problems. The subject matter is closely related to the first four books of Euclid's Elements. source

15.2

1838 English edition of Euclid

15.3 The word "information"

The English word was apparently derived from the Latin stem (information-) of the nominative (informatio): this noun is derived from the verb informare (to inform) in the sense of "to give form to the mind", "to discipline", "instruct", "teach". Source:

A Dictionary Definition: knowledge acquired through experience or study source

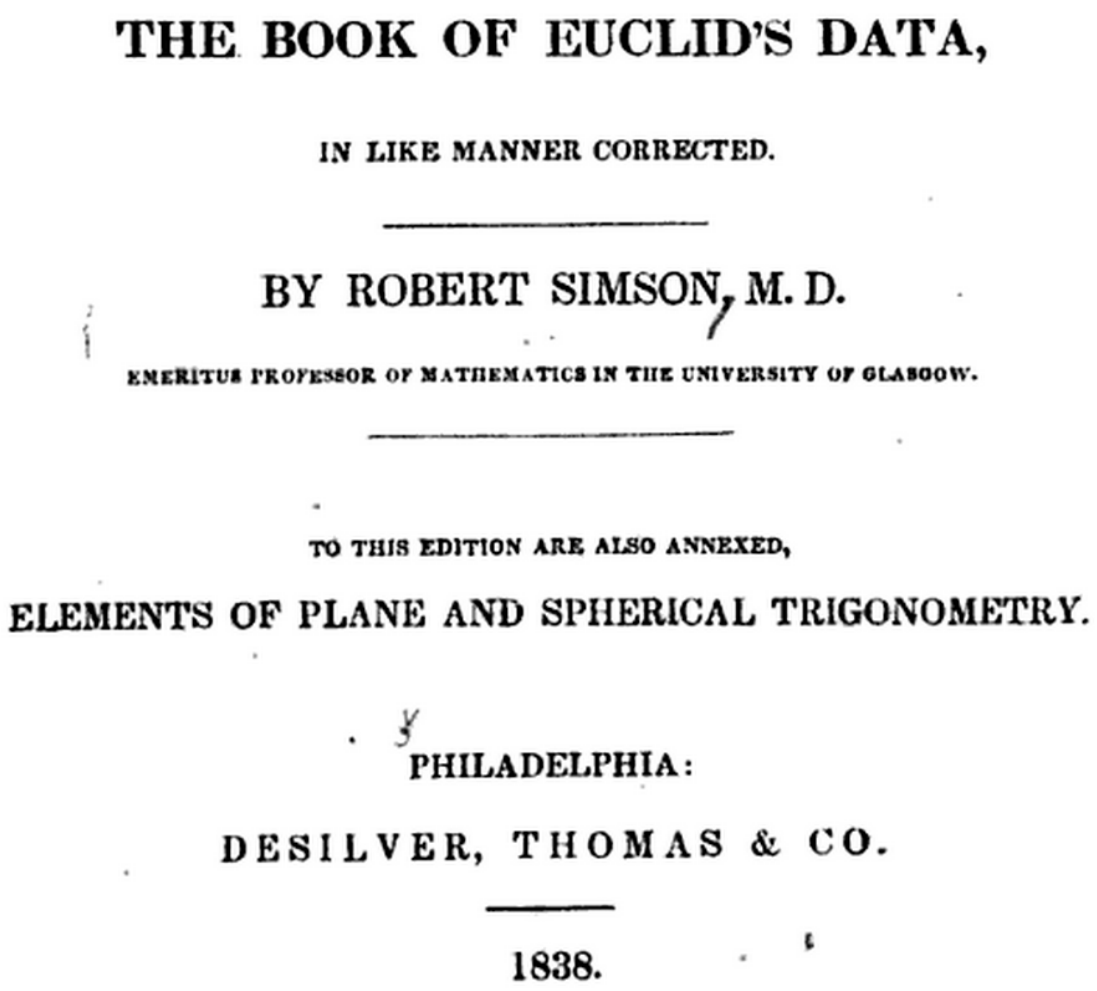
15.4 In this course

Data are measurements, observations, records, etc. of the sort that appear in a data table.

Information is the knowledge or belief that guides decisions or provides explanations.

15.4.1 Our Task

Turn data into information.



THE BOOK OF EUCLID'S DATA,

IN LIKE MANNER CORRECTED.

BY ROBERT SIMSON, M. D.

EMERITUS PROFESSOR OF MATHEMATICS IN THE UNIVERSITY OF GLASGOW.

TO THIS EDITION ARE ALSO ANNEXED,

ELEMENTS OF PLANE AND SPHERICAL TRIGONOMETRY.

PHILADELPHIA:

DESILVER, THOMAS & CO.

1838.

Figure 15.1:

15.5 Data Reports

Knowledge and belief are attributes of the human mind.

- Knowledge is belief that there is consensus about.

We derive knowledge and belief from our experiences including the statements of those we deem to have authority.

A **data report** is a presentation of data from which we are inclined to draw conclusions. Some forms:

- A data table itself. It must be small to be assimilated by us.
- A data graphic. Good data graphics are those that display patterns (or the lack thereof) in a form that is readily assimulated and faithful to the data being reported.
- A data model, such as a statistical model written in a mathematical formalism.

Data graphics can be interpreted without (much) specialized training. Models typically require some specialized skills interpretation.

15.6 Glyph-ready Data

Making data graphics can be straightforward so long as the data underlying the graphic are in the appropriate form.

- We'll call such a form **glyph-ready**.

Data Wrangling is the process of transforming or reshaping the data tables that arrive at our door into a glyph-ready form.

- The glyph-ready form depends on the kind of graphic you wish to make.
- Data reports in general also use data in glyph-ready form, so wrangling techniques useful for graphics are also useful for data modeling or the construction of short, human-interpretable tables.

Chapter 16

Glyphs, Frames, and Scales

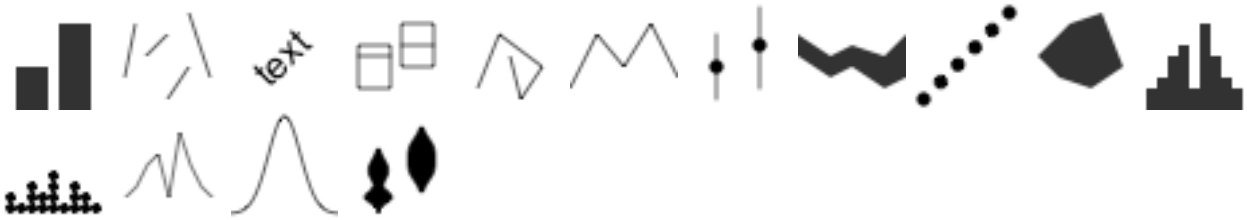
16.1 Glyphs and Data

In its original sense, in archeology, a glyph is a carved symbol.

Heiro**glyph**	Mayan **glyph**
	

16.2 Data Glyph

16.2.1 A data glyph is also a mark, e.g.



The features of a data glyph encodes the value of variables.

- Some are very simple, e.g. a dot: 
- Some combine different elements, e.g. a pointrange: 
- Some are complicated, e.g. a dotplot: 

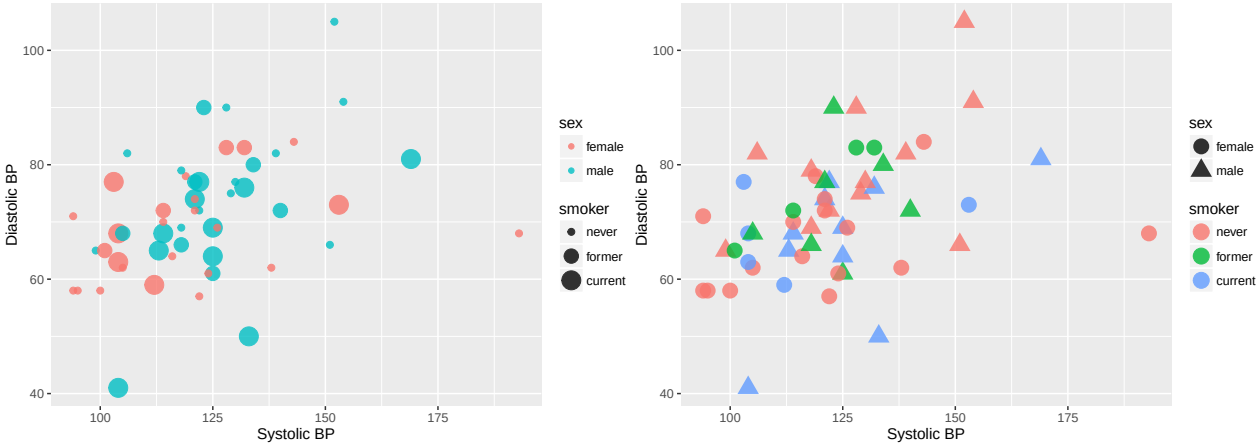
See: *http://docs.ggplot2.org/current/*

16.3 Data Glyph Properties: Aesthetics

Aesthetics are **visual properties** of a glyph.

- Aesthetics for points: location (x and y), shape, color, size, transparency

```
## Warning: Using size for a discrete variable is not advised.
```



- Each glyph has its own set of aesthetics.

16.4 Why "Aesthetic"?

Syllabification: aes·thet·ic
Pronunciation: /es'THedik/

(also **esthetic**)

Definition of *aesthetic* in English:
adjective

1 Concerned with beauty or the appreciation of beauty:
'the pictures give great aesthetic pleasure'

Origin

Late 18th century (in the sense 'relating to perception by the senses'): from Greek *aisthētikos*, from *aisthēta* 'perceptible things', from *aisthesthai* 'perceive'. The sense 'concerned with beauty' was coined in German in the mid 18th century and adopted into English in the early 19th century, but its use was controversial until late in the century.

16.5 Some Graphics Components

glyph The basic graphical unit that represents one case. Other terms used include *mark* and *symbol*.
aesthetic a visual property of a glyph such as position, size, shape, color, etc.

- may be **mapped** based on data values: `sex -> color`
- may be **set** to particular non-data related values: `color is black`

scale A mapping that translates data values into aesthetics.

- example: male -> blue; female -> pink

frame The position scale describing how data are mapped to x and y
guide An indication for the human viewer of the scale. This allows the viewer to translate aesthetics back into data values.

- Examples: x- and y-axes, various sorts of legends

16.6 Scales

The relationship between the variable value and the value of the aesthetic the variable is mapped to.

- Systolic Blood Pressure (SBP) has units of mmHg (millimeters of mercury)
- Position on the x-axis measured in distance, e.g. inches.

The conversion from SBP to position is a *scale*.

- Smoker is "never", "former", "current"
- Color is red, green, blue, ...

The conversion from Smoker to color is a *scale*.

16.7 Guides

Guide: an indication to a human viewer of what the scale is. Example:

- Axis ticks and numbers
- Legends

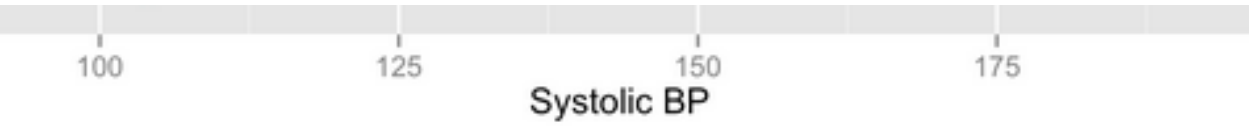
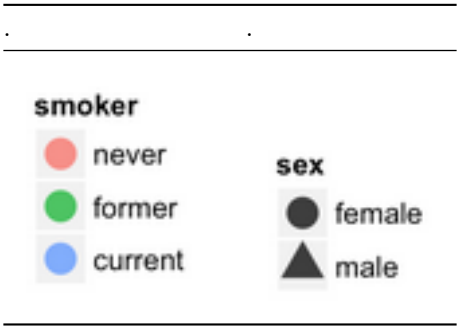


Figure 16.1:



- Labels on faceted graphics

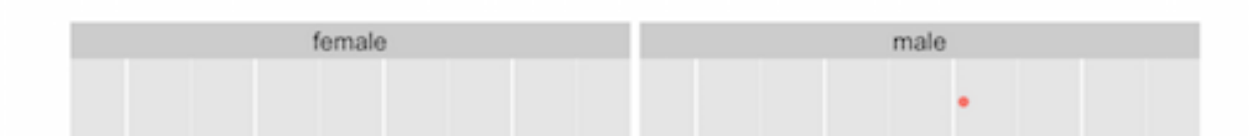
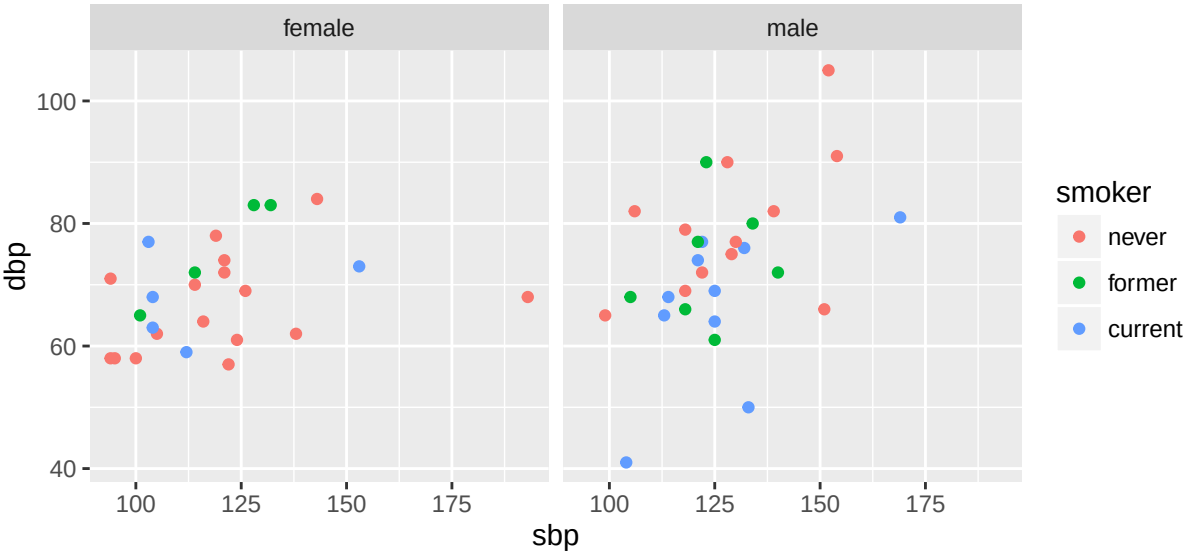


Figure 16.2:

16.8 Facets – using x and y twice



- x is determined by `sbp` and `sex`
- basically a separate frame for each `sex`

16.9 Designing Graphics

Graphics are designed by the human expert (you!) in order to reveal information that's latent in the data.

16.9.0.1 Design choices

- What variables constitute the frame. And some details:
  - axis limits
  - logarithmic axes, etc.
- What variables should be mapped to other aesthetics of the glyph.
- Whether to facet and with what variable.

More details, ..., e.g. setting of aesthetics to constants

16.10 Good and Bad Graphics

Remember ...

> Graphics are designed by the human expert (you!) in order to reveal information that's latent in the data.

Your choices depend on what information you want to reveal and convey.

Learn by reading graphics and determining which ways of arranging thing are better or worse.

A basic principle is that a graphic is about *comparison*. Good graphics make it easy for people to perceive things that are similar and things that are different. Good graphics put the things to be compared "side-by-side", that is, in perceptual proximity to one another.

16.11 Perception and Comparison

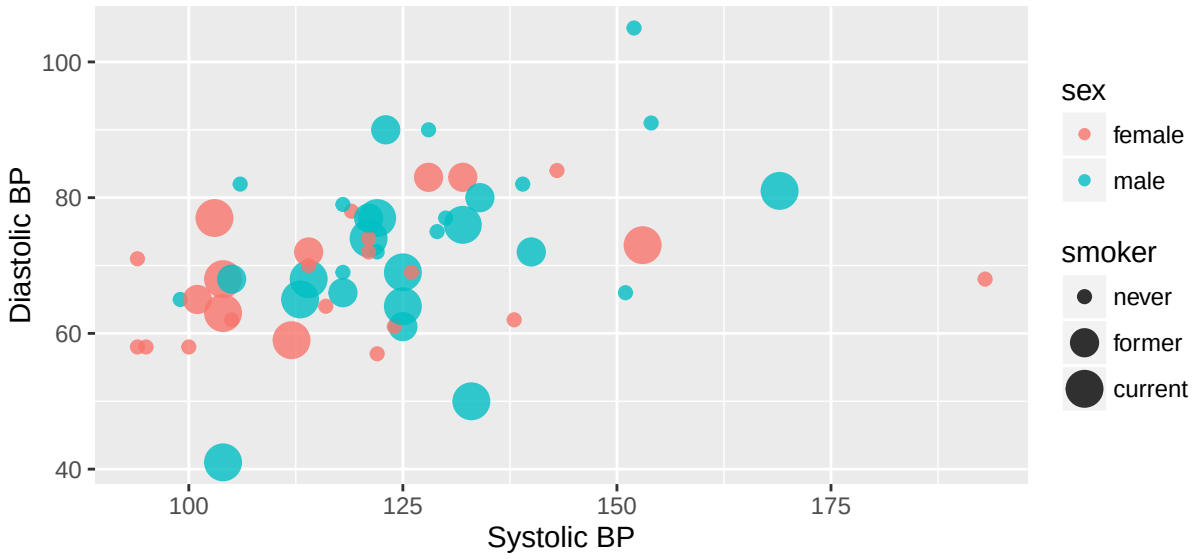
In roughly descending order of human ability to compare nearby objects:

1. Position
2. Length
3. Area
4. Angle
5. Shape (but only a very few different shapes)
6. Color

Color is the most difficult, because it is a 3-dimensional quantity.
- color gradients — we're good at - discrete colors — must be carefully selected.

16.12 Count the ways this graphic is bad

```
## Warning: Using size for a discrete variable is not advised.
```



16.13 Glyph-Ready Data

Glyph-ready data has this form:

- There is one row for each glyph to be drawn.
- The variables in that row are mapped to aesthetics of the glyph (including position)

Glyph-ready data

sbp	dbp	sex	smoker
129	75	male	never
105	62	female	never
122	72	male	never
128	83	female	former
123	90	male	former
122	77	male	current

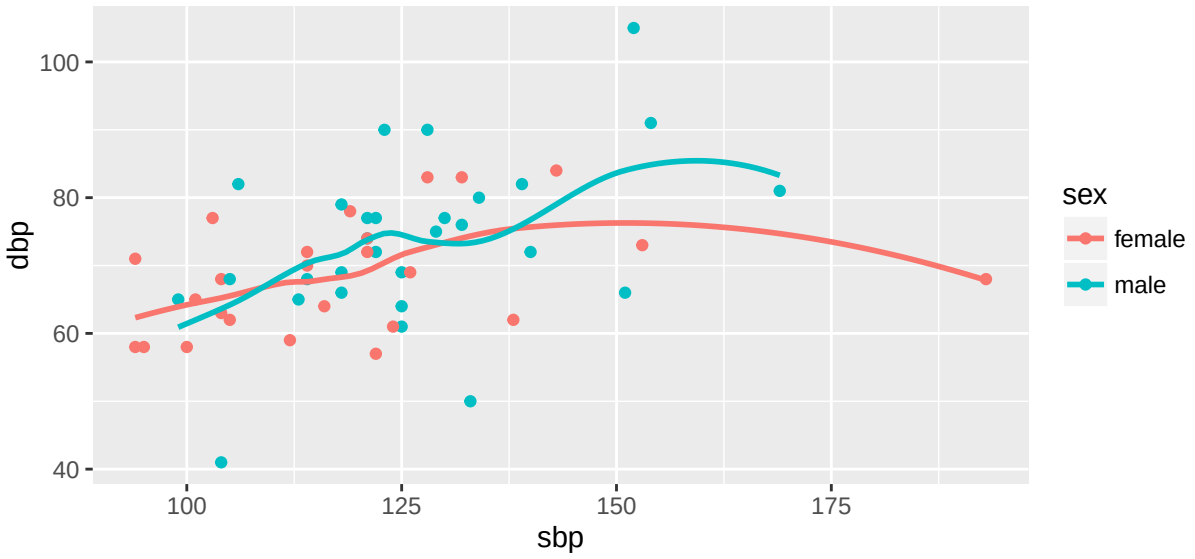
Mapping of data to aesthetics

```
   sbp -> x
   dbp -> y
smoker -> color
   sex -> shape
```

Scales determine details of
`data -> aesthetic` translation

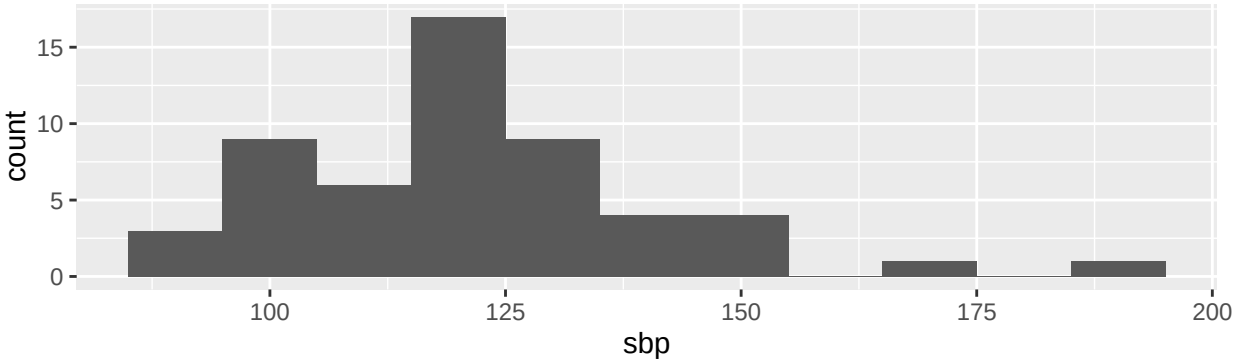
16.14 Layers – building up complex plots

Each layer may have its own data, glyphs, aesthetic mapping, etc.



- one layer has points
- another layer has the curves

16.15 Stats: Data Transformations



- What are the glyphs, aesthetics, etc. for this plot?
- How is the data for this plot related to the "raw" data?

sbp	dbp	sex	smoker
129	75	male	never
105	62	female	never
122	72	male	never
128	83	female	former

16.16 What's Next

1. **Eye-training**
   - recognize and describe glyphs, aesthetics, scales, etc.
   - identify data required for a plot
2. **Data wrangling**
   - get data into glyph-ready format (`dplyr`, `tidyr`)

3. **Graphics construction**
    - start with: map variables to aesthetics interactively with `scatterGraphHelper()`, `barGraphHelper()`, `densityGraphHelper()`
    - move on to: describe data, glyphs, aesthetics, etc. to R using `ggplot2`

Chapter 17

Activity: Mapping the stars

The image above is a map of the stars constructed by the European Space Agency's Gaia space telescope. It reportedly shows 1,000,000,000 stars.[1]

1. Although the map represents one billion stars, the image itself is only 660 by 398 pixels: a total of 262,680 pixels altogether. How can a billion stars be displayed in only one-quarter of a million pixels?
2. Why is the image oval?
3. Why are there broad, curving bands of shading? How might these reflect layers of the graphic that display different quantities? (See the codebook for some ideas about variables that might reflect the available data rather than the stars themselves.)

Gaia data are available in CSV form at this site. A codebook is here

Download one of the CSV files and see what you can make of it. For instance, …

4. You can see the `.csv` in the name. What does the `.gz` mean at the end of the file name?
5. How many stars are there in this one file? From the number of such `.csv.gz` files available, estimate how many stars there are in the complete catalog.
6. Make a map of the stars in your one file. (Suggestion: in developing your plot, just use several thousand stars from the file. Otherwise things will be slow. Select the stars at random.)
   - Use `phot_g_mean_flux` as the intensity and `ecl_lon` and `ecl_lat` as the position variables.
   - Explore a bit and decide what are good aesthetics for representing the intensity. (Hints: color? size?)
   - Does faceting make sense?

[1] See this story on the BBC web site.

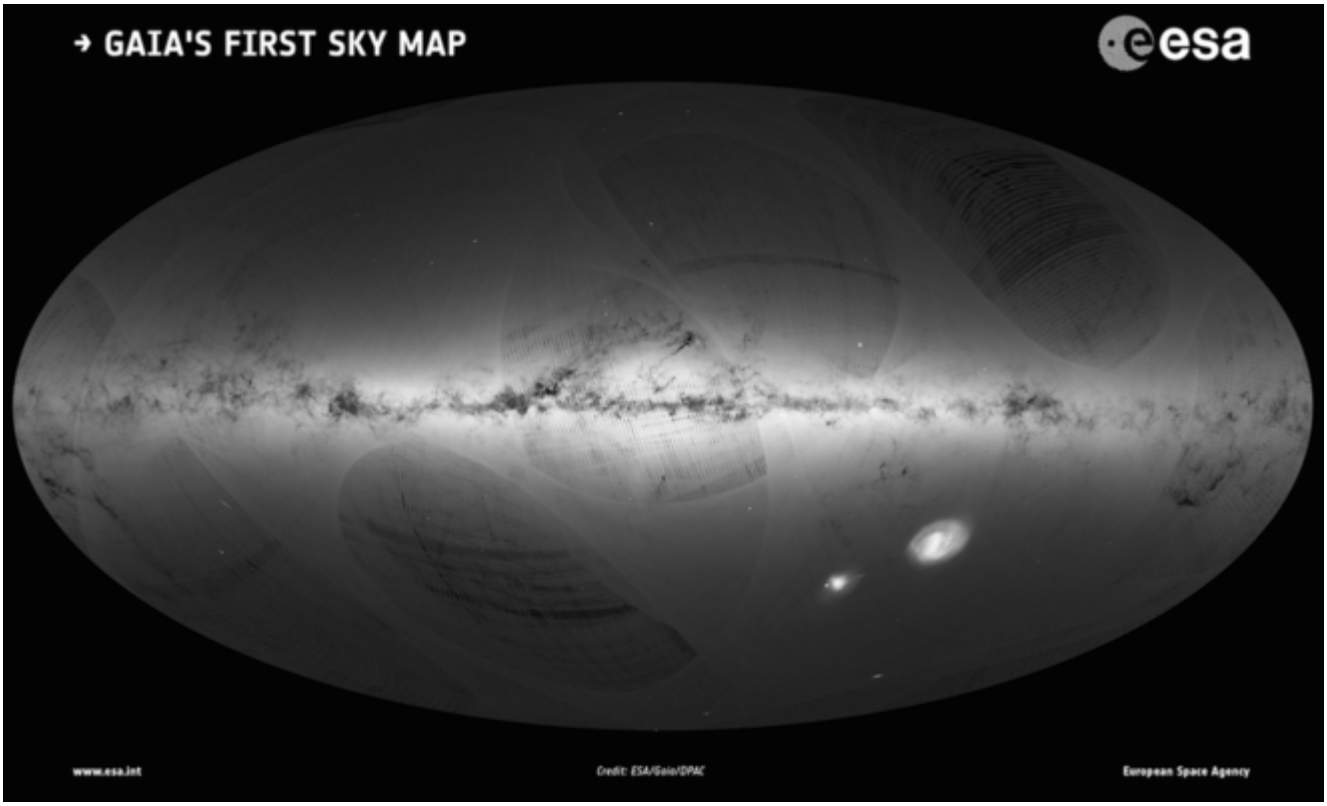
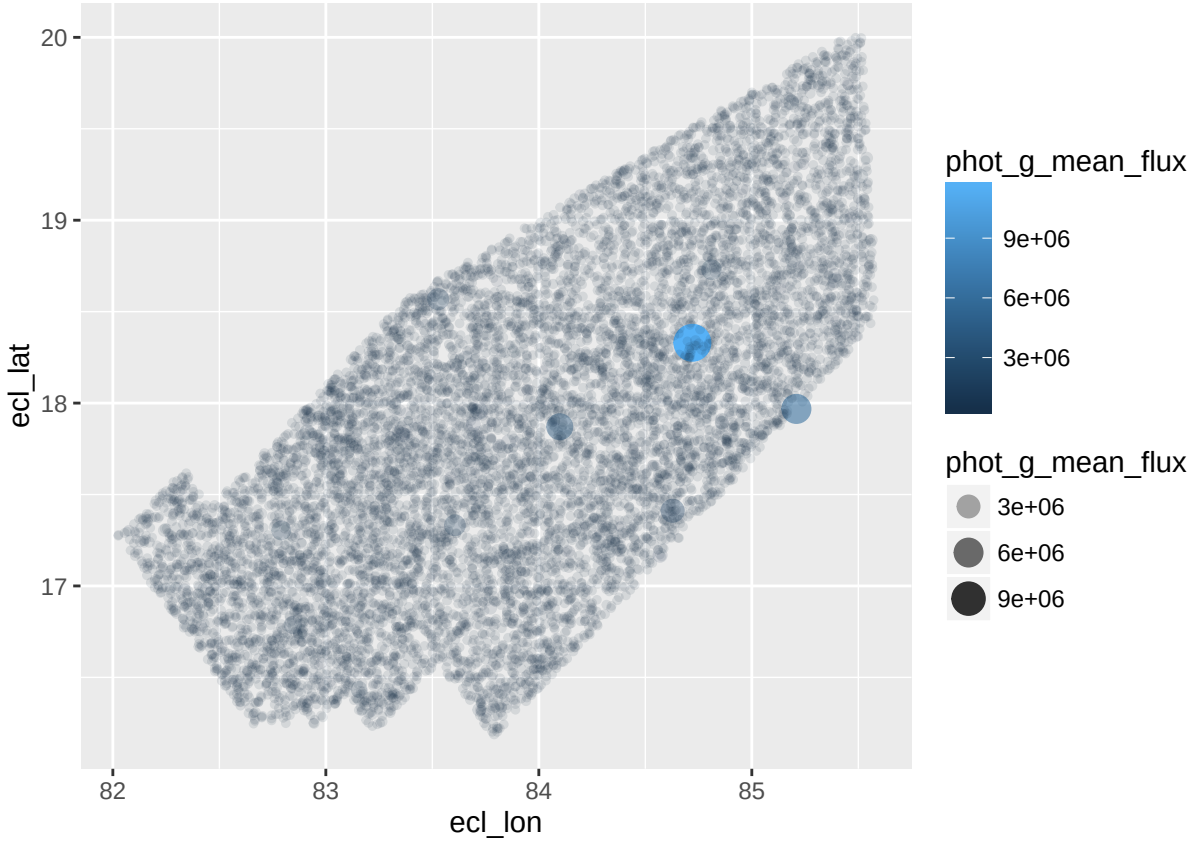
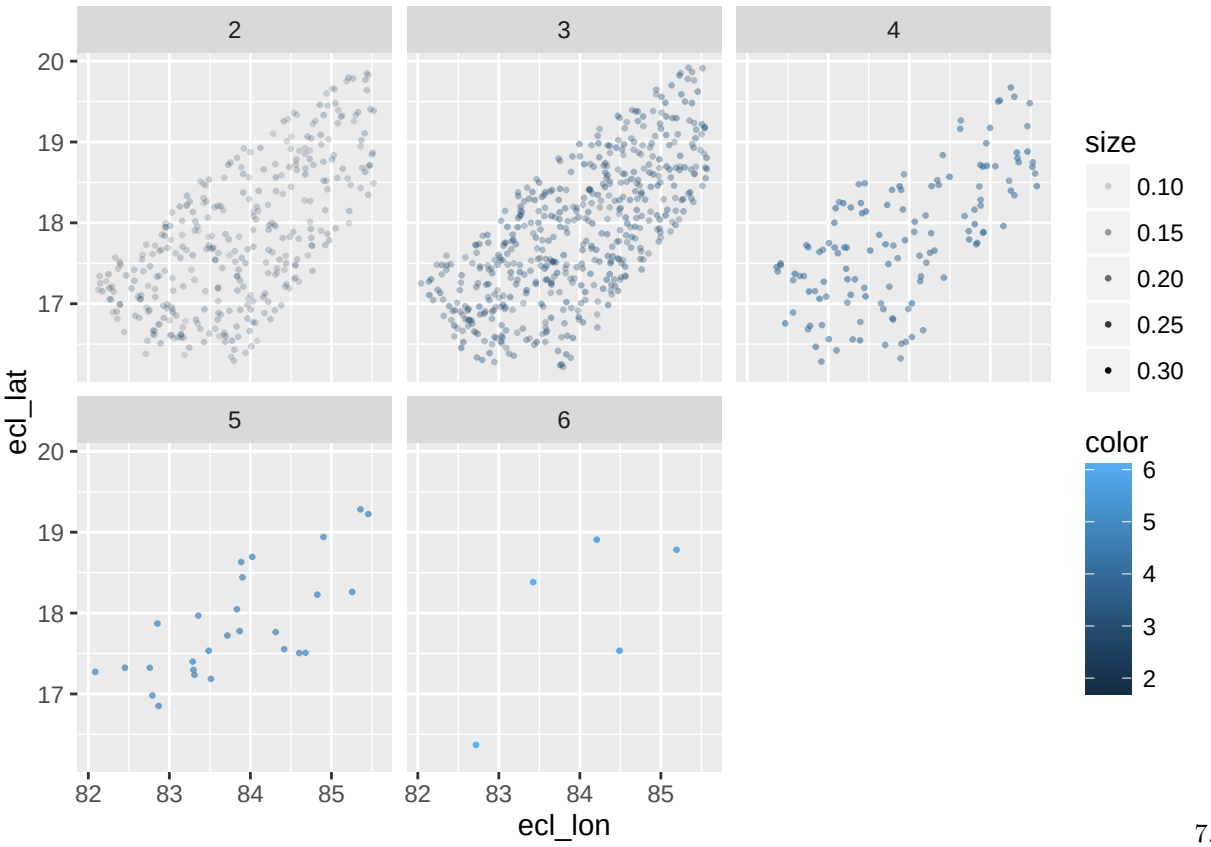


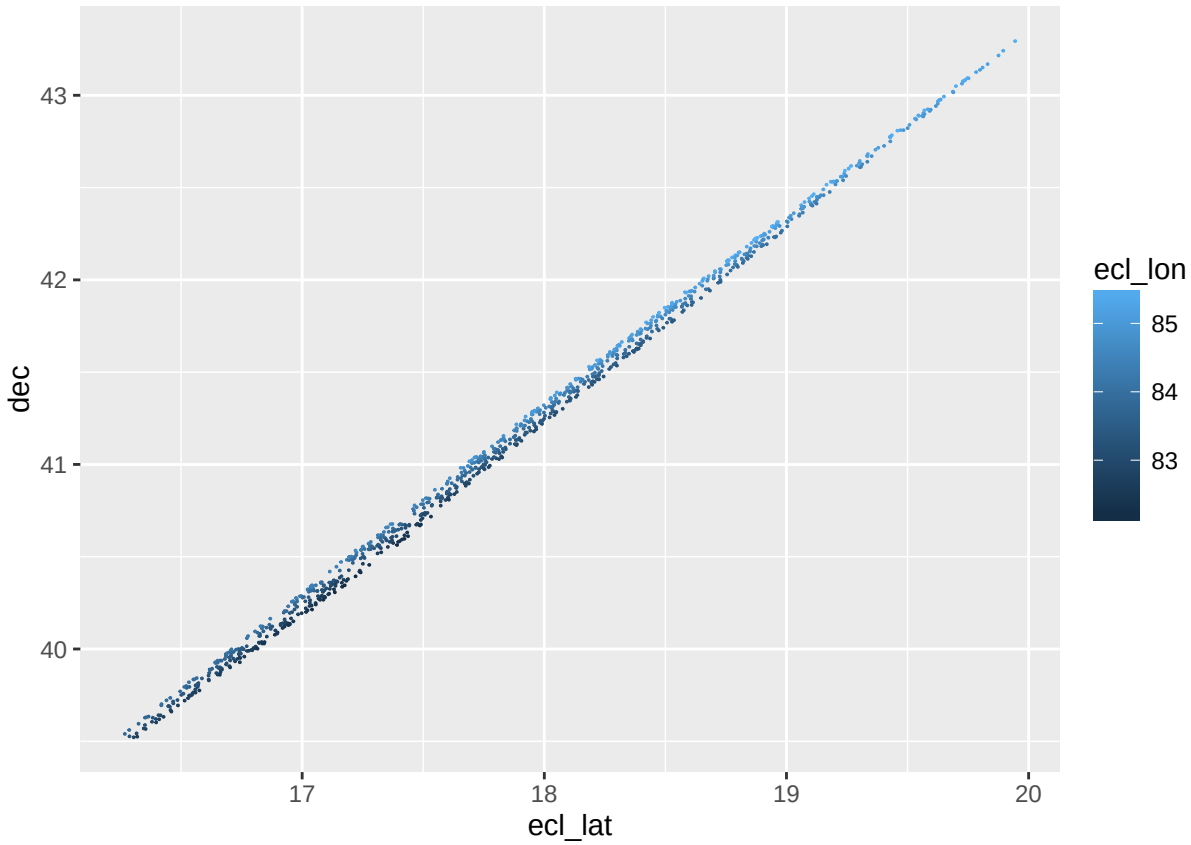
Figure 17.1: Stars plotted on the celestial sphere by the Gaia space telescope. (Sept. 2016)

A simple plot:





7. Is there a relationship between the `ra` and `dec` variables and the `ecl_lon` and `ecl_lat` variables? Try different ways assigning the variables to aesthetics until you find one that tells the story.



8. Optional: Requires some mathematical sophistication. Make a conformal-map style presentation of the relationship between the `ra`/`dec` coordinate system and the `ecl_lat`/`ecl_lon` system. Suggestion: Pull out only those stars that fall within a narrow band of the edges of a square in one of the coordinate systems. Then make separate plots of those stars in the two systems, perhaps using color to encode which stars in one plot correspond to stars in the other plot.

Chapter 18

Introduction to Graphics and Wrangling

PDF handout

In today's activity, you are going to deconstruct some graphics and carry out some data wrangling operations.[1]

18.1 Deconstructing graphics

Considering each of the above graphics in turn, figure out:

- What mode of graphic is it? (e.g. density plot, scatter plot, bar plot, …)
- What variables from the respective data tables are involved?
- What role each of those variables plays in the graphic?
- In Figure 2, why is there no data variable being used for the y-axis?

Here is the basic structure of the commands for making the graphics. You can try various combinations of the variables appearing in the graphics and see which graphic you think is the most informative.

[1] Remember to load the `DataComputing` package: `library(DataComputing)`

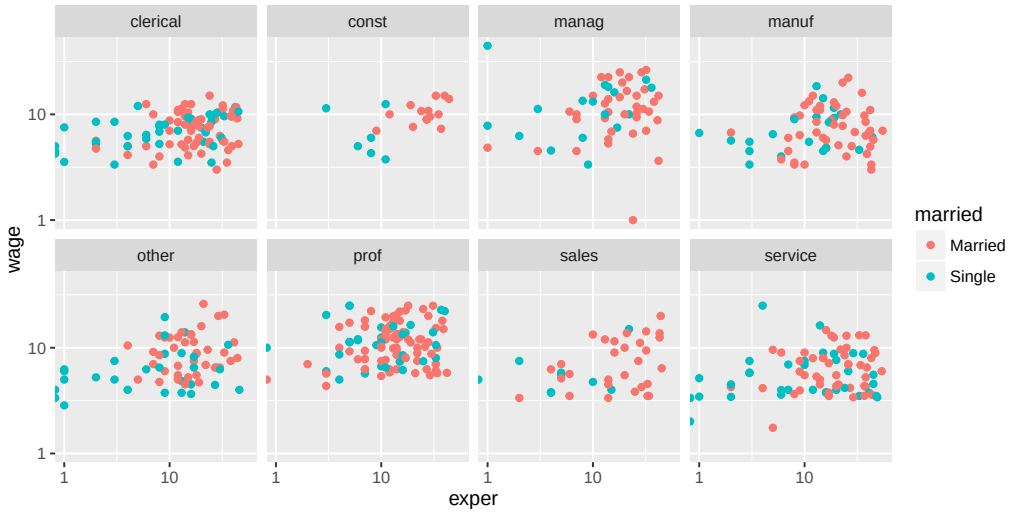


Figure 18.1: A representation of some of the variables from the CPS85 data table in the mosaicData package.

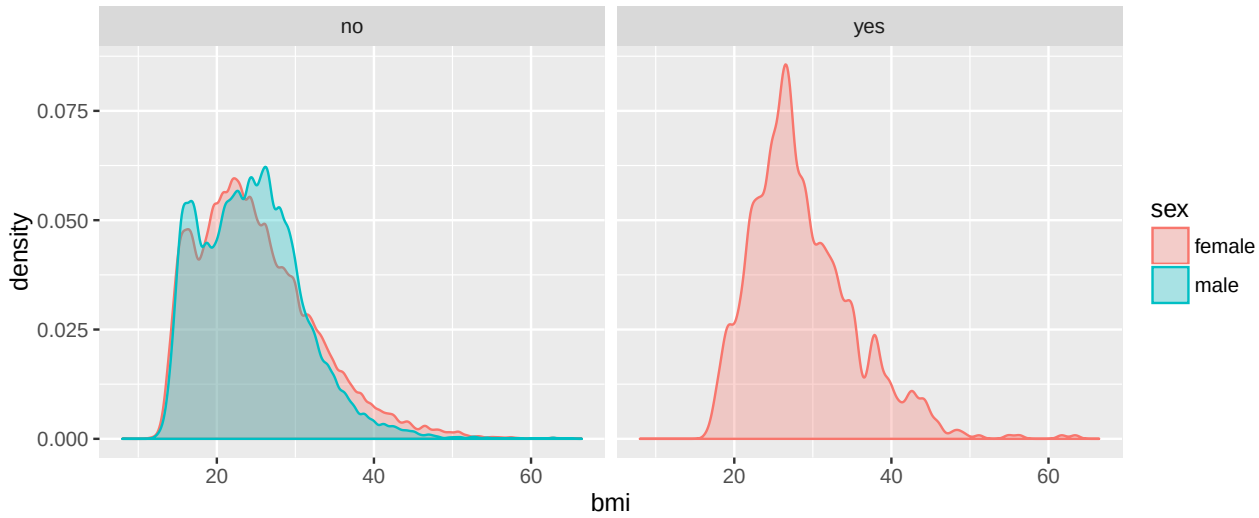


Figure 18.2: Variables from the NCHS data table in the DataComputing package. The `yes' and`no' refers to whether the person is pregnant.

```
ggplot(data = CPS85, aes(x = ????, y = ????, color = ????)) + geom_point() + facet_wrap( ~ ????)
ggplot(data = NCHS, aes(x = ????)) + geom_density(aes(color = ????)) + facet_wrap(~ ???)
```

Put the R statement that generates each graph into your report so that the graphs appear when you compile your `.Rmd` file.

18.2 Wrangling

Diamonds

Refer to the `diamonds` data table in the `ggplot2` package. Take a look at the codebook (using `help()`) so that you'll understand the meaning of the tasks. (Motivated by Garrett Grolemund.)

Each of the following tasks can be accomplished by a statement of the form

For each task, give appropriate R functions or arguments to substitute in place of `verb1`, `verb2`, `args1`, `args2`, and `args3`.

1. Which color diamonds seem to be largest on average (in terms of carats)?
2. Which clarity of diamonds has the largest average "table" per carat?

Voting

Using the `Minneapolis2013` data table, answer these questions. `summarise()`, `group_by()`, and `tally()` will be useful in answering the questions.

1. How many cases are there?
2. Who were the top 5 candidates in the `Second` vote selections.
3. How many ballots are marked "undervote" in
   - `First` choice selections?
   - `Second` choice selections?
   - `Third` choice selections?
4. What are the top 3 `Second` vote selections among people who voted for Betsy Hodges as their first choice?
5. Which `Precinct` had the highest fraction of `First` vote selections marked as "undervote"? (To calculate the fraction, use `mean(First == "undervote")` in the argument to `summarise()`.)

Chapter 19

(PART) Data Verbs

Chapter 20

Basic Data Verbs

20.1 Three kinds of verbs for data wrangling

1. Data verbs
2. Reduction verbs
3. Transformation verbs

20.1.1 Non-wrangling verbs

There will also be verbs for graphics, loading data, etc., but for wrangling we'll need mainly these three types. Examples:

- `library()` – attaches the software distributed in a package to your session of R
- `read.csv()` and other file-reading functions. Creates a data table given the location of a file containing those data.
- `scatterGraphHelper()` – takes a data table as input, but produces graphics as output.
- `data()` – accesses data from a package. `data()` is not a data verb!

20.2 Data Verbs

What distinquishes a data verb from a reduction or transformation verb?

- Data verbs create a new data table, from an input data table.

20.2.1 Some commonly used data verbs.

1. `summarise()`
2. `group_by()`
3. `filter()`
4. `mutate()`
5. `select()`
6. `arrange()`
7. join (there's a family of joins – more on this later)

20.3 Reduction verbs

Characteristics?

Variable in, a single number out.

Examples?

20.4 Transformation verbs

Variable in, variable out.

Examples?

Chapter 21

More transformation verbs

Recall that a *transformation verb* takes a variable as an input and produces a variable of the same length as the output.

Some familiar transformation verbs from primary and secondary school:

- arithmetic operations (`+`, `-`, `/`, `*`)
- mathematical functions such as logs and exponentiation

There are additional transformations which will be unfamiliar, simply because they did not fit into the algebra- and trigonometry-based high-school curriculum

1. Rank transforms
2. Lead and lag transforms
3. Date transforms
4. Conditional transforms
5. Character transforms

21.1 Rank transforms

Many questions take forms such as these:

- "Find the largest …"
- "Find the three largest …"
- "Find the smallest within each group …"

The functions `min()` and `max()` are **reduction** verbs. They tell you the single lowest or highest value in a set. Because they are reduction verbs, they are often used in `summarise()`, which reduces a set of cases to a single case.

sex	most_popular
F	99674
M	94758

Notice that `name` was not carried along. When you `summarise()`, the only variables that appear in the output are the grouping variables, and the variables you create through the arguments to `summarise()`.

If you wanted to know the *names* that are most popular, you will need to rely on `filter()`. Get rid of the cases that are not the most popular.

name	sex	count	year
Linda	F	99674	1947
James	M	94758	1947

`Filter()` needs a criterion. The criterion `count == max( count )` (with the double equals sign `==`) passes through the case where the value of `count` matches the largest value of `count`. That will be the biggest case.

name	sex	count	year
James	M	87428	1946
Linda	F	99674	1947
James	M	94758	1947
Robert	M	91652	1947
John	M	88318	1947
Linda	F	96210	1948
James	M	88610	1948
Robert	M	85492	1948
Linda	F	90994	1949
James	M	86779	1949
James	M	86204	1950
James	M	87194	1951
Robert	M	86323	1951
James	M	87029	1952
Robert	M	86548	1952
Robert	M	86102	1953
James	M	85934	1953
Michael	M	88481	1954
James	M	86273	1954
Robert	M	86251	1954
Michael	M	88281	1955
David	M	86179	1955
Michael	M	90629	1956
Michael	M	92711	1957
Michael	M	90512	1958
David	M	85932	1960
Michael	M	86914	1961
Michael	M	85316	1970

Note: Almost all of these are from the late 1940s and early 1950s. In part, this reflects the baby boom. Perhaps it also reflects the conformity that people associate with that era.

QUESTION: How would you estimate "conformity" for each year?

Possibility: Take the 10 most popular names each year. Find out what fraction of the total number of births that was.

The `rank()` operation is helpful here.

The `rank()` function does something simple but powerful: it replaces each number in a set with where that number stands with respect to the others. For instance, look at the tiny data table `Set` shown in Table **??**. What's the rank of the number 5 in the `numbers` variable.

numbers	the_rank
2	1.5
5	4.0
4	3.0
7	5.0
2	1.5
9	7.5
9	7.5
8	6.0

Or, seen another way

numbers	the_rank
2	1.5
2	1.5
4	3.0
5	4.0
7	5.0
8	6.0
9	7.5
9	7.5

Notice how ties are broken. Also note that the biggest numbers have the *highest* ranks. This is different than the convention in everyday language, where the "Number 1 ranked team" is the best team. To follow this convention, use `rank(desc(numbers))`.

numbers	the_rank
2	7.5
2	7.5
4	6.0
5	5.0
7	4.0
8	3.0
9	1.5
9	1.5

Suppose you want to find the 3rd most popular name of all time. Use `rank()`.

name	total
Robert	4809858

Or, to find the top three most popular names, replace `==` in the above by `<=`.

name	total
James	5114325
John	5095590
Robert	4809858

When applied to grouped data, `rank()` will be calculated separately within each group. That is, the rank of a value will be with respect to the other cases in that group. For instance, here's the third most popular name for each sex.

name	sex	count	year
Linda	F	99674	1947
James	M	94758	1947
Robert	M	91652	1947
Linda	F	96210	1948
Linda	F	90994	1949
Michael	M	92711	1957

21.1.1 Tied ranks

Sometimes, two or more numbers are tied in rank. The `rank()` function deals with these by assigning all the tied values the same rank, which is the mean of the ranks those values would have had if they were even slightly different. There are other rank-like transformation verbs that handle ties differently. For instance, `row_number()` breaks ties in favor of the first case encountered.

numbers	the_rank	ties_broken
2	1.5	1
2	1.5	2
4	3.0	3
5	4.0	4
7	5.0	5
8	6.0	6
9	7.5	7
9	7.5	8

21.2 Leads and lags

numbers	next_one
2	NA
5	2
4	5
7	4
2	7
9	2
9	9
8	9

Find the names that increase the most from one year to the next. Let's take "increase the most" to mean "the biggest proportional increase", but consider only names that have more than 100 kids in the earlier year.

Find proportional increase for each name for each year, but push this to zero if the base year had less than 100 kids.

name	sex	count	year
Hillary	F	5	1922
Hillary	F	7	1942
Hillary	F	13	1943
Hillary	F	19	1944
Hillary	F	19	1945
Hillary	F	29	1946
Hillary	F	51	1947
Hillary	F	43	1948
Hillary	F	48	1949
Hillary	F	54	1950
Hillary	F	40	1951
Hillary	F	43	1952
Hillary	F	57	1953
Hillary	F	78	1954
Hillary	F	71	1955
Hillary	F	100	1956
Hillary	F	87	1957
Hillary	F	94	1958
Hillary	F	83	1959
Hillary	F	75	1960
Hillary	F	111	1961
Hillary	F	112	1962
Hillary	F	148	1963
Hillary	F	132	1964
Hillary	F	111	1965
Hillary	F	123	1966
Hillary	F	155	1967
Hillary	F	178	1968
Hillary	F	198	1969
Hillary	F	258	1970
Hillary	F	251	1971
Hillary	F	242	1972
Hillary	F	250	1973
Hillary	F	313	1974
Hillary	F	323	1975
Hillary	F	331	1976
Hillary	F	442	1977
Hillary	F	718	1978
Hillary	F	921	1979
Hillary	F	835	1980
Hillary	F	736	1981
Hillary	F	752	1982
Hillary	F	696	1983
Hillary	F	1009	1984
Hillary	F	1047	1985
Hillary	F	1082	1986
Hillary	F	1080	1987
Hillary	F	1083	1988
Hillary	F	1285	1989
Hillary	F	1524	1990
Hillary	F	1788	1991
Hillary	F	2522	1992
Hillary	F	1064	1993
Hillary	F	409	1994
Hillary	F	310	1995
Hillary	F	312	1996
Hillary	F	294	1997

Notice "Woodrow" and "Wilson" in 1911 to 1912. Why?

Notice "Samantha" and "Darin" in 1964. Why? (Hint: "Bewitched")

FLAW: There might be years left out for some names. We'll have to wait until we study joins to see how to fix this.

21.3 Times and Dates

With the `lubridate` package:

- Transform text dates into an R type with numerical properties.
  - `ymd()`, `dmy()`, and so on.
- Extract parts of the date:
  - `day()`
  - `jday()`
  - `week()`
  - `hour()`
  - `wday()`
  - `minute()`
  - `month()`
  - `year()`

```
## [1] "character"
```

```
## [1] "Date"
```

middle
1999-11-26

```
## [1] "2016-10-13 08:57:18 CDT"
```

21.4 Conditional transforms

For all transform verbs, the output depends on some way on the input. In *conditional transforms*, you use the input to choose one or more possible outputs.

One such transform verb is `ifelse()`. This takes three arguments: 1. a test, written in the form of a "logical"" such as `x > 3` 2. the output if the test is `TRUE` 3. the output if the test is `FALSE`

For example, the table below shows measurements made on patients, including the date of the measurement and the date of treatment. A transformation is to be done to say whether the measurement was *before* or *after* the treatment.

```
head(Treatments)
```

subject	what	value	date	treatment_date
BHO	sbp	160	2007-06-19	2012-08-05
GWB	sbp	115	1998-04-21	2005-11-14
BHO	sbp	155	2005-11-08	2012-08-05
WJC	sbp	145	2002-11-15	1998-09-30
WJC	sbp	NA	2010-03-26	1998-09-30
WJC	sbp	130	2013-09-15	1998-09-30

The calculation of *before* or *after* is simple: `date < treatment_date`." To render the result as the values `"before"` and `"after"`, use `ifelse()`:

subject	what	value	date	treatment_date	when
BHO	sbp	160	2007-06-19	2012-08-05	before
GWB	sbp	115	1998-04-21	2005-11-14	before
BHO	sbp	155	2005-11-08	2012-08-05	before
WJC	sbp	145	2002-11-15	1998-09-30	after
WJC	sbp	NA	2010-03-26	1998-09-30	after
WJC	sbp	130	2013-09-15	1998-09-30	after

21.5 Text

Simple operators:

- `tolower()` - convert to lower case
- `toupper()` - convert to upper case
- `nchar()` - the number of characters in a string
- `substr()` - extract a substring at a particular location.

Example:

name	sex	count	year	small	large	length	middle
Christina	M	22	1967	christina	CHRISTINA	9	hr
Rotha	F	7	1907	rotha	ROTHA	5	ot
Wayman	M	9	1997	wayman	WAYMAN	6	ay
Song	F	11	1994	song	SONG	4	on
Julian	M	535	1948	julian	JULIAN	6	ul

For later:

- `gsub()`
- `grepl()`
- `DataComputing::extractMatches()`

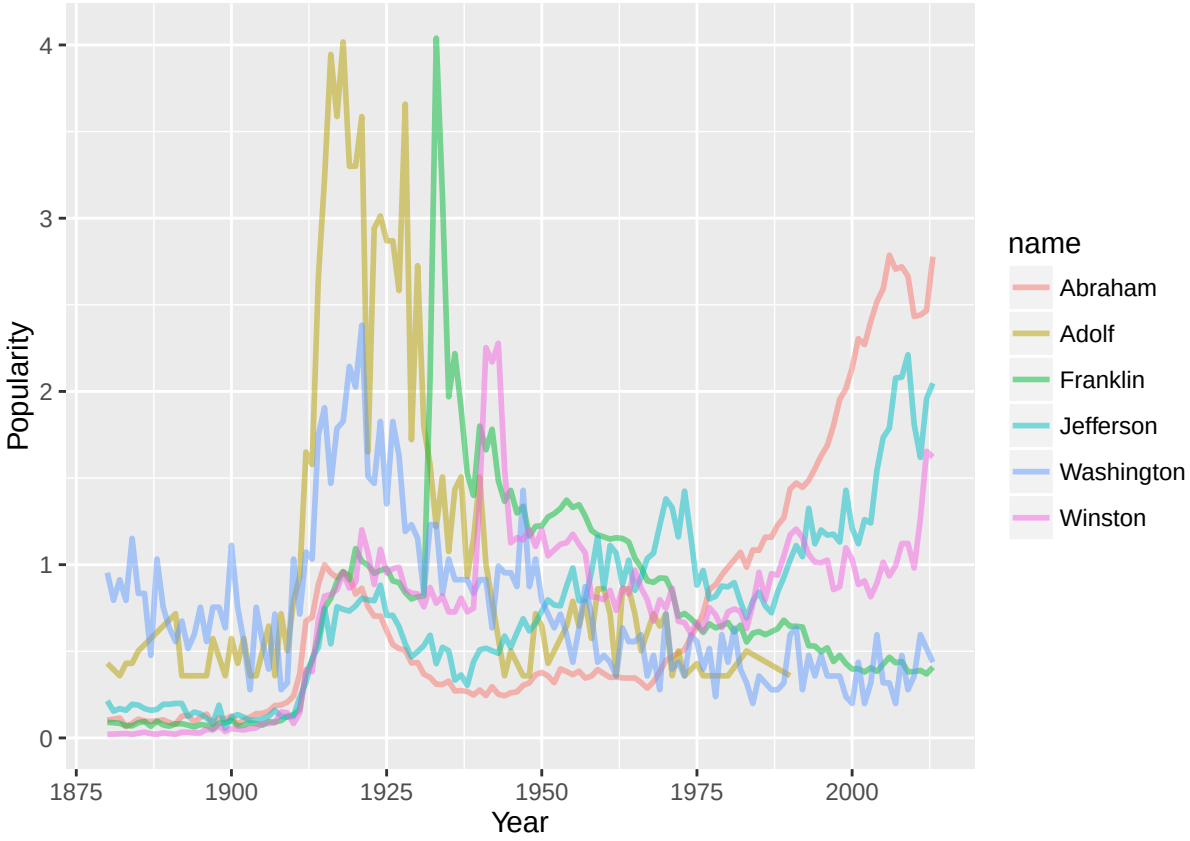
Chapter 22

Trends in Popularity of Names

The relative popularity of different names for babies varies over the years and decades. Let's construct a visualization of how the popularity of names varies in time.

22.1 Objective

Create a graph like the following using name of interest to you.



The raw material you have is the `BabyNames` data set in the `DataComputing` package.

Point out which variables are categorical. These can potentially be used for defining groups of cases.

22.1.1 First, Individually ...

22.1.1.1 Step 1.

Analyze the graphic to figure out what a glyph-ready data table should look like. Mostly, this involves figuring out what variables are represented in the graph. Write down a small example of a glyph-ready data frame that you think could be used to make something in the form of the graphic.

- What variable(s) from the raw data table do not appear at all in the graph?
- What variable(s) in the graph are similar to corresponding variables in the raw data table, but might have been transformed in some way.

22.1.1.2 Step 2

Consider how the cases differ between the raw input and the glyph-ready table.

- Have cases been **filtered** out?
- Have cases been grouped and **aggregated/summarized** within groups in any way?
- Have any new variables been introduced?

22.1.1.3 Step 3

Using English, write down a sequence of steps that will accomplish the transfiguration from the raw data table to your hypothesized glyph-ready data table.

22.1.1.4 Step 4: Confer with your colleagues

As a group, compare your different analyses in Steps 1 through 3. Your goal is to develop a consensus for the design in Step 3.

22.1.1.5 Step 5: Implementation

Now you can start writing the commands themselves. Do so, try to identify and solve any problems that arise, and make your glyph-ready data.

For graphing, you can use this template:

Chapter 23

Case study in basic data verbs: Moby Dick

23.1 Prolog: Scraping and arranging the data

A text file of the book is available at http://www.gutenberg.org/ebooks/2701. At that page is a link to a UTF-8 encoded text document named `"pg2701.txt"`. I downloaded the file and stored it on my machine as `Data/pg2701.txt`. I can read that using `readLines()`.

```
Moby <- readLines("Data/pg2701.txt")
```

You could also read the file directly from Project Gutenberg

```
con <- file("http://www.gutenberg.org/ebooks/2701.txt.utf-8")
Moby <- readLines(con)
close(con)
```

The result, stored in the `Moby` object, is a character vector of 22108 strings. Some of these are prefatory matter, some postscript.

The text itself begins after a line

```
start_text <- "START OF THIS PROJECT GUTENBERG"
```

and ends before a line

```
end_text <- "END OF THIS PROJECT GUTENBERG"
```

Using these as delimiters includes some transcriber's notes, etc. For simplicity, I'll take as the start the line

```
start_text <- "CHAPTER 1\\."
```

The last line is simply "orphan." ending line

```
end_text <- "^orphan\\.$"
```

Why the funny spelling? The `start_text` and `end_text` are being specified as a "regex" (sometimes called *regular expression*) indicating that the word "orphan" is at the very beginning of the line, followed by a period and the end of the line.

Regexes are a way of describing patterns. For our purposes, we'll use them to identify the first and last line of Melville's work in the Project Gutenberg text. As it happens, there are two instances of "CHAPTER 1." in *Moby Dick*. The second is a book within the book. We want to start with the early instance.

```
first_line <- min(grep(start_text, Moby))
last_line <- grep(end_text, Moby)
Moby <- Moby[first_line : last_line]
```

We want to break the strings up into individual words. We'll do this "by hand" because I want to render the text as a simple set of words and punctuation. Steps:

1. Change punctuation so that it is an isolated character.
2. Split up each line by spaces into words.
3. Convert to lower case (because I'm not interested in capitalization).

```
tmp <- Moby
characters <- unlist(strsplit(tolower(Moby), split = NULL))

# Step 1
punctuation <- c(".", ",", ";", ":", "?", "!", '"',
                 "'", "&", "-", "(", ")", "[", "]")
for (symbol in punctuation) {
  result <- paste0(" ", symbol, " ")
  tmp <- gsub(symbol, result, tmp, fixed=TRUE )
}
# Step 2
Words <- unlist(strsplit(tmp, split = " "))
# Step 3
Words <- data.frame(word = tolower(Words),
                    stringsAsFactors = FALSE)
# Get rid of empty strings
# Words <-
#  Words %>%
#  filter(word != "")
```

What are the character frequencies in the book?

```
table(characters) %>%
  data.frame(stringsAsFactors = FALSE) %>%
  arrange(desc(Freq))
```

characters	Freq
	190500
e	115021
t	86551
a	76496
o	68135
n	64555
i	64384
s	63107
h	61779
r	51160
l	42049
d	37658
u	26316
m	22904
c	22143
w	21774
g	20493
f	20476
,	18947
p	16961
y	16602
b	16600
v	8418
k	7937
.	7385
-	5741
;	4143
”	2879
’	2850
!	1741
q	1544
j	1061
x	1006
?	999
z	623
(	201
)	201
:	192
0	123
1	123
2	54
5	52
7	47
8	47
*	45
3	45
4	34
6	31
9	31
_	4
[	2
]	2
&	2
$	2

23.2 Most common words

```
Popular <-
  Words %>%
  group_by(word) %>%
  tally() %>%
  arrange(desc(n)) %>%
  head(50)
```

23.3 Most common sequences

```
Sequences <-
  Words %>%
  filter(grepl("[a-zA-Z]", word)) %>%
  mutate(two = lead(word, 1), three = lead(two, 1),
         four = lead(three, 1))

CommonPairs <-
  Sequences %>%
  group_by(word, two) %>%
  tally() %>%
  ungroup() %>%
  arrange(desc(n))
```

Popular triplets

```
Triplets <-
  Sequences %>%
  group_by(word, two, three) %>%
  tally() %>%
  ungroup() %>%
  arrange(desc(n))
```

Chapter 24

Cities of the World

The data table `WorldCities` (in the `DCF` package) identifies cities around the world that have large populations or are large for their region.

- Check the table for plausibility: is it possibly what it is claimed to be?. For instance … What's the total number of people represented? Explain why or why not the data pass this plausibility test. Create another plausibility test and describe it. It can be very simple. If you can, implement it and state whether the data table passes the test.
- How many cities larger than 100,000? Larger than 1,000,000?
- Make a scatterplot of the latitude and longitude of cities larger than 100K.
  - Decide what variables to map to the x and y aesthetics.[1]
  - Use the size of the dot to show the city's population. In other words, map the variable population to the `size` aesthetic.
  - Use transparency, called `alpha`, to handle overplotting. Alpha can run from zero to one: zero is completely transparent (a.k.a. invisible); one is completely opaque. You will be *setting* `alpha` the same for every city. Recall that in `ggplot` graphics, variables are *mapped* to aesthetics using the `aes()` function. In contrast, aesthetic properties that are the same for every case are *set* outside the `aes()` function. In a typical use, the `ggplot()` command will look like

    ```
    ggplot( data=???, aes( x=???, y=??? ))
    ```

    The layers of the plot will be used like this:

    ```
    geom_point( alpha=???, aes( size=??? ) )
    ```

    where, of course, you will replace the `???` with appropriate variables or constants or data tables.
- When you have your plotting commands complete, use those commands to make another graphic, but add this expression to govern the `size` attribute: `+ scale_size_area()`. This will make the *area* of the dot proportional to the value of the variable mapped to it. Without `scale_size_area()`, the *diameter* of the dot is proportional to the variable. Explain which scale, area or diameter, you think is most informative. (Include both graphics in your Rmd file along with your explanation.)
- Create a data table `BiggestByCountry` that has the one biggest city in each country.
- Plot the locations of `BiggestByCountry` as another layer in your graphic. Make them red.
- Add to the graphic the names of the cities from `BiggestByCountry`. Hint: use `geom_text()`. Set the `size=2`. Remember, *setting* is different from *mapping* a variable. You'll use the `label=` aesthetic to represent the city names.

[1] Remember, "aesthetic" is being used in its original sense: how things are perceived.

- Find the countries where the biggest city is more than 5M people

  The resulting table will have a couple of dozen cases. Display as output in your report all the cases but just these variables: city name, country, and population.

Chapter 25

Additional Exercises on Data Verbs

25.1 How many births

You might think that you can find the number of babies born each year in the US by using the Social Security `BabyNames` data. Just group by year and add up the counts:

But some babies are missing from this tally. In particular, the `BabyNames` data only reports names, years, and sex for which there were five or more births. Presumably there were some babies whose names were so unusual that they are unique, or shared by only two, three, or four babies.

Your job, make an estimate of how many such babies there are. For simplicity, make that estimate for a single year, say 2010.

Here's an approach. Count the total number of -names with 5, 6, … 19, 20 births. That is, consider all the names of each sex with only 5 babies reported and find the total number of babies that fall into that class. The same for names-and-sex with 6 births, and so on. See if you can spot a pattern with how the number of babies changes depends on each count.Then extrapolate out to 1, 2, 3, 4 and use that to estimate the total population.

The pattern is very consistent across the different levels of "numbers of names". It looks like we're missing about 100,000 babies in 2010.

Chapter 26

Bicycle use patterns

PDF handout

In this activity, you'll examine some factors that may influence the use of bicycles in a bike-renting program. The data come from Washington, DC and cover the last quarter of 2014.

We will use the "Trips-history" data file You can access the data like this.

Important: To avoid repeatedly re-reading the files, start the above chunk with `{r cache = TRUE}` rather than the usual `{r}`.

The `Trips` data table is a random subset of 10,000 trips from the full quarterly data. Start with this small data table to develop your analysis commands. When you have this working well, you can access the full data set of more than 600,000 events by removing `-Small` from the name of the data_site.

26.1 Time of day

It's natural to expect that bikes are rented more at some times of day than others. The variable `sdate` gives the time (including the date) that the rental started.

Make these plots and interpret them:

1. A density plot of the events versus `sdate`. Use `ggplot()` and `geom_density()`.
2. A density plot of the events versus time of day. You can use `lubridate::hour()`, and `lubridate::minute()` to extract the hour of the day and minute within the hour from `sdate`, e.g.

```
Trips %>%
  mutate(time_of_day =
       lubridate::hour(sdate) +
         lubridate::minute(sdate) / 60) %>%
  ... further processing ...
```

3. Facet (2) by day of the week. (Use `lubridate::wday()` to generate day of the week. Use `+ facet_wrap( ~ day_of_week)` to perform the faceting.)
4. Set the `fill` aesthetic for `geom_density()` to the `client` variable.[1] You may also want to set the `alpha` for transparency and `color=NA` to suppress the outline of the density function.
5. Same as (4) but using `geom_density()` with the argument `position = position_stack()`.

[1] `client` describes whether the renter is a regular user (level `Registered`) or has not joined the bike-rental organization (`Causal`).

- In your opinion, which of these graphics is most effective at telling an interesting story?

6. Rather than faceting on day of the week, consider creating a new faceting variable like this:

```
mutate(wday = ifelse(lubridate::wday(sdate) %in% c(1,7), "weekend", "weekday"))
```

- Is it better to facet on `wday` and fill with `client`, or vice versa?

Chapter 27

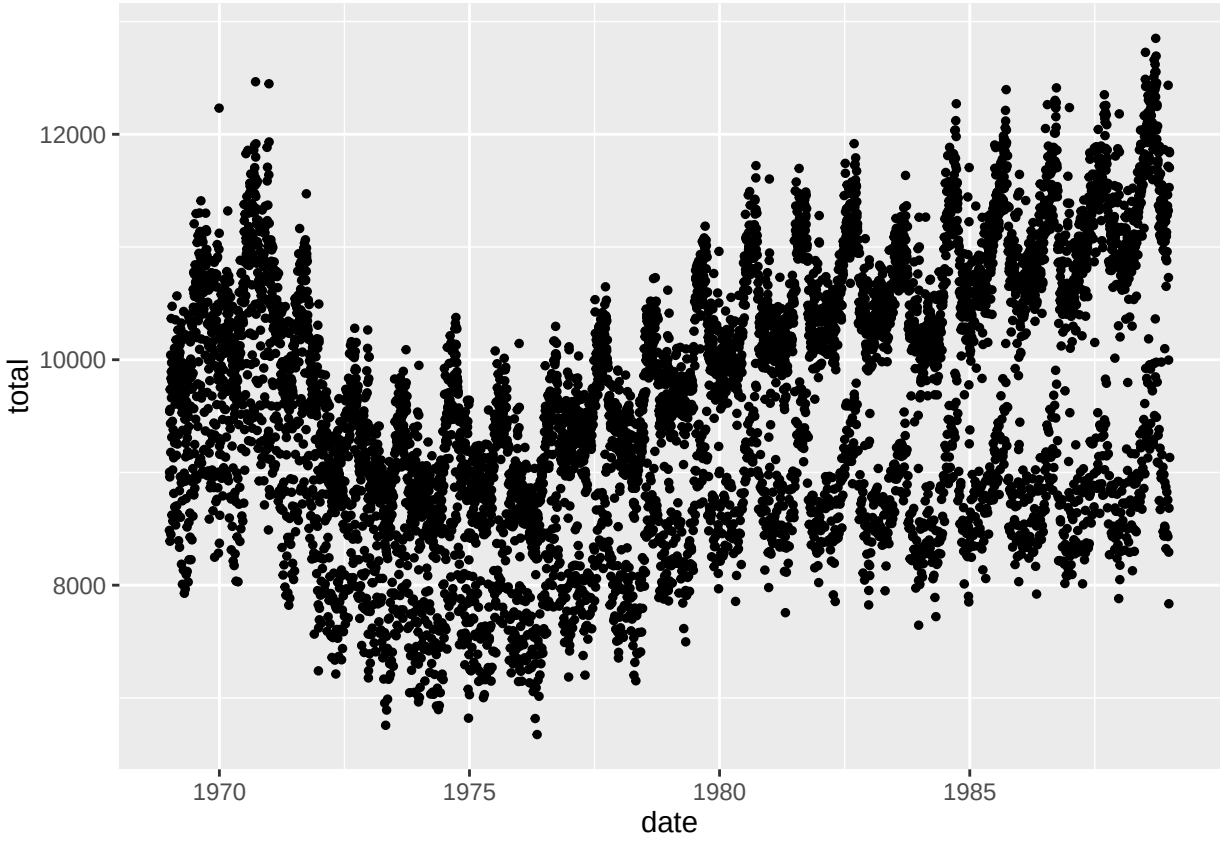
Births and Holidays

PDF handout

The number of daily births in the US varies over the year and from day to day. What's surprising to many people is that the variation from one day to the next can be huge: some days have only about 80% as many births as others. Why?

The data table `Birthdays` in the `mosaicData` package gives the number of births recorded on each day of the year in each state from 1969 to 1988. (It would be nice to have more recent data, but I don't have them at hand.) For this activity, we'll work with data aggregated across the states.

1. Create a new data table, `DailyBirths`, that adds up all the births for each day across all the states. Plot out daily births vs date.



The `date` variable in `Birthdays` prints out in the conventional, human-readable way. But it is actually in a format (called `POSIX` date format) that automatically respects the order of time. The `lubridate` package contains helpful functions that will extract various information about any date. Here are some you might find useful:

- `year()`
- `month()`

- `week()`
- `yday()` — gives the day of the year as a number 1-366. This is often called the "Julian day."
- `mday()` — gives the day of the month as a number 1-31
- `wday()` — gives the weekday (e.g. Monday, Tuesday, …). Use the optional argument `label=TRUE` to have the weekday spelled out rather than given as a number 1-7.

Using these `lubridate` functions, you can easily look at the data in more detail.

2. To examine *seasonality* in birth rates, look at the number of births aggregated over all the years by

   a. each week
   b. each month
   c. each Julian day

3. To examine patterns within the week, look at the number of births by day of the week.

4. Pick a two-year span of the `Birthdays` that falls in the 1980s, say, 1980/1981. Extract out the data just in this interval, calling it `MyTwoYears`. (Hint: `filter()`, `year()`). Plot out the births in this two-year span day by day. Color each date according to its day of the week. Explain the pattern that you see.

27.1 Births and holidays

5. A few days each year don't follow the pattern in (4). We're going to examine the hypothesis that these are holidays. You can find a data set listing US federal holidays at `http://tiny.cc/dcf/US-Holidays.csv`. Read it in as follows:[1]

6. Add a couple of layers to your plot from (4).

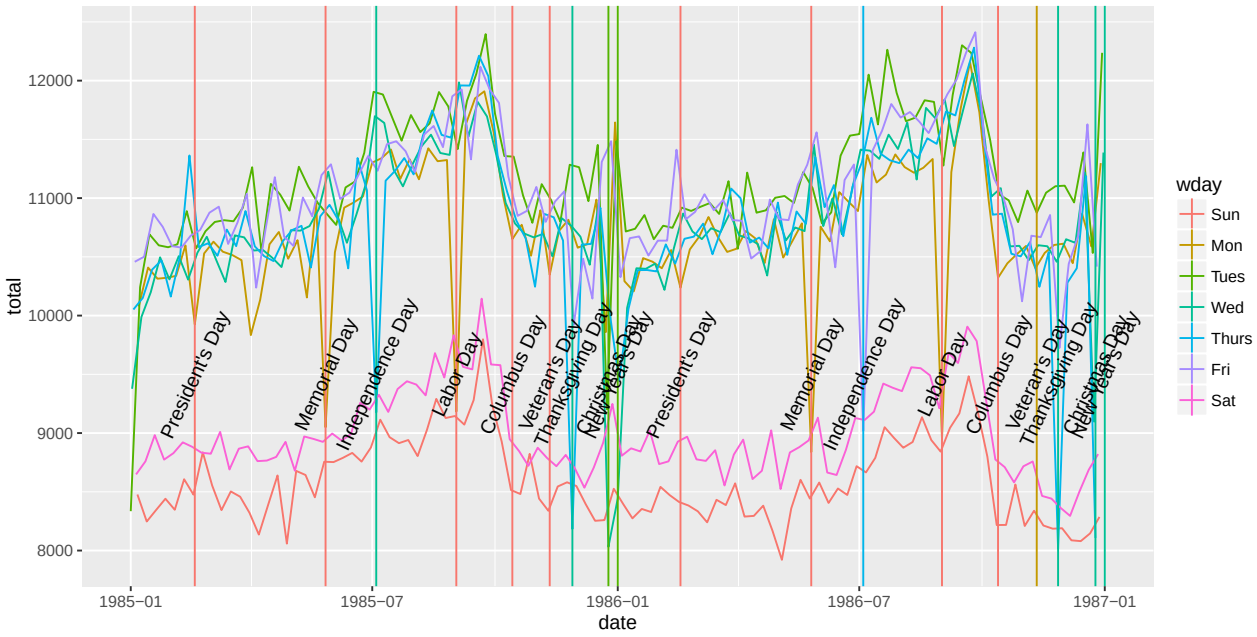
   1. Draw a vertical bar at each date which is a holiday. You'll use the `geom_vline()` glyph. You can give a `data =` argument to `geom_vline()` to tell it to plot out the information from `Holidays` rather than `MyTwoYears`.[2]

   2. Add a text label to each of the vertical bars to identify which holiday it is. Use the `geom_text()` glyph.[3]

[1] The point of the `lubridate::dmy()` function is to convert the character-string date stored in the CSV to a POSIX date-number.

[2] Unfortunately, due to what I believe is a bug in `geom_vline()`, you will have to set the x position of the bars by `as.numeric(date)` rather than just `date()`.

[3] Hints: You'll have to make up a y-coordinate for each label. You can set the orientation of each label with the `angle` aesthetic.



7. The plot in (6) is too busy. Let's explore some other ways to display the data to make it clearer to the view that holidays tend to be low-birth days.

Add a variable to `MyTwoYears` called `is_holiday`. It should be `TRUE` when the day is a holiday, and `FALSE` otherwise. One way to do this is with the transformation verb `%in%`, for instance,

```
is_holiday = date %in% Holidays$date
```

Make a new plot where you map `is_holiday` to the shape or the size aesthetics, or perhaps for faceting. Or perhaps change the way color is used in the graph to show weekends and holidays separately from non-holiday weekdays. Make the graph as simple as you can until you get one that clearly tells the story. You may want to simplify by eliminating any components of the graph (e.g. holiday labels? dots? lines? vertical lines?) that aren't essential to telling the story.

Chapter 28

(PART) Joins and Reshaping

Chapter 29

Combining data from different sources

Glyph-ready data often combines data from different sources.

- Perhaps they come from different experiments or institutions.
- Often, they were collected with different objectives than yours.
- Perhaps they are completely different types of data,

Example: Medicare data

- `MedicareProviders`: Name and location
- `DirectRecoveryGroups`: Descriptions of standardized medical procedures
- `MedicareCharges`: Charges and payments for different DRGs by different providers
- `ZipDemographics`: Population and age structure in each ZIP code.

Example: Holiday births

Goal: Compare the number of births on holidays to those on non-holiday weekdays.

- The `mosaicData::Birthdays` table gives the daily number of births in US states from 1969 to 1988.
- `Birthdays` doesn't *directly* tell us which days are holidays, but …

We can use the `date` variable to look up each case in a list of holidays, e.g.

```
Holidays <- read.file("http://tiny.cc/dcf/US-Holidays.csv")
```

29.1 Relational databases

Storing data in separate tables can be beneficial even when the data are coming from the same source:

- There is no "one size fits all" glyph-ready format. Often the kinds of analysis that will be done are not specifically anticipated when data are collected.
- Glyph-ready data often contains **redundancies**. This makes it hard to update or correct data.

Strategy: Don't even try to smash all data into one big table. Instead, join related tables as needed.

Example: Grades and Enrollment

29.2 Joins

A *join* is a data verb that combines two tables.

- These care called the *left* and the *right* tables.

There are several kinds of join.

- All involve establishing a correspondance — a match — between each case in the left table and zero or more cases in the right table.
- The various joins differ in how they handle multiple matches or missing matches.

29.3 Example: Average class size

Goal: Figure out the average class size seen by each student.

- `enroll` comes from `Courses` table.
- Student (`sid`) comes from `Grades`.
- `sessionID` is in both tables.

Once `Courses` and `Grades` are joined, it's straightforward to find the average enrollment seen by each student.

-

Statistical Digression

Why are these numbers different?

29.4 Establishing a match between cases

A match between a case in the *left* table and a case in the *right* table is made based on the values in pairs of corresponding variables.

- **You** specify which pairs to use.
- A pair is a variable from the left table and a variable from the right table.
- Cases must have *exactly equal* values in the left variable and right variable for a match to be made.

Example:

```
Grades %>%
  left_join(Courses, by = c(sessionID = "sessionID")) %>%
  head(4)
```

The default value of `by=` is all variables with the same names in both tables.
- This is **not reliable** unless you've checked.

29.5 Kinds of join

Different kinds of join have different answers to these questions.

- What to do when there is **no match** between a left case and any right case?
- What to do when there are **multiple matching cases** in the right table for a case in the left table?

Most popular joins: `left_join()` and `inner_join()`

1. No match of a left case to a right case
   - `left_join()` Keep the left case and fill in the new variables with NA
   - `inner_join()` Discard the left case. Less popular joins:
   - `full_join()` Keep left case as well as unmatched right cases.
   - `semi_join()` Discard the left case.
   - `anti_join()` Keep the left case but discard any left case with a match in the right table
2. Multiple matches of right cases to a left case
   - `left_join()` and `inner_join()` do the same thing: Keep **all combinations**. Less popular joins:

- `full_join()` Keep all combinations.
- `semi_join()` Keep just one copy of the left case.
- `anti_join()` Discard the left case.

29.6 Example: Grade-point averages

Here are three data tables relating student grades in courses at Macalester in 2005

```
Grades <- read.file("http://tiny.cc/mosaic/grades.csv")
Courses <- read.file("http://tiny.cc/mosaic/courses.csv")
GradePoint <- read.file("http://tiny.cc/mosaic/grade-to-number.csv")
```

29.7 Activity: Which to Join?

For each of these, say what tables you would need to join and what the corresponding variables will be.

1. How many students in each department?
2. What fraction of grades are below B+ in each department?
3. What's the grade-point average (GPA) for each student?
4. Grade-point average for each department or instructor
5. What's the 95% confidence interval on the GPA for each student?
6. (Statistically more sophisticated) To what extent does the grade reflect the student or the department or instructor?

29.8 Example: Joining for cleaning

Birds at the Ordway Nature Preserve

As a group, let's correct the species names using this data table.

- Count how many birds there are of each species in `OrdwayBirds` using the corrected species names.

Chapter 30

Wide vs Narrow Data Tables

A data table is comprises *cases* and *variables*.

Each *variable* comprises *values* (or levels).

There is no hard distinction between a variable and a value. What's a variable in one situation may be a value in another, and vice versa.

A data table

30.1 Cases, Variables, and Values

- Variables: Who, X, and Y
  - Values:
    * Who is a person's name
    * X is numeric
    * Y is a language name
    * dorm is a building name
- Cases: Alice, Lesley, Yu

30.2 Two formats

- Narrow
- Wide

30.3 Narrow and Wide

Data in Key/Value format are **narrow**

The corresponding **wide** format has

- separate variables for each level in Key
- sets the values for those variables from the info in Value

Narrow:

Wide:

30.4 Narrow is relative

30.5 Too narrow

There's nothing to identify a case!

30.6 Gather — from Wide to Narrow

Syntax:

```
WideInput %>%
  gather(key_name, value_name, ...)
```

The `...` are the variables to be gathered together, e.g.

```
StudentsNarrow <- Students %>% gather(key, value, x, y)
```

30.7 Cases in Narrow data

Aside from Key and Value, all the other variables identify the case.

The gathering makes multiple rows for each row in the wide form. The variables **not** used for narrowing are copied into the new multiple cases.

30.8 Spread — from Narrow to Wide

Syntax:

```
NarrowInput %>% spread(key, value)
```

Process:

1. Group by **all** variables other than Key and Value These groups become the cases
2. Create new variables for each level in Key
3. Within each group, spread out the Values into the new variables.

Chapter 31

Biblical Names

A list of Bible-related names is available this way:

Using this data table and `BabyNames`:

- Make a data table showing the most popular biblenames over all the years.
- Make an informative plot showing the trends over the years of Bible-related names as a proportion of all names.

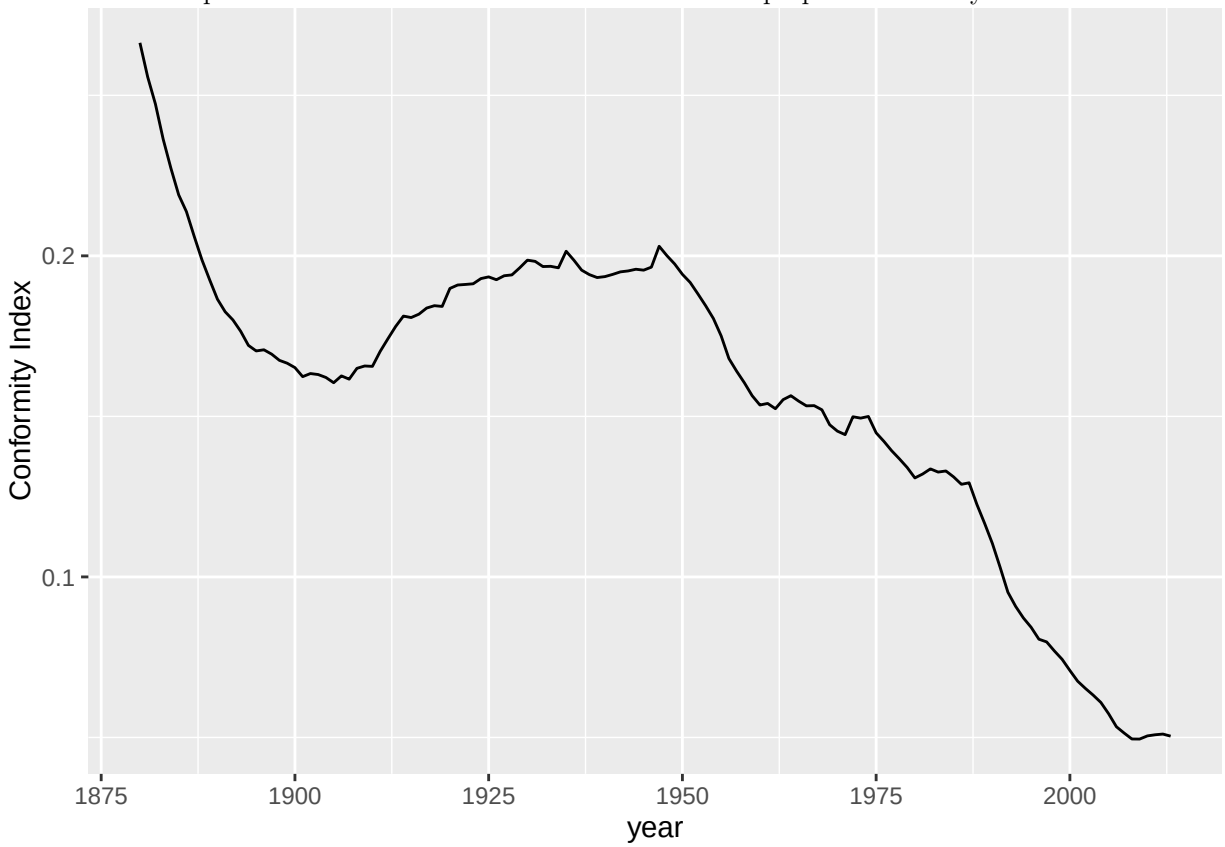
Chapter 32

Conformity and Names

Calculate the total number of babies given the 10 most popular names each year as a fraction of the total number of babies born that year.

Question: What would have happened if `group_by()` came after `filter()`?

We'll need an operation to combine two tables to find the proportion each year. Here's a teaser:



Chapter 33

Example: Gender-neutral names: when wide is easier

Some operations are easy in wide format, but hard in narrow and *vice versa*

33.1 Excerpt from `BabyNames`

Questions:

1. How many babies of each name and sex?
2. For each name, is it primarily given to girls or boys? Which names are gender neutral?

33.2 In narrow format

1. How many babies of each name and sex?

Easy!

33.3 In Wide format

2. For each name, is it primarily given to girls or boys? Which names are gender neutral?

- `sex` is the key variable
- `total` is the value variable

```
BabyTotalsWide <- BabyTotals %>%
  spread(sex, total)
BabyTotalsWide
```

33.4 With sexes side by side ...

Chapter 34

Example: A Penny Collection

The same data can be represented in different ways. One or another way may be easiest, depending on the sort of calculation you are performing.

A friend, when cleaning out a closet in 2016, came across a small, glass bottle with coins in it. The coins were US 1-cent pieces, called "pennies". All of the coins were from before 1959. As with current pennies, the pennies had a picture of Lincoln on one side — the "obverse". On the reverse side, the old-style pennies have a "wheat" design:

The friend was interested to know if these coins were rare. First step in finding out, make a catalog of what he had. She noticed that some of the coins were stamped with an "S" and some with a "D," but many had no stamp at all. So she sorted out the coins by year and stamp, and counted.

34.1 Three formats for the penny data table

Format 1

After counting the pennies in each stack, she compiled the data table available at `"http://tiny.cc/dcf/my_pennies_format_`

Questions:

1. What is the case in the `Format_1` data table?
2. How would you wrangle the table to produce a count of all the pennies, together?
3. How would you calculate the mean age of the pennies? (Hint: not so easy. Multiply the year by the penny counts, add up, and divide by the total number of pennies.)

Format 2

Seeing that these calculations are hard to read and understand, the collector decided to reformat the data to look like this:

year	count	stamp
1958	4	none
1958	6	D
1957	11	none
1957	9	D
⋮	⋮	⋮

The complete data set in this format is available at `"http://tiny.cc/dcf/my_pennies_format_2.csv"`.

1. What is the case in the `Format_1` data table?
2. How would you wrangle the table to produce a count of all the pennies, together?
3. How would you calculate the mean age of the pennies? (Hint: This is much easier than for `Format_1`.)
4. How do you wrangle the `Format_1` data table into the form of `Format_2`.
5. The two formats differ: one is in narrow form, the other in wide form. Which is which?

Question:

1. Suppose that some additional pennies were added to the collection with new stamps, say "W" and "Z". Which of the formats, wide or narrow, would be better suited so that your wrangling statements to find the number of coins and the mean age would continue to work?
2. Counts, wide: year, Phila., Denver, SF
   - What's the case?
3. Counts, narrow: year, origin, count
   - Converting between (1) and (2)
4. Pennies, narrow: year, origin
5. Pennies, wide
   - Converting between (3) and (4)
   - Producing (1) and (2) from (4)

Format 3

Another friend suggested that it might be better to store the data with one row for each penny, like this:

The complete data set in this format is available at `"http://tiny.cc/dcf/my_pennies_format_3.csv"`.

Questions

1. How can you tell the case in `Format_3` is a single penny?
2. How would you count the number of pennies in the collection?
3. How would you find the mean year for the coins in the collection?

34.2 More tasks

Use at least one of the formats to do the following. Even better if you try to do it with more than one format, but this might be hard

1. Identify the years for which D and S stamps together outnumber P stamps.
2. Calculate the mean year for each stamp.

34.3 Extending the data

- Suppose you want a "condition" for each coin?
- Suppose you want to include coins other than pennies?
- Suppose you want to have prices that depend on year, origin, denomination, condition?

Chapter 35

Stocks and dividends

PDF handout

Only the PDF handout is available, not the HTML format document.

Chapter 36

A Graph for the Economist

PDF handout

The *Economist* is a well-regarded weekly news magazine. The following graphic accompanied their article about the release of the "College Scorecard" data in Sept. 2015.

Your task is to reproduce this graph from the College Scorecard data, and perhaps enhance it.

36.1 The data

The Scorecard data is too voluminous to work with conveniently in class; it takes too long to download. You'll be working with a subset available at `tiny.cc/dcf/ScorecardSmall.Rda` which contains a single object, the data table `ScorecardSmall`.

The subset includes all 7804 institutions in the original 2013 Scorecard file, but just 54 variables. Some that you may be interested in are:

1. `CONTROL`: public (1) or private (2) institution. (You can discard cases with `CONTROL == 3`. They are not in the Economist's graphic.)
2. `INSTNM`: name of the institution
3. `ADM_RATE`: admissions rate in percent
4. `CCSIZSET`: Carnegie size classification of the institution. Values 1, 6, 7, 8 correspond to schools with fewer than 1000 students.
5. `AVGFACSAL`: Average faculty salary per month
6. `TUITFTE`: Tuition revenue received by the institution per student full-time-equivalent.
7. `NPT4_PUB`: average net cost for students in public institutions
8. `NPT4_PRIV`: average net cost for students in private institutions
9. `NPT41_PUB` : average net cost for students at public institutions whose families are in the lowest of five income groups. Similarly, `NPT42_PUB` is for students whose family income is in the 2nd group, and so on up to the 5th group. The groups are defined as $0 to $30K per year, $30-48K, $48-75K, $75-110K, $110K or more. There is also `NPT41_PRIV`, and so on, for private institutions.

All of the `NPT4` variables are for students receiving aid from the federal government under Title IV.

36.2 What's the case?

The case in the Scorecard data is an institution. In the *Economist* graphic, however, the case is a level of family income (as in `NPT4`) at an institution. That is, from the perspective of the graphic, the Scorecard data is in wide form. You'll have to convert it to narrow form to make the graph.

1. Select just the variables you need from the Scorecard data.
2. Use `gather()` to convert from wide to narrow format.
3. After (2) you will have a variable with levels like `NPT43_PUB`, `NPT45_PRIV`, etc. You will want to translate these to `Q3`, `Q5`, etc. For your convenience, the file `http://tiny.cc/dcf/NPT4-names.csv` contains a table with the appropriate translations. You can use a join of the narrow-format Scorecard data with this table to perform the translations.

Chapter 37

Statistics: Collective properties of data

Whenever you use `summarise()` you are reducing a set of cases into a single case. That single case reflects the *collective properties* of the cases. For instance, you might calculate the mean of some quantity. That mean takes in the values for *all* of the cases. Even when you are finding the value for a single case, say the one with the maximum value, that case is the maximum only with respect to the other cases. If those cases had been different, the max might change.

Of course, we intend the summary to be *representative* of the cases being summarized, particularly when the quantity being calculated is meant to describe a "typical" value for all of the cases. That being the case, it's fair to ask *how representative* that summary is of the set of cases, or *how well* that summary represents the cases. This is a central issue in basic statistics. Statistics provides a framework for approaching that question of *how representative* a summary is.

To illustrate, let's look at how people rate movies.

The three data tables in the `MovieLens.rda` file comprise a set of 100,000 ratings of movies by individuals. These data were collected in the late 1990s by the *grouplens* research team at the University of Minnesota. Grouplens provides the data directly at http://grouplens.org/datasets/movielens/100k/. `MovieLens.rda` is a reformatting of the data that makes the file substantially smaller. You can access the file in this way.

Once the data are downloaded, you can use `load()` to bring the data tables into your R session.

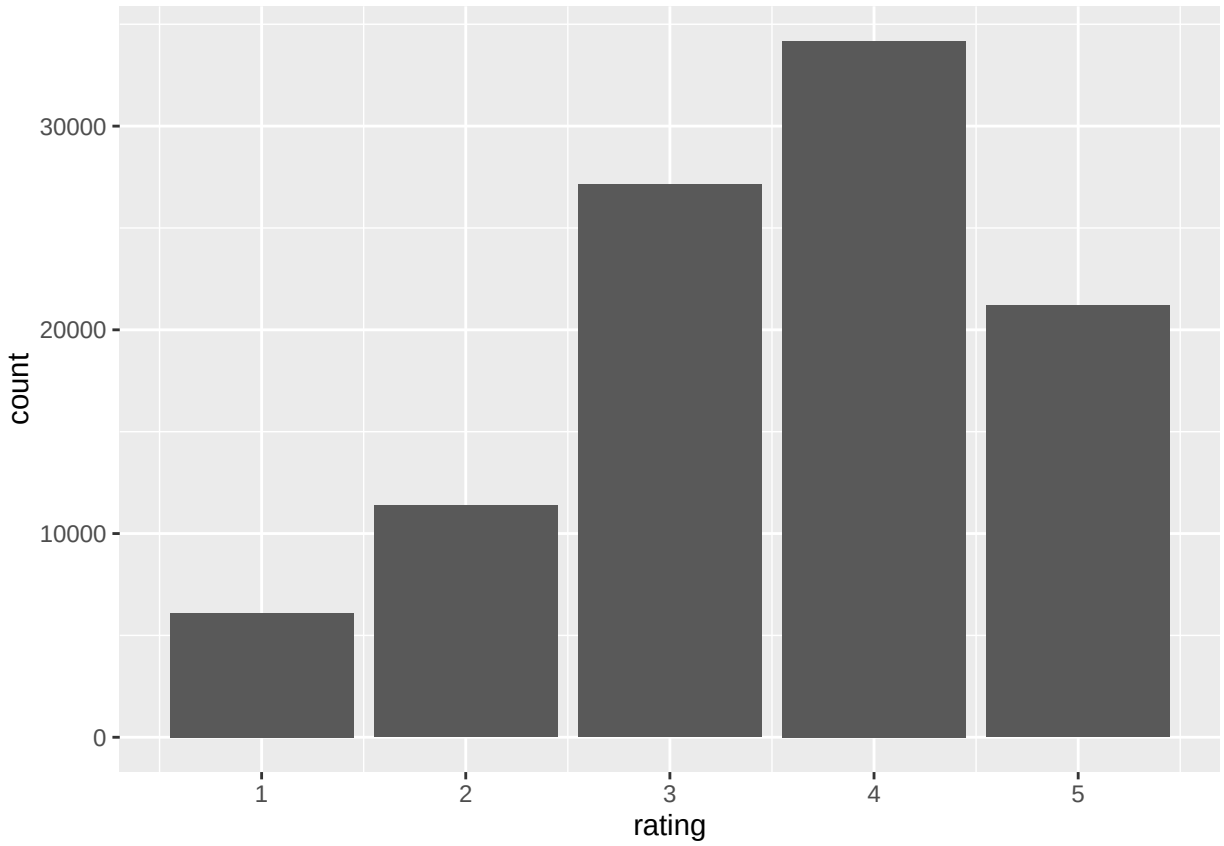
`MovieLens.rda` contains three data tables:

- `Ratings` has the individual movie ratings and the time at which they were entered. It also includes an ID variable for both the user and the movie.
- `Movies` provides the name of the movie and information about genres.
- `Users` gives basic information about the person who made the rating.

37.1 The mean as a summary

Calculating the mean rating is easy:

So how representative is the mean rating? One way to assess this is to display the distribution of ratings:



The mean value, 3.52986, falls right in the middle of this distribution.

One way to assess the representativeness of the mean is to calculate an auxiliary statistic that summarizes how far from the mean a typical case is. The standard way to measure this typical deviation from the mean is called the *standard deviation* and is computed with the `sd()` reduction function.

The typical rating is within about 1.1 points of the mean.

Statisticians compute another statistic to indicate how *uncertain* the mean is. The uncertainty doesn't arise from the calculation itself — computer calculations such as `mean()` are very reliable. The uncertainty refers to a kind of thought experiment.

Suppose that a much larger set of ratings had been collected, from a much larger group of people. Imagine that the particular cases we are working with in `Ratings` were a random choice from this much larger set. And then imagine that many other, similar datasets were created, each of which is a random selection from the much larger dataset. The means calculated from these many different datasets would presumably differ from the one we get from our particular set in `Ratings`. The uncertainty refers to how much the means would differ among the hypothetical, thought-experiment data sets.

The most common statistical measure of this uncertainty of the mean is called the "standard error of the mean." Perhaps surprisingly, there is a simple method to calculate the standard error of the mean even without having to generate a new data set.

The value of the standard error of the mean is very small compared to the mean. Statisticians display the uncertainty with an interval of ± 2 times the reliability around the mean, in other words, 3.530 ± 0.007. Note that the values have been somewhat rounded. As a rule, there's very little meaning to the second (non-zero) digit of the standard error. And there's no point in reporting the mean itself to a precision that goes beyond that of the standard error.

We could reasonably call this interval the "uncertainty interval." But the standard term for it in statistics is the "confidence interval"

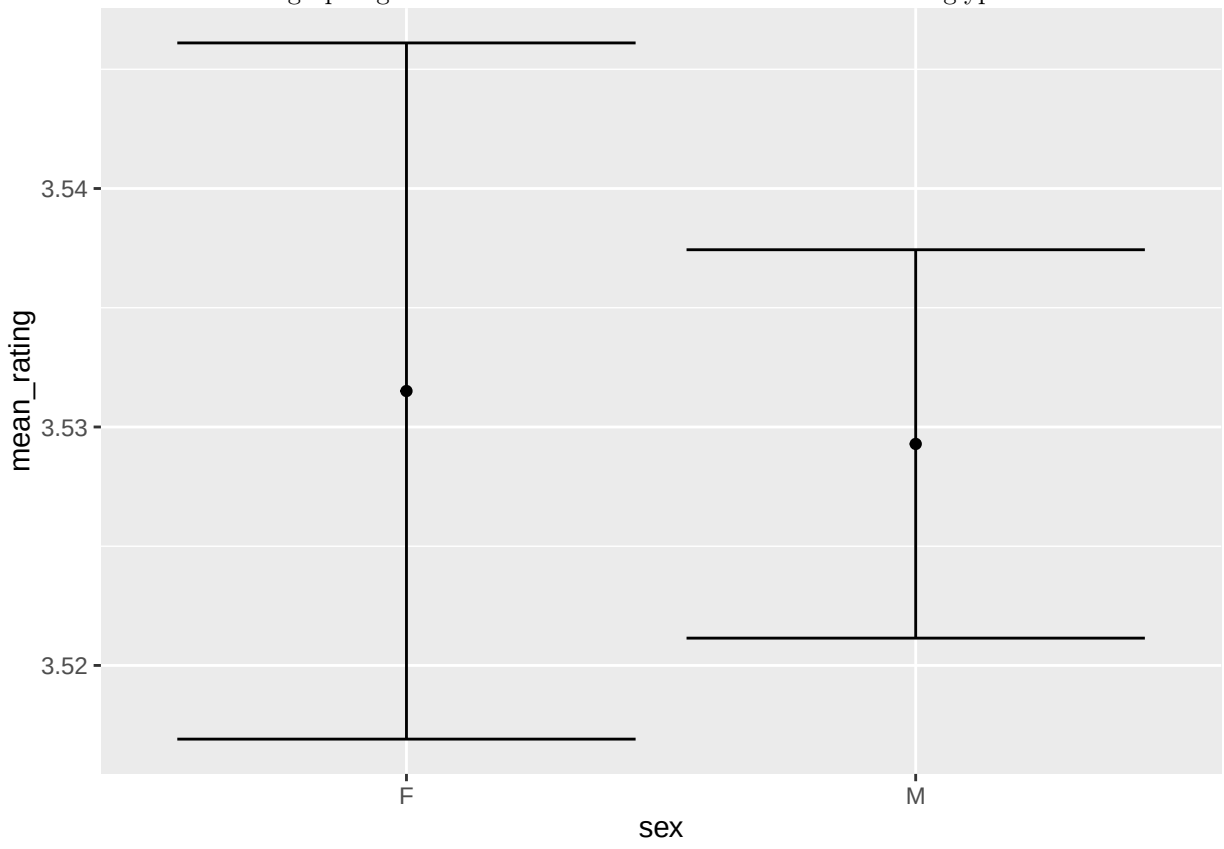
Why should we care about the uncertainty of the mean? The typical situation where this is an issue is when we are comparing two means to see whether they are different. For instance, let's see if women and men have different mean movie ratings. To start, we need to identify each rating with the sex of the rater.

Now we're in a position to compute the mean for each sex, together with the uncertainty in those means.

The means for men and women are a bit different — women give higher ratings by 0.0023. But look at the reliabilities. They are larger than the difference in means. This suggests that although the precise numerical value of the means is different, the difference is itself not reliable. It might as well be zero!

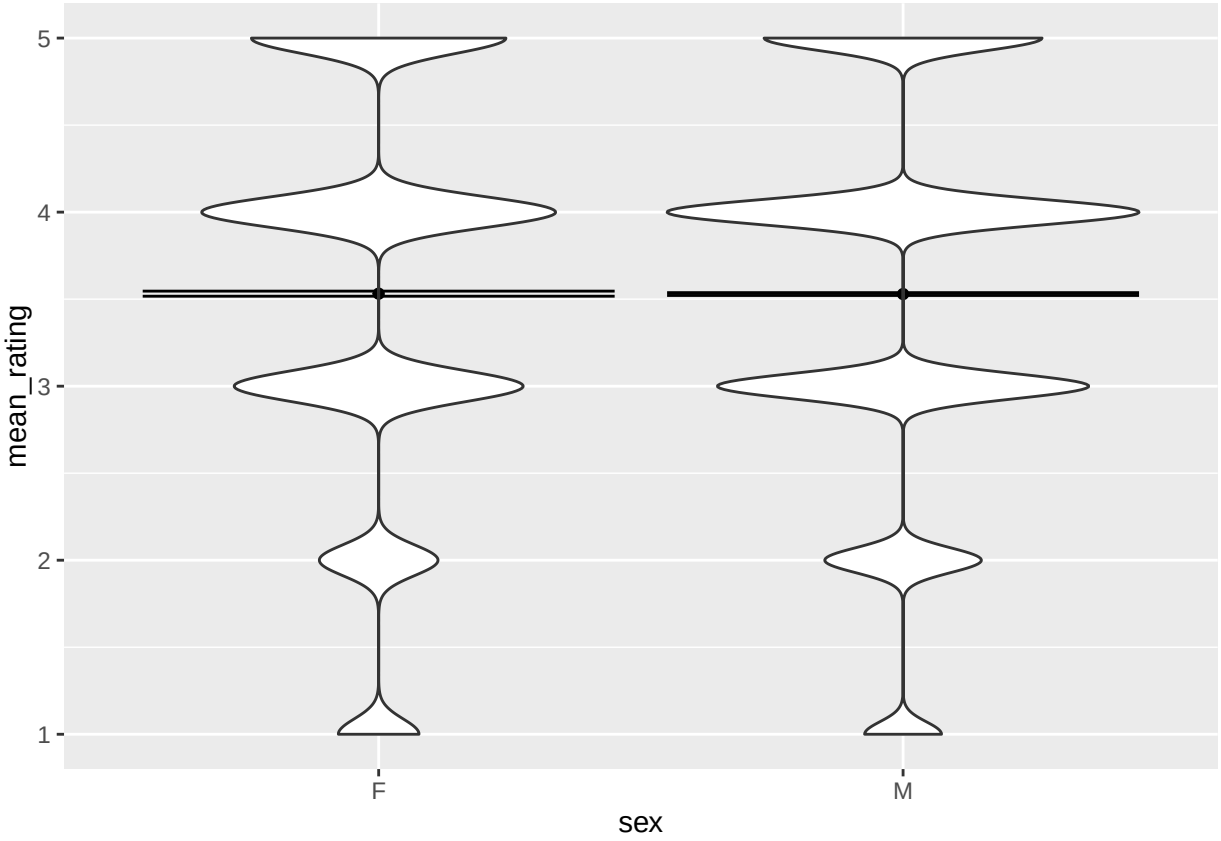
Plotting out the means along with their uncertainties.

There is a standard format for graphing the means and their uncertainties that involve a glyph called an "error bar."



As always when reading a graph, note the scales.

Many people mistakenly interpret a plot like this as indicating the range of ratings themselves, rather than the uncertainty in the mean rating. So it can be helpful to display the distribution of the ratings themselves on the same graph.



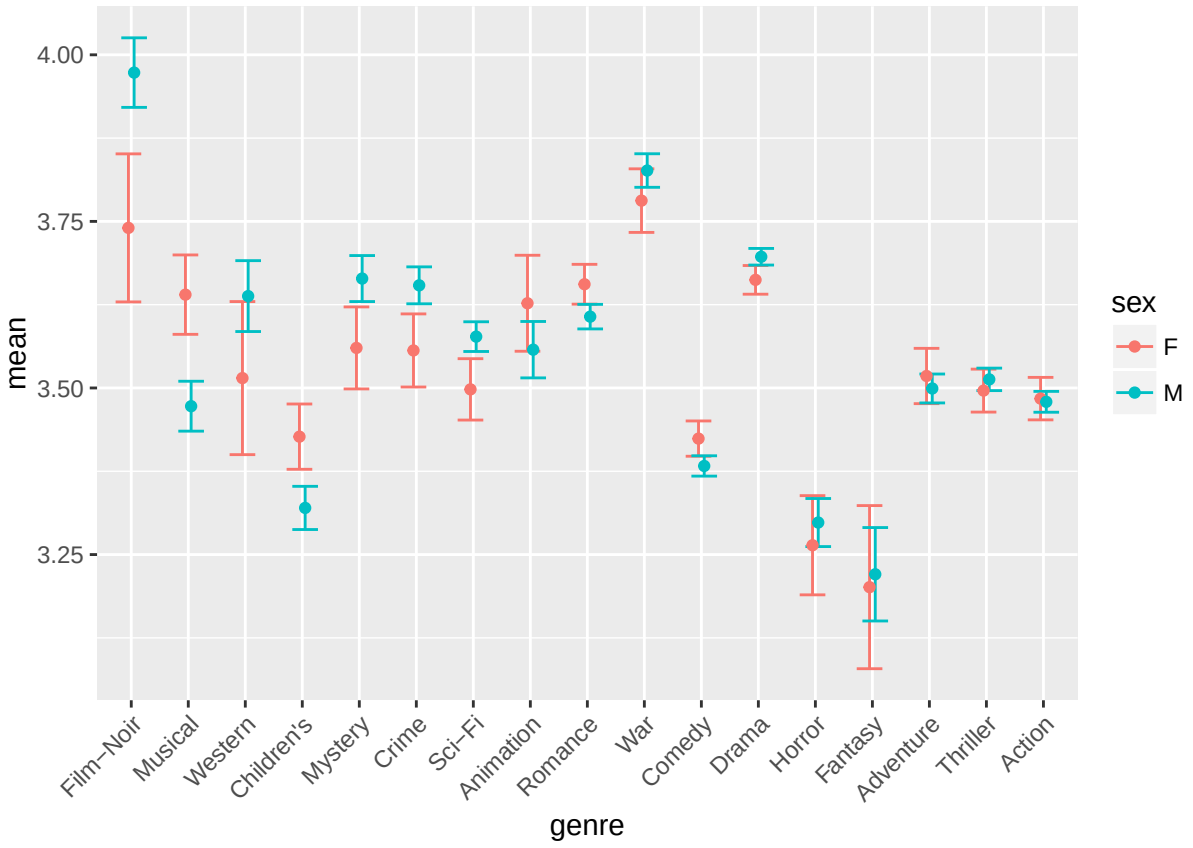
Again, note the scales. You can see that the uncertainty in the mean is much, much smaller than the distribution of ratings.

Activity

Find the means broken down by sex and movie genre. Do the sexes rate individual genres differently?

Think carefully about how you are going to combine the genre information with the ratings. Movies often fall into more than one genre. In calculating the means, you might want to create a new case of the values in `Ratings` for each of the genres of a movie.

Read each of the following chunks and say what it is doing.

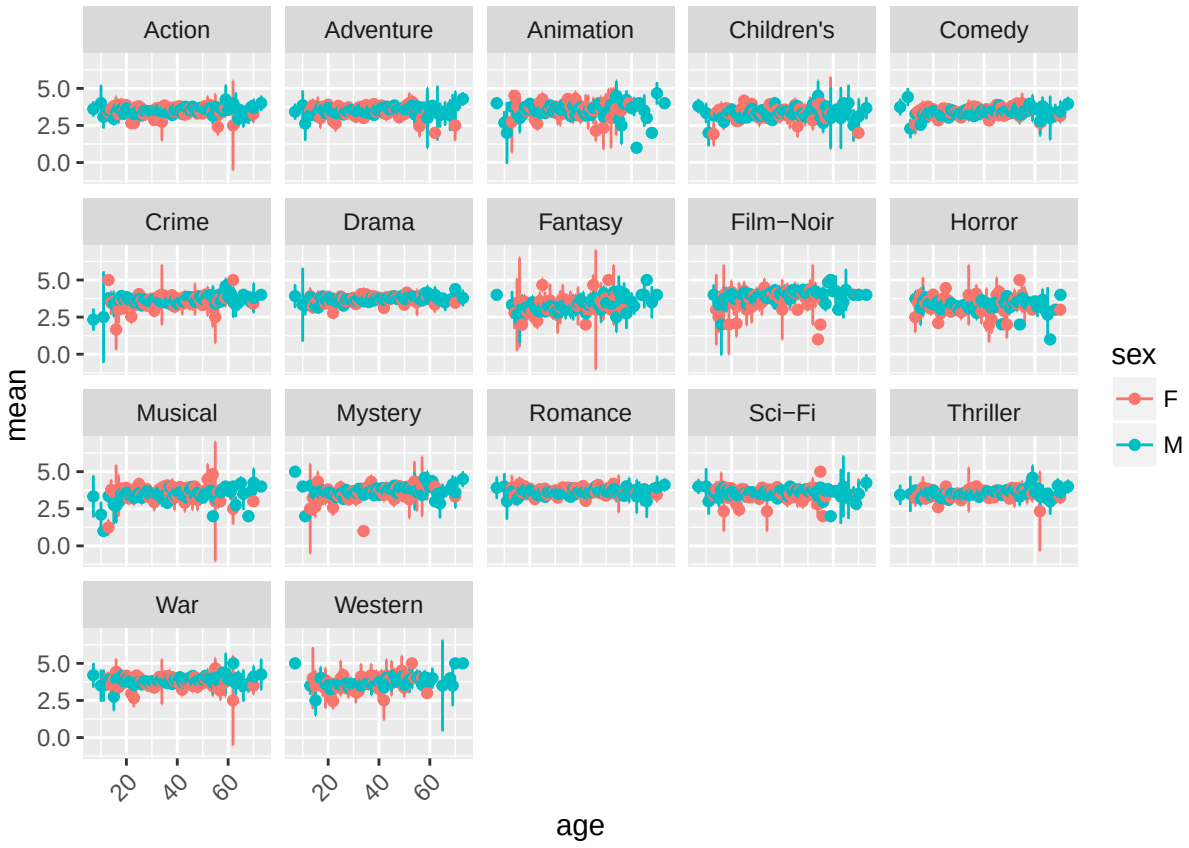


37.2 Continuous explanatory variables

In the previous, we broke down the ratings by sex and genre. But maybe age is an important determinant of opinion.

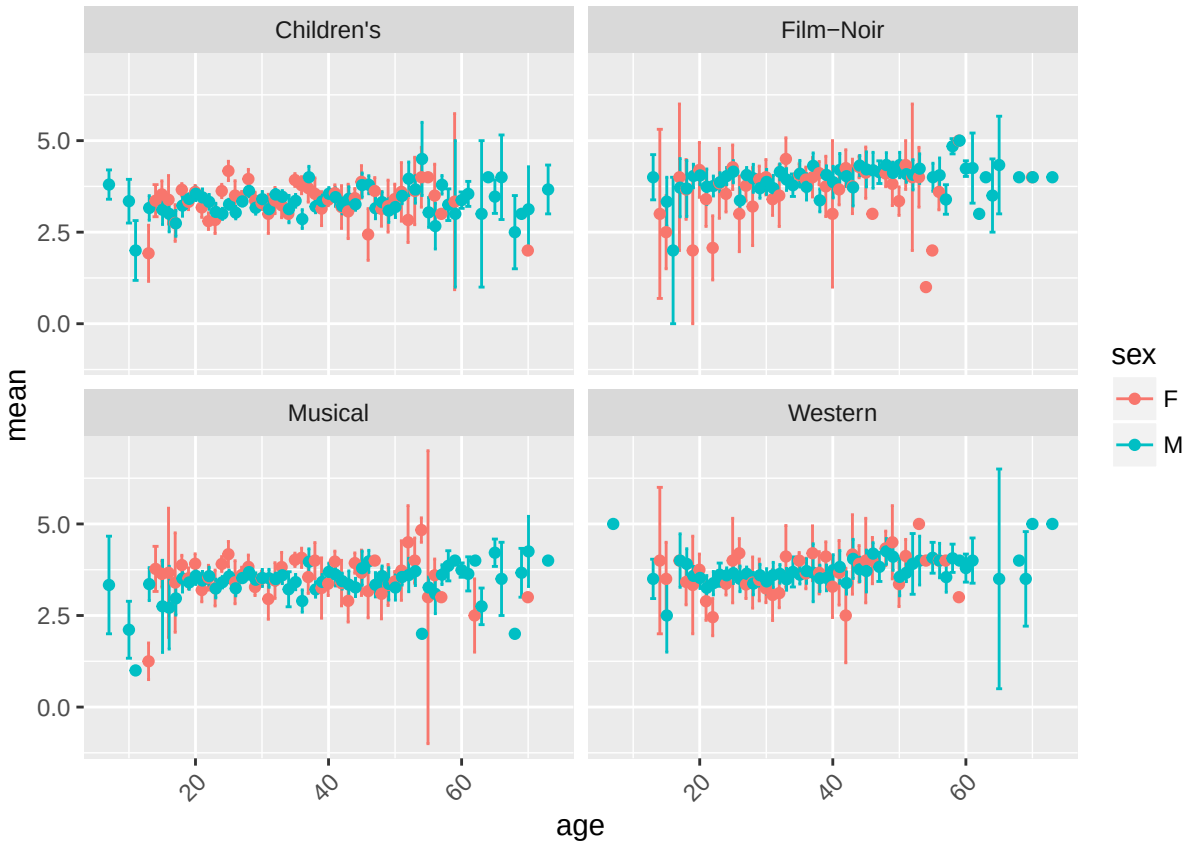
In these chunks, we simply add age as another grouping variable when computing the mean and the uncertainty in the mean.

In plotting this, what roles would you assign to each of the variables?



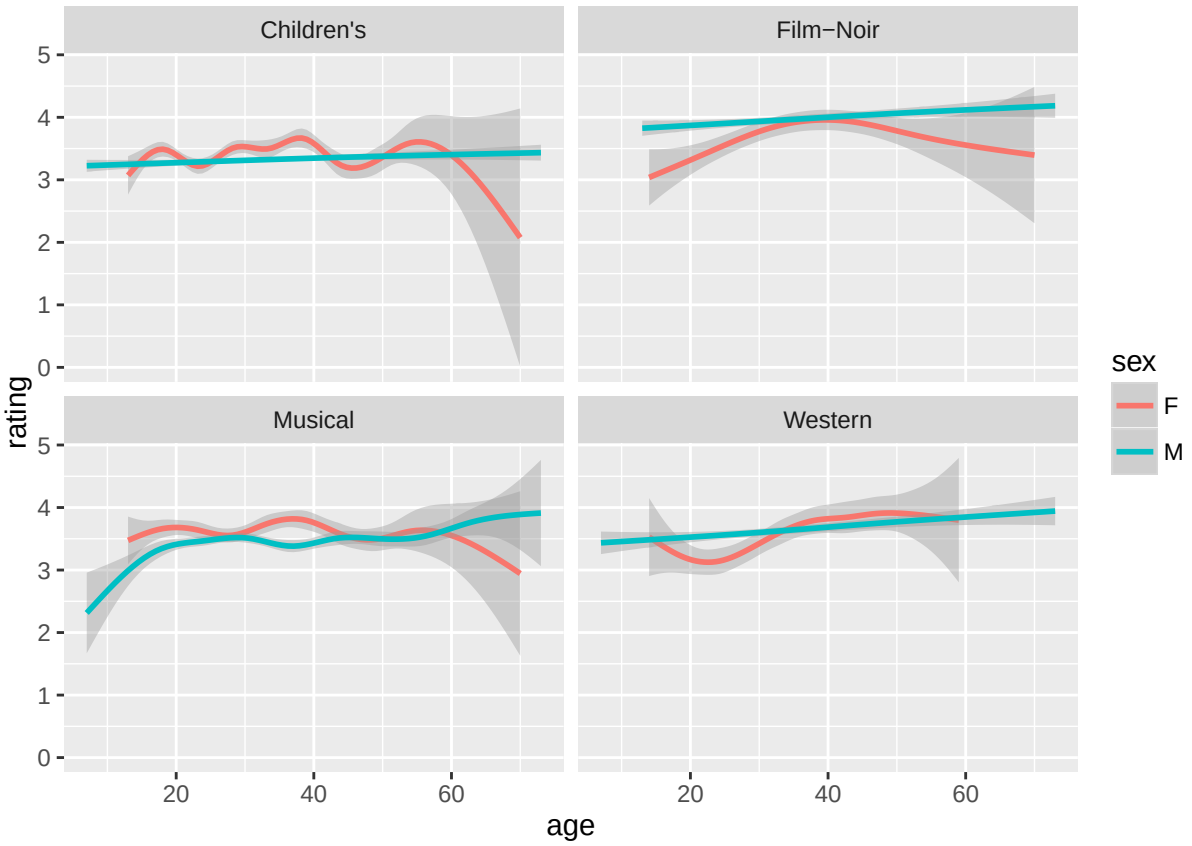
How could you simplify this plot?

Let's focus on just a few genres, the ones that showed large differences between the sexes.



One of the problems with this plot is that the error bars are much larger than they were in some of the previous plots. This makes it hard to see differences between the sexes. Why this growth in the error bars? Because we've divided the data up into about 50 different ages. Roughly, this means that each error bar is based on about 1/50th as much data. Since the uncertainty goes as $1/\sqrt{(n)}$, this means that the error bars will be about $\sqrt{50} \approx 7$ times bigger than before. You could fix this by, say, dividing up into just a few age groups.

A statistical method called "regression" enables the mean rating to be assessed by age without having to break up age into any groups at all. In the next plot, we'll use a regression method called "smoothing" to do this, transforming the error bars into error bands.



Look at the film-noir genre. Men's mean ratings are high and go up over the years. For women, the pattern is different. They don't like film-noir as much as men, except around age 40. (Note: These data are not tracing individual people as they age. Instead, these are different people who were born at different years.)

Project idea: How do ratings compare by month or time of day? Is it possible to tell whether the times of day are keyed to the local clock time, or the time at which the database server received the information.

37.3 Correlations

Another very important technique in introductory statistics involves "correlation."

Consider three situations about how two different variables might relate to one another:

1. When one variable is up, the other one tends to be up as well. This is called a "positive correlation."
2. When one variable is up, the other could equally well be up or down. This is called "uncorrelated."
3. When one variable is up, the other one tends to be down. This is called a "negative correlation."

Now "tends to" can be a matter of degree. The *correlation coefficient* indicates how strong the tendency is on a scale of zero to 1. Zero means uncorrelated. One means "exactly in line with one another." The sign on the correlation coefficient indicates whether the correlation is positive or negative. If the variables are uncorrelated, the correlation coefficient is near zero.

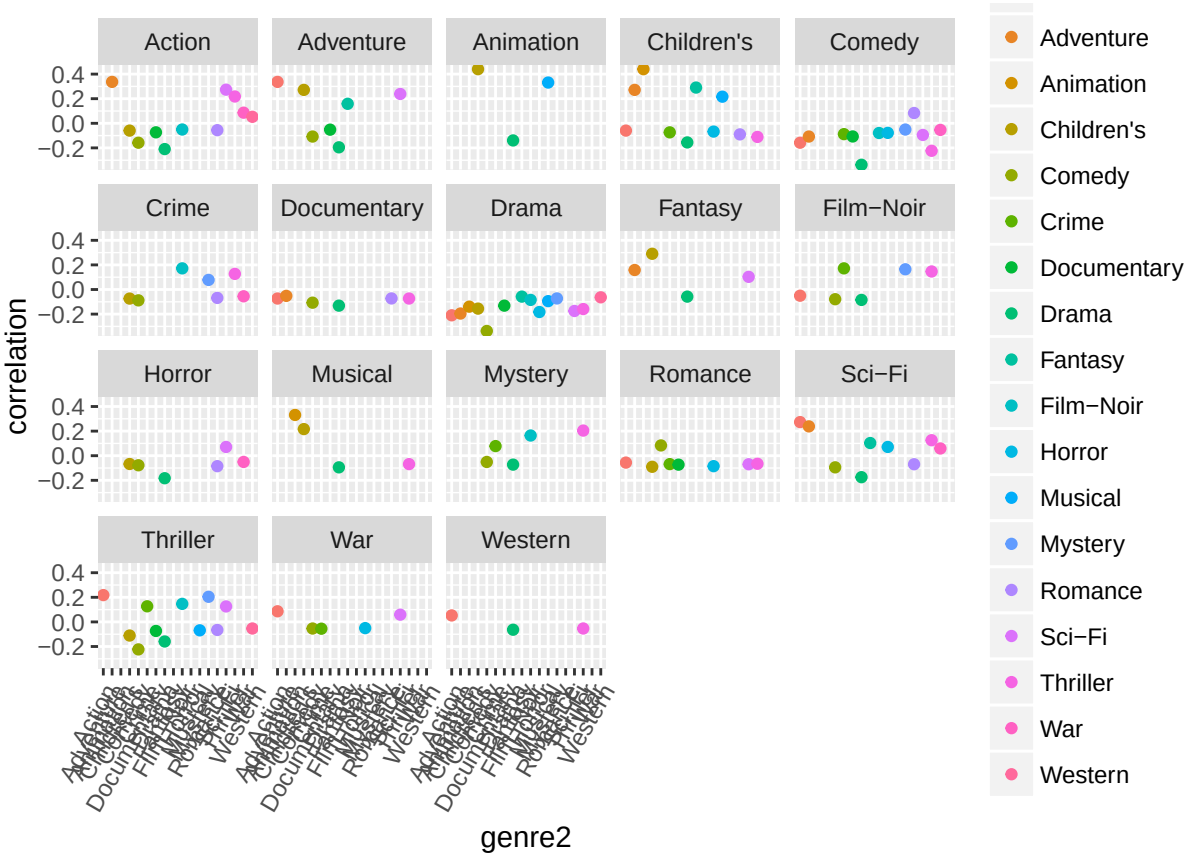
To illustrate, let's look at the correlation between musical and war genres, and also the correlation between crime and film-noir genres. We'll base the correlation on whether the genres are aligned from movie to movie.

It looks like being a war movie tells you nothing about whether it is a musical, while being a crime movie tells a bit about whether it is also a film-noir.

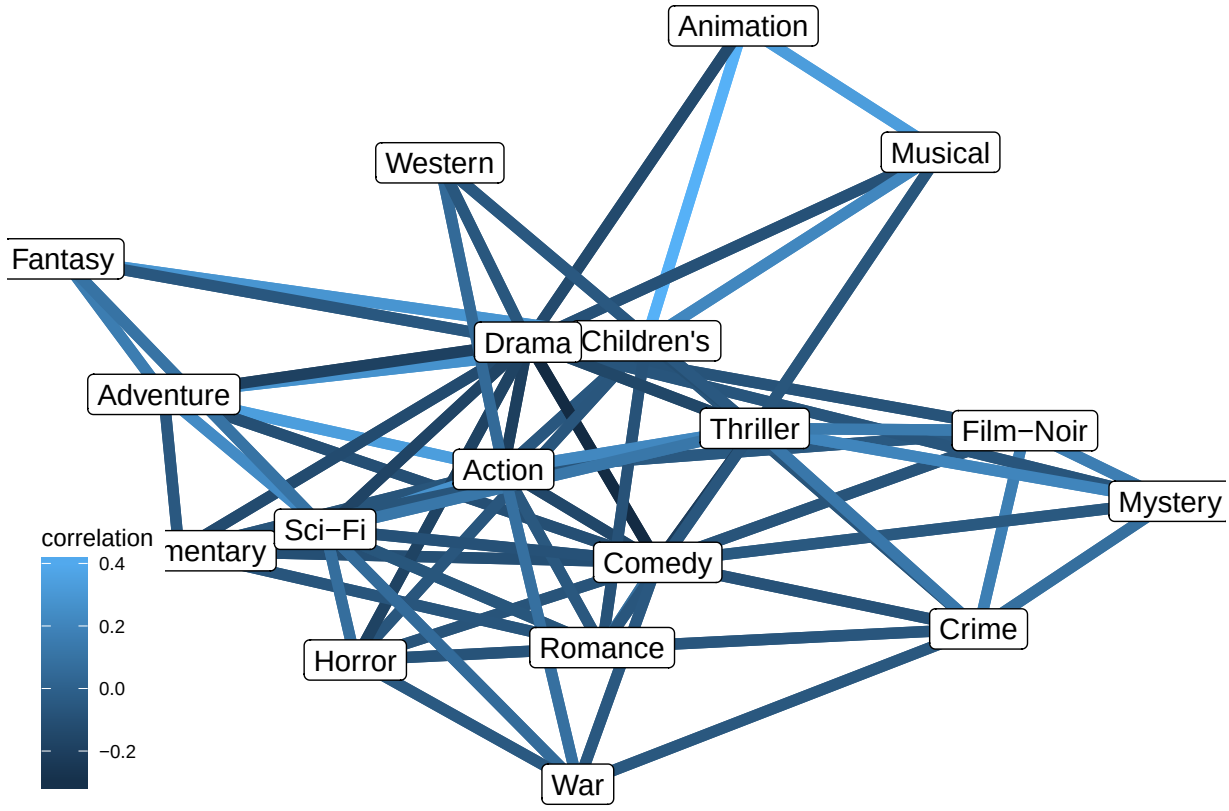
It's also possible to calculate an uncertainty in the correlation coefficients. This can help in deciding whether to treat the correlation as different from zero.

There's no reason in these data to believe that the correlation between war and musical genres is anything but zero. In contrast, crime and film-noir have a positive correlation, but nowhere near perfect correlation.

37.4 A primitive display of genre correlations



37.5 A network display



As a network

Chapter 38

Calling 311

In New York City, dialing 311 connects you to a complaint/request service.[1] New York provides information about individual calls to 311 on a web site.

I want to see how the number of 311 calls for different reasons varies over time of day and day of the week. I'm also interested in finding out whether there are hot spots for 311 calls, and how this depends on factors such as income.

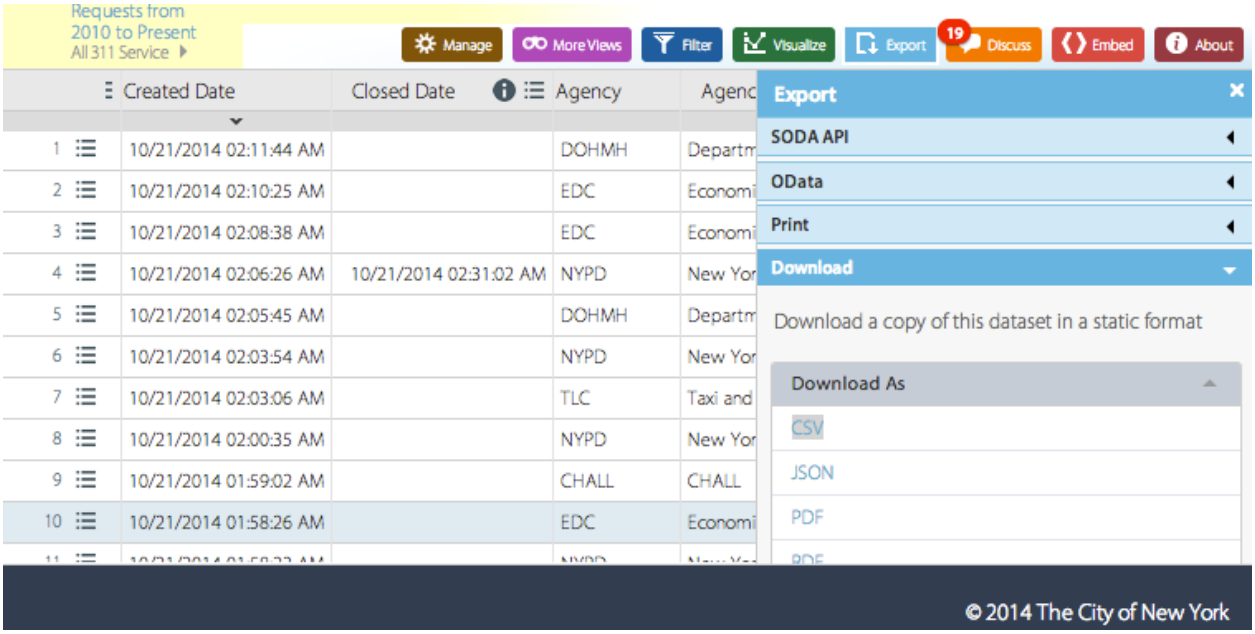


Figure 38.1: 311 data web site

That site provides an "Export" feature. You can download the data in several formats: CSV is appropriate. Using a web browser, I downloaded it to my `Downloads` directory.

As of October 12, 2016, the data file is more than 9GB in size and encompasses almost 14 million complaints. It has 53 variables, but most of these are not relevant to most of the complaints or requests. For example, `"Bridge Highway Segment"` is relevant only to 311 calls that have to do with bridges, and `"Park Facility Name"` only to calls involving parks.

[1] In the US, 911 is the phone number for emergency services. 311 is for non-emergency services.

Whenever the size of a data set is larger than the amount of random-access memory in your computer, it's likely that you will need special techniques to analyze it. This applies also to data that is about the same size as the random-access memory. On the computer on which this is being written, a 2013 MacBook Pro, I've got 16GB of random-access memory. So, rather than just reading in the data file to R, all in one go, I'm going to be a little careful and try to figure out what will be easy and what won't.

Many file-reading functions in R will let you read just a few rows from a data set. Here, I'll read just the first two thousand rows, to get an idea of which variables are important.

To illustrate, `"Taxi Pick Up Location"` is blank in the vast majority of the first two thousand cases, so presumably this is relevent to just a small fraction of complaints.

To get started, `CreatedDate`, and `Complaint Type`, seem like good variables to explore. These are columns 2 and 6. I can tell `fread()` just to read those columns.

I'll read in a few rows just to make sure things are working:

Can I use this method to read in a large number of cases? Rather than just trying to read the whole data set, I'll try it for different numbers of rows and measure how long it takes. I built up slowly, first testing 10 rows, then 100, then 1000. I found I could get to one-million rows while still just taking a few seconds. So it seem plausible that I can read in the whole thing with just the regular kinds of commands.

Now I'll check whether I can convert the `"Created Date"` variable to a time object. Rather than use all 13 million cases, I'll start small again and work up with the number of rows to read.

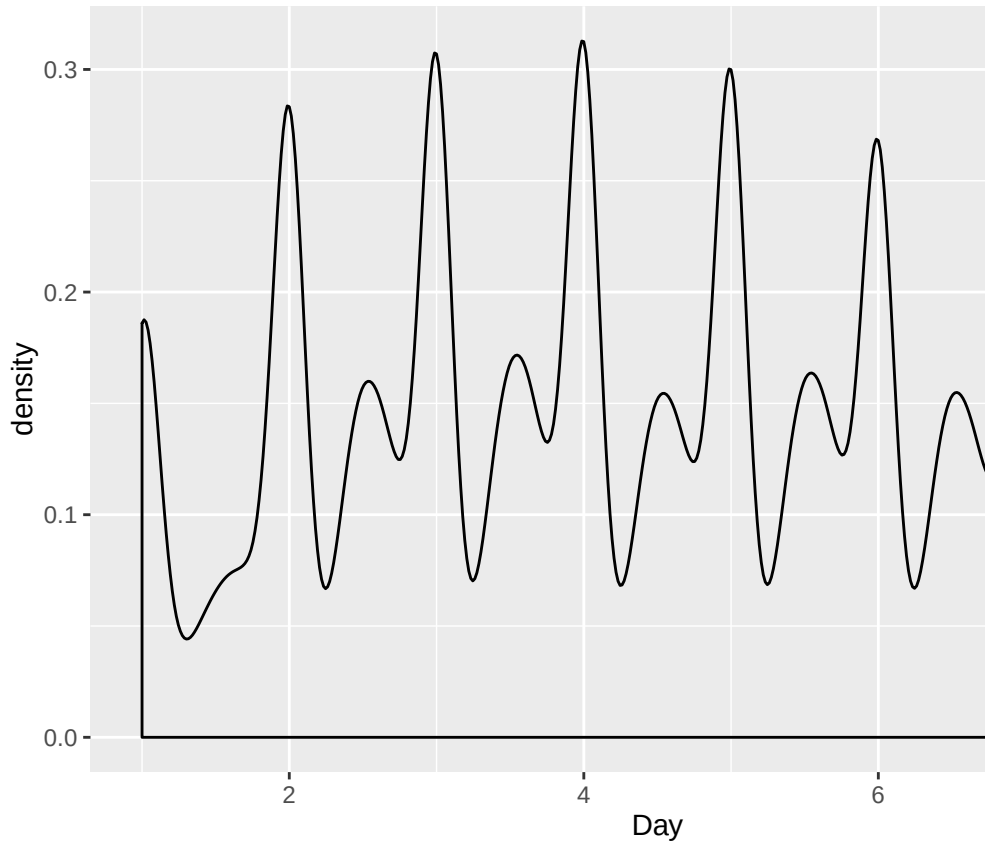
Again, no problem reading in a million rows.

It's also helpful to see how large is the object being created.

Sixteen million bytes is only about 0.1% of the fast memory on my computer. The whole data set, with 13 million cases will take up only about 1-2% of the memory. No problem!

Having established this, the question is whether the analysis of all 13 million cases can be done in a reasonable amount of time. Let's try tabulating the complaint types in the 1-million case data table:

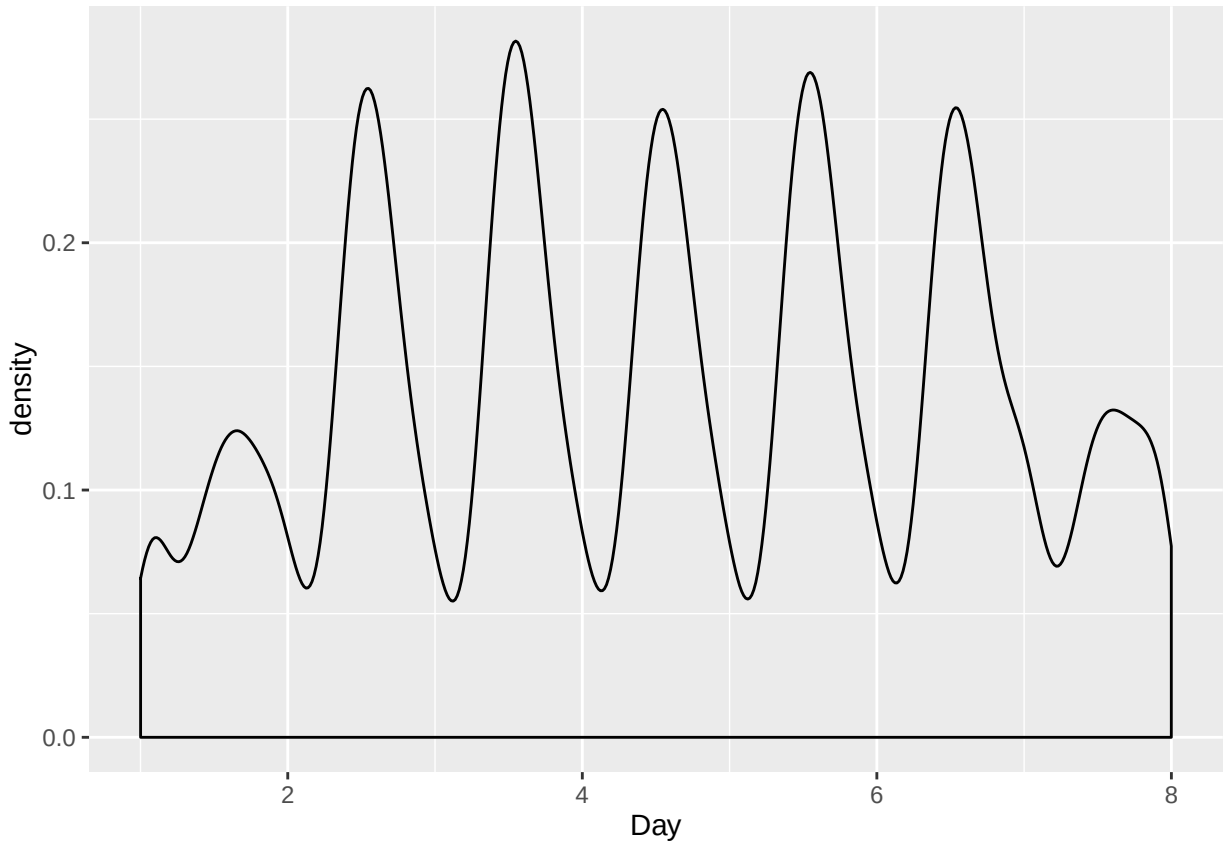
Or, to see when the complaints occur:



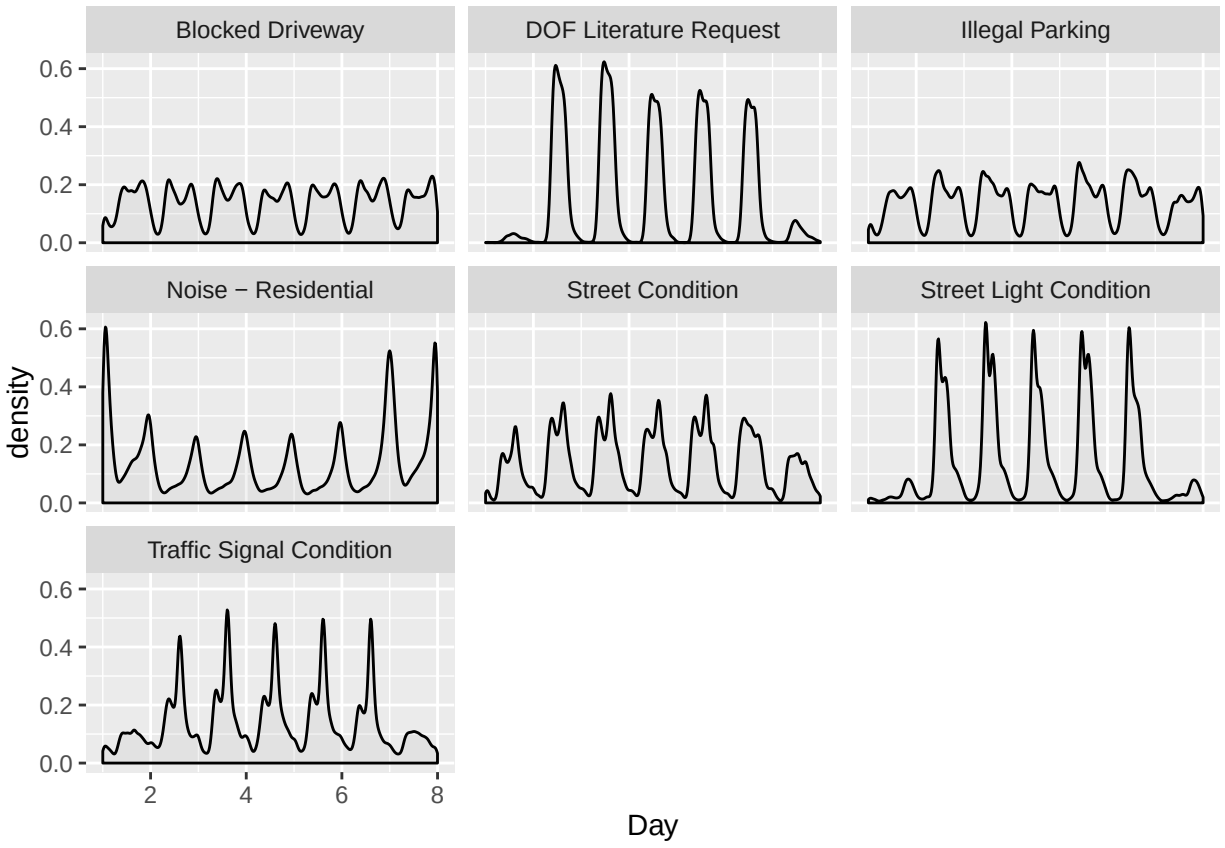
Or, graphing out when the events occur:

There's a peak near the start of each day. Let's filter out the events that are recorded just at midnight:

About 60% of the complaints have time stamps other than midnight.



Let's look at the categories with the largest number of complaints. What's a useful way of displaying how the patterns vary with time.


Blocked Driveway
DOF Literature Request
Illegal Parking
Noise − Residential
Street Condition
Street Light Condition
Traffic Signal Condition
density
0.0
0.2
0.4
0.6
2
4
6
8
Day

Chapter 39

Regex examples.

Here are some examples of patterns in names and the use of a regular expression to detect them. We'll work with the baby names data, summarised to give the total count of each name for each sex.

The regular expression is the string in quotes. `grepl()` is a function that compares a regular expression to a string, returning TRUE if there's a match, FALSE otherwise.

1. The name contains "shine", as in "sunshine" or "moonshine"
2. The name contains three or more vowels in a row.
3. The name contains three or more consonants in a row.
4. The name contains "mn"
5. The first, third, and fifth letters are consonants.
6. How often do boys' and girls' names end in vowels?

Girls' names are almost five times as likely to end in vowels as boys' names.

7. What are the most common end vowels for names?

To answer this question, you have to extract the last vowel from the name. The `extractMatches()` transformation function can do this.

Credit: Adam Lucas

Chapter 40

Extracting matching parts of regular expressions

The `extractMatches()` data verb from the `DataComputing` package helps you to identify the specific characters that match in a regular expression.

Details:

1. Like all data verbs, `extractMatches()` takes a data table as input and produces a data table as output. That data table will have one or more additional columns with the extractions
2. syntax is `Data_table %>% extractMatches(regex, var,...)`. The `var` is the variable to be used in matching with the regex. The `...` provides a way for you to name the extracted segments.
3. To indicate that you want to extract the content that matches part of the regexp, wrap that part in parentheses.
4. When there is no match `extractMatches()` returns `NA`.

Example 1: This regex looks for the letters "n" or "r" followed by a vowel. Both the specific letter and the specific vowel are extracted.

Example 2: What are the most common end vowels for Bible names?

To answer this question, you have to extract the last vowel from the name. The `extractMatches()` transformation function can do this.

Example 2: What Bible Names start and end with a vowel, and what are those vowels? In this example, we'll arrange for the first match to be called `beg_vowel` and the second to be called `end_vowel`.

Chapter 41

Tasks for you

1. Which Bible names in `BibleNames` have the word "man" in the meaning (not first or last word)?
2. Grab everything but the last letter of `names` in `BabyNames` that end with a vowel (example: Grab Ev from Eva)
3. What will this regex extract?

Chapter 42

Some resources

- Data scraping: https://github.com/ropensci/user2016-tutorial
- Text analysis of tweets: http://varianceexplained.org/r/trump-tweets/

Bibliography