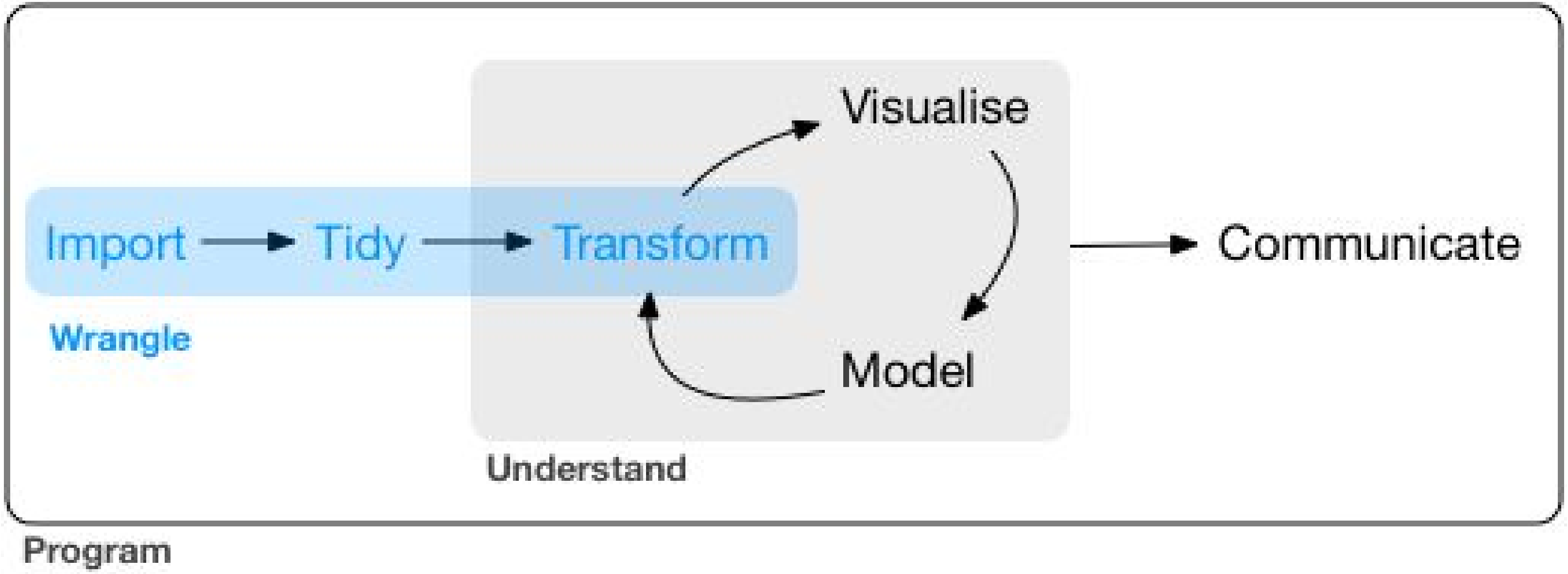
R for Data Science

Chapters 9 - 16

II WRANGLE

Chapter 9 Introduction

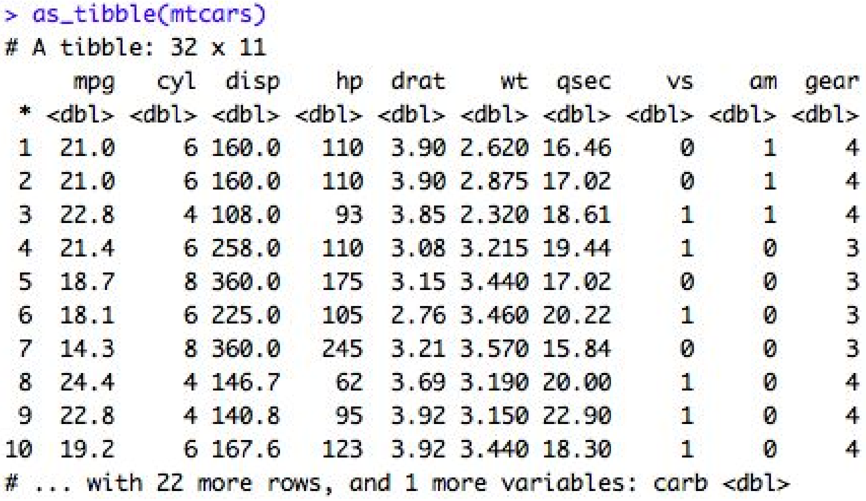

Import
Tidy
Transform
Visualise
Model
Communicate
Wrangle
Understand
Program

Chapter 10 Tibbles

Tibble
(aka fancy data frame)

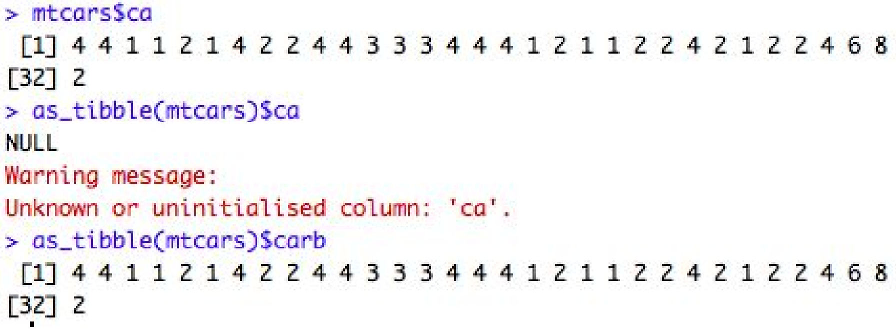
Main Differences

printing



```
> as_tibble(mtcars)
# A tibble: 32 x 11
     mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear
 * <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1  21.0     6 160.0   110  3.90 2.620 16.46     0     1     4
 2  21.0     6 160.0   110  3.90 2.875 17.02     0     1     4
 3  22.8     4 108.0    93  3.85 2.320 18.61     1     1     4
 4  21.4     6 258.0   110  3.08 3.215 19.44     1     0     3
 5  18.7     8 360.0   175  3.15 3.440 17.02     0     0     3
 6  18.1     6 225.0   105  2.76 3.460 20.22     1     0     3
 7  14.3     8 360.0   245  3.21 3.570 15.84     0     0     3
 8  24.4     4 146.7    62  3.69 3.190 20.00     1     0     4
 9  22.8     4 140.8    95  3.92 3.150 22.90     1     0     4
10  19.2     6 167.6   123  3.92 3.440 18.30     1     0     4
# ... with 22 more rows, and 1 more variables: carb <dbl>
```

subsetting



```
> mtcars$ca
 [1] 4 4 1 1 2 1 4 2 2 4 4 3 3 3 4 4 4 1 2 1 1 2 2 4 2 1 2 2 4 6 8
[32] 2
> as_tibble(mtcars)$ca
NULL
Warning message:
Unknown or uninitialised column: 'ca'.
> as_tibble(mtcars)$carb
 [1] 4 4 1 1 2 1 4 2 2 4 4 3 3 3 4 4 4 1 2 1 1 2 2 4 2 1 2 2 4 6 8
[32] 2
```

Chapter 11 Data import

`readr`

`read_csv` vs `read.csv`

1. `read_csv` ~10x faster (also see `data.table::fread()`)

2. `read_csv` makes a tibble

3. More reproducible

parse_*()

```
parse_logical(c("TRUE", "FALSE", "NA"))

parse_number("It cost $123.45")

parse_character("El Ni\xf1o was particularly bad
this year",        locale = locale(encoding =
"Latin1"))

parse_datetime("20101010")
```

Chapter 12 Tidy data

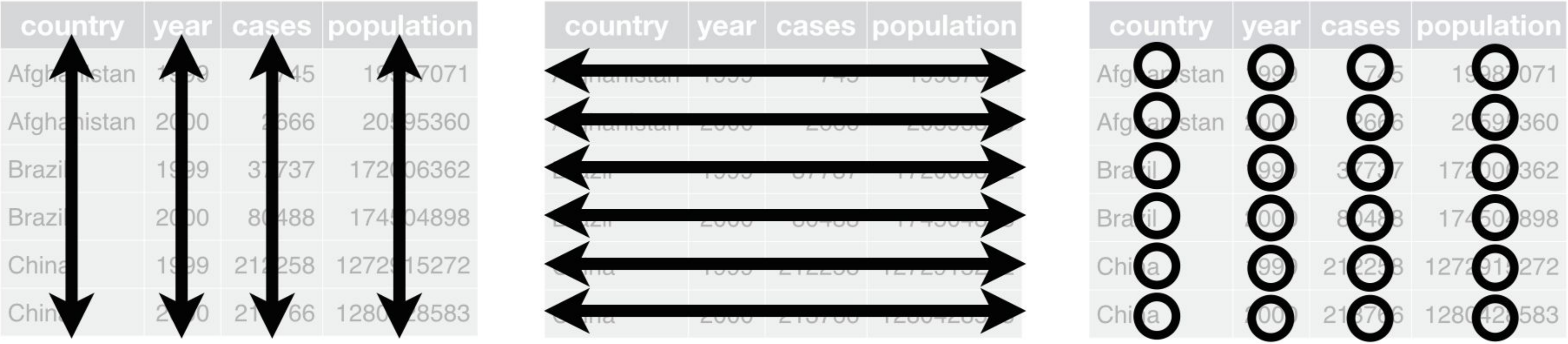
tidyr

1. Each variable must have its own column.

2. Each observation must have its own row.

3. Each value must have its own cell.

tidyr



country	year	cases	population
Afghanistan	1999	745	19987071
Afghanistan	2000	2666	20595360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	213766	1280428583

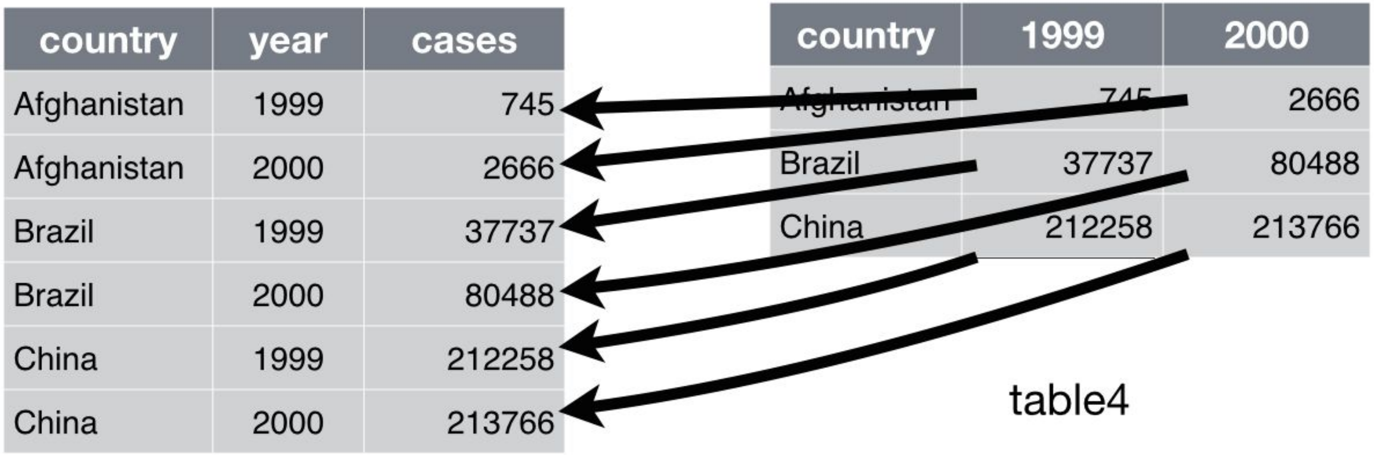
variables

observations

values

tidyr

`gather()` - turn multiple columns into 2 columns



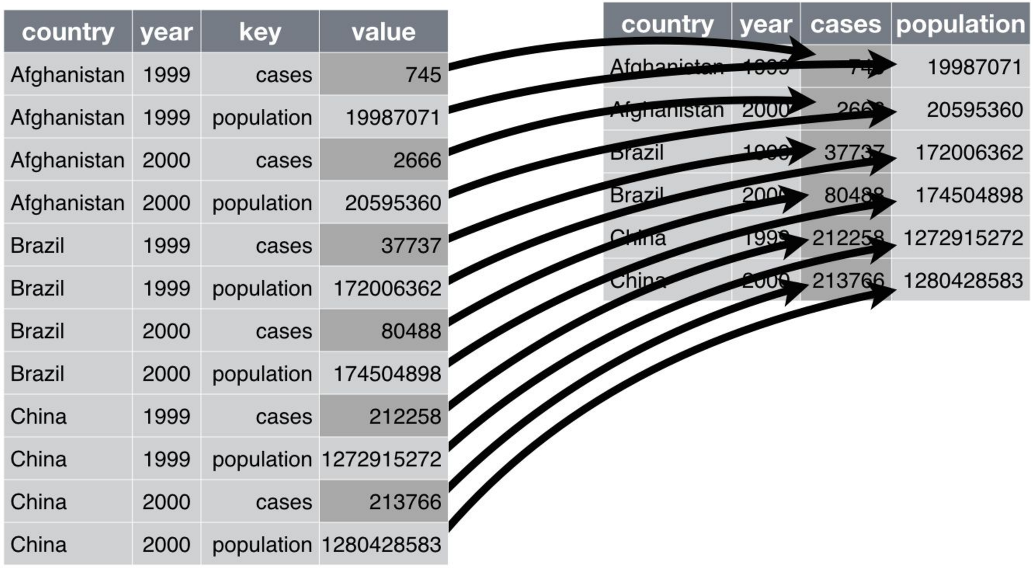
country	year	cases
Afghanistan	1999	745
Afghanistan	2000	2666
Brazil	1999	37737
Brazil	2000	80488
China	1999	212258
China	2000	213766

country	1999	2000
Afghanistan	745	2666
Brazil	37737	80488
China	212258	213766

table4

tidyr

`spread()` - turn 2 columns into multiple columns



country	year	key	value
Afghanistan	1999	cases	745
Afghanistan	1999	population	19987071
Afghanistan	2000	cases	2666
Afghanistan	2000	population	20595360
Brazil	1999	cases	37737
Brazil	1999	population	172006362
Brazil	2000	cases	80488
Brazil	2000	population	174504898
China	1999	cases	212258
China	1999	population	1272915272
China	2000	cases	213766
China	2000	population	1280428583

table2

country	year	cases	population
Afghanistan	1999	745	19987071
Afghanistan	2000	2666	20595360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	213766	1280428583

tidyr

separate() - break one column into multiple

unite() - combine multiple columns into one

complete() - fill in missing factorial

fill() - fill in missing values based on most recent non-NA

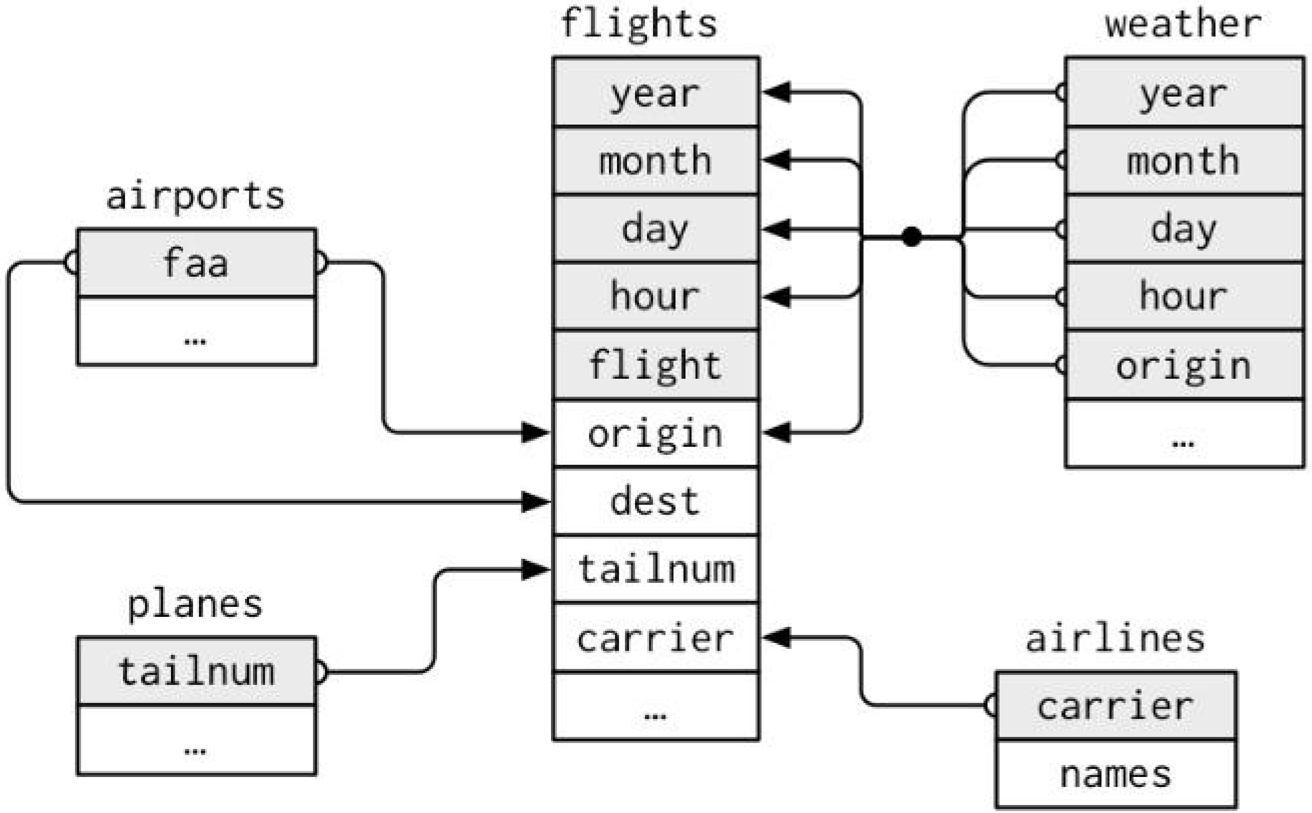
Chapter 13 Relational data

Verbs to work with relational data

Mutating joins: add new variables to one data frame from matching observations in another

Filtering joins: filter observations from one data frame based on whether or not they match an observation in the other table

Set operations: treat observations as if they were set elements


flights
year
month
day
hour
flight
origin
dest
tailnum
carrier
...
weather
year
month
day
hour
origin
...
airports
faa
...
planes
tailnum
...
airlines
carrier
names

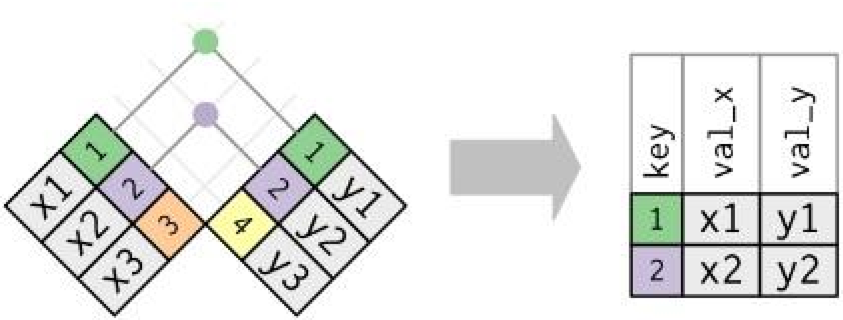
Keys

Primary key: Identifies an observation in its own table

Foreign key: Identifies an observation in another table

Mutating Join

Matching observations by keys to copy variables from one table to another



key	val_x	val_y
1	x1	y1
2	x2	y2

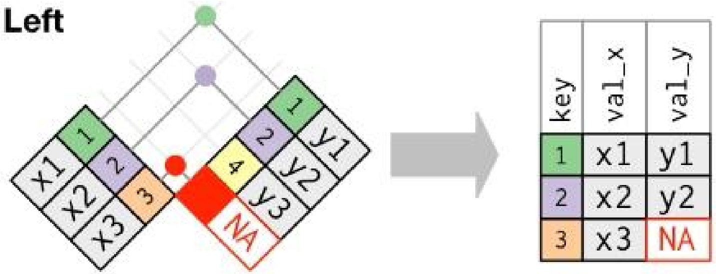
number of dots = number of matches = number of rows

Inner join: matching pairs of observations whenever the keys are equal

Unmatched rows are not included in the result

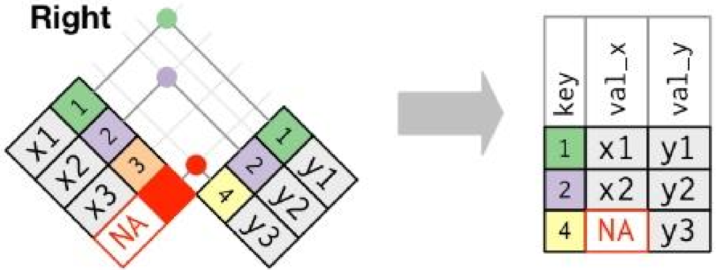
Outer joins: Keeps observations that appear in at least one of the tables

Left join: keeps all observations in x



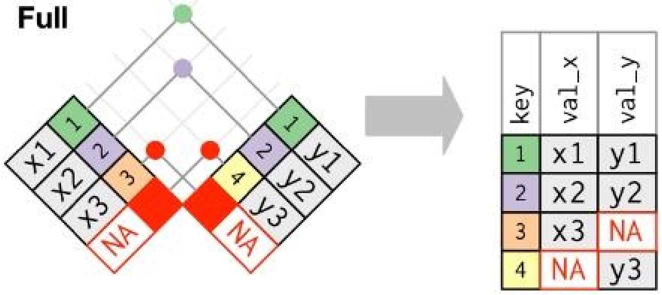
key	val_x	val_y
1	x1	y1
2	x2	y2
3	x3	NA

Right join: keeps all observations in y

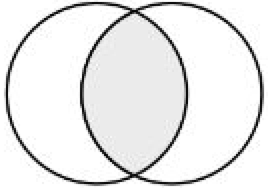


key	val_x	val_y
1	x1	y1
2	x2	y2
4	NA	y3

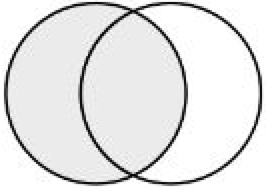
Full join: keeps all observations in x and y



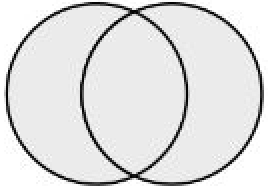
key	val_x	val_y
1	x1	y1
2	x2	y2
3	x3	NA
4	NA	y3



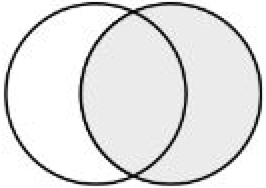
`inner_join(x, y)`



`left_join(x, y)`



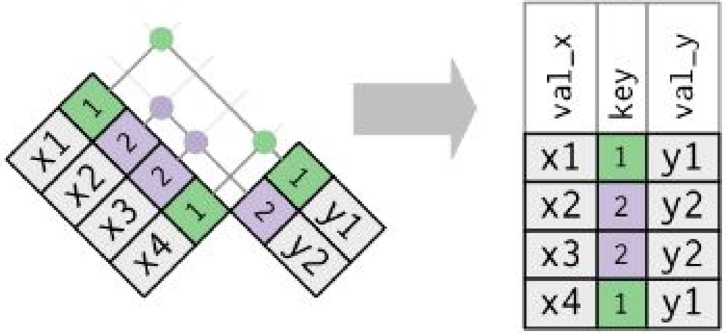
`full_join(x, y)`



`right_join(x, y)`

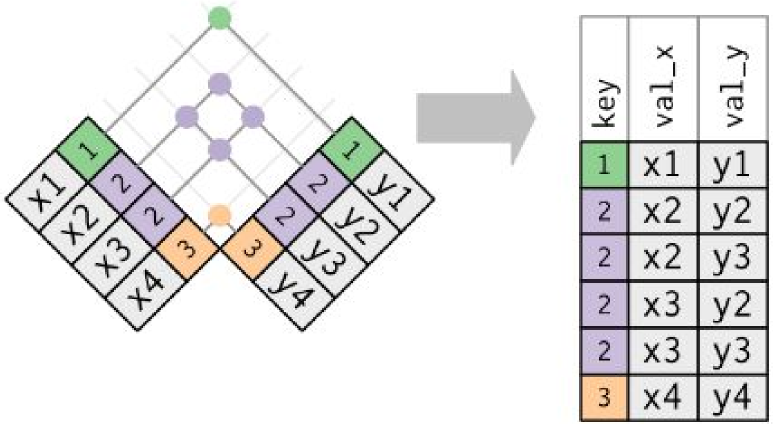
Duplicate Keys

One table with duplicate keys



val_x	key	val_y
x1	1	y1
x2	2	y2
x3	2	y2
x4	1	y1

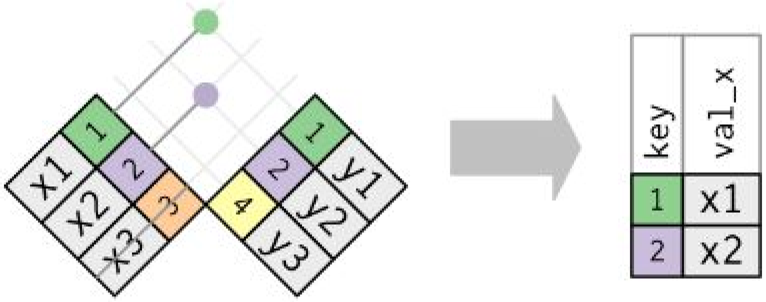
Both tables with duplicate keys



key	val_x	val_y
1	x1	y1
2	x2	y2
2	x2	y3
2	x3	y2
2	x3	y3
3	x4	y4

Filtering Joins

semi_join(x, y) keeps all observations in x that have a match in y



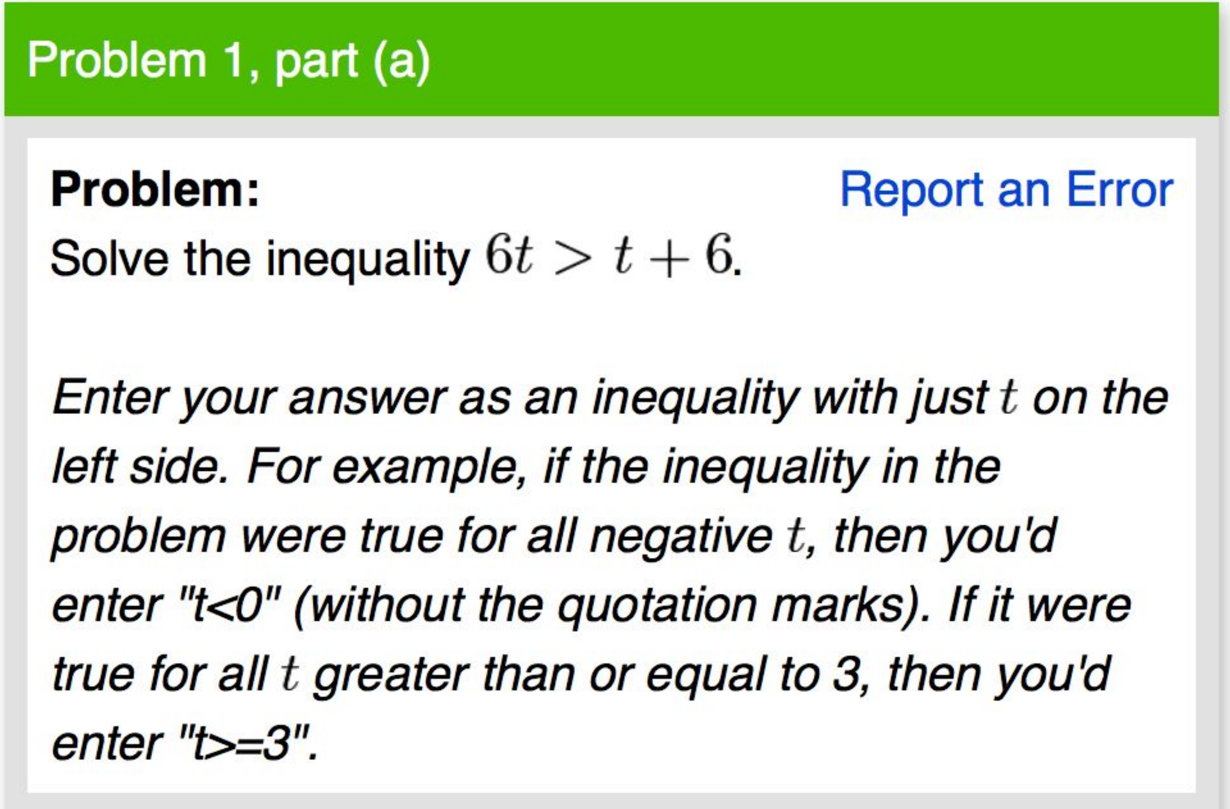
key	val_x
1	x1
2	x2

anti_join(x, y) drops all observations in x that have a match in y

- Good for finding mismatches

Chapter 14 Strings

Problem 1: Strings and Quotation Marks



Problem 1, part (a)

Problem:

Report an Error

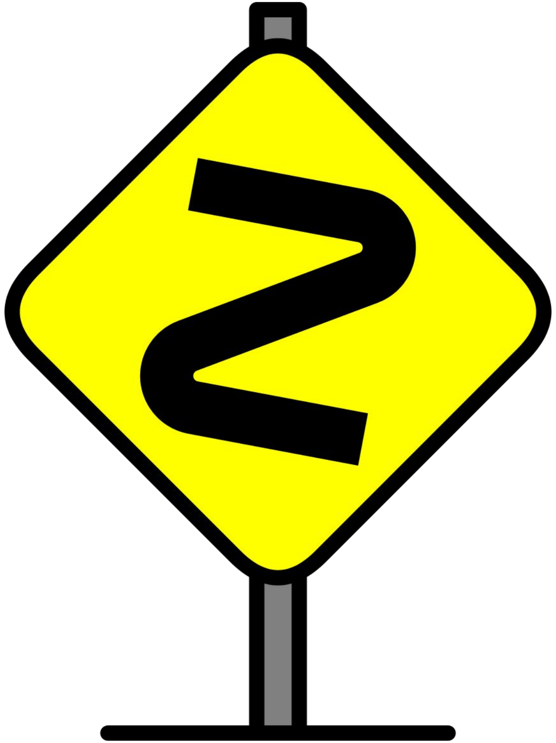
Solve the inequality $6t > t + 6$.

Enter your answer as an inequality with just t on the left side. For example, if the inequality in the problem were true for all negative t, then you'd enter "t<0" (without the quotation marks). If it were true for all t greater than or equal to 3, then you'd enter "t>=3".

Problem 2: Strings mean more to people than they do to computers.

```
Example: str_c(c("A", "B", "C", "D"),
c("up", "down"), sep="!", collapse="*")
```

Problem 3: Regular Expressions (matching)



Chapter 15 Factors

`forcats`

`fct_reorder()` - set order of a factor based on another variable

`fct_relevel()` - move certain levels to front of order

`fct_reorder2()` - like `fct_reorder()` but takes 2 arguments to order by

`fct_infreq()` - order levels in increasing frequency

`forcats`

`fct_recode()` - changing display values of a level

`fct_collapse()` - to group together a lot of levels under one new level

`fct_lump()` - puts together all small levels into a single level

Chapter 16 Dates and times

`lubridate`

Making Date(time) Columns

`ymd()`, `mdy()`, `dmy()`
to specify format

`make_date()`/ `make_datetime()`
from multiple columns

lubridate

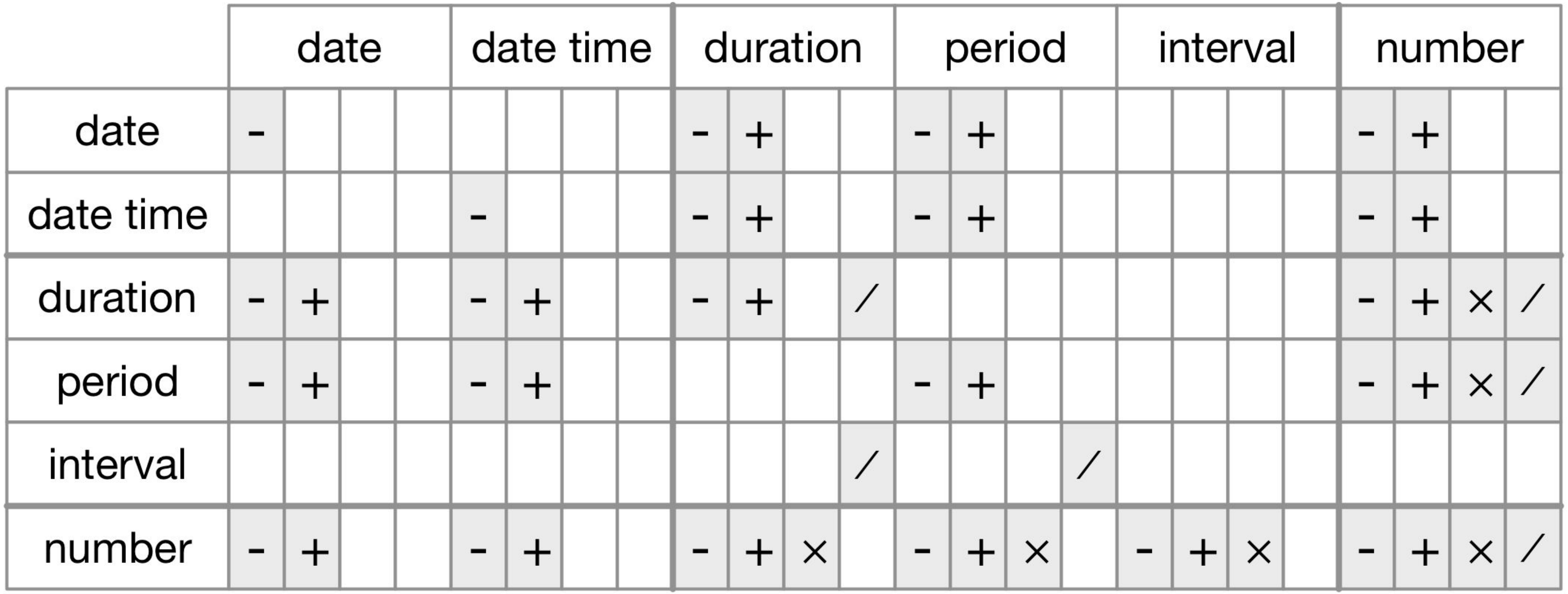
Working with Date(time) Columns

`year(), month()`
to pull out specific information

`as.duration()`
`ddays(), days()`
grab timespans

lubridate

Working with Date(time) Columns



	date				date time				duration				period				interval				number			
date	-								-	+			-	+							-	+		
date time					-				-	+			-	+							-	+		
duration	-	+			-	+			-	+		/									-	+	×	/
period	-	+			-	+							-	+							-	+	×	/
interval												/				/								
number	-	+			-	+			-	+	×		-	+	×		-	+	×		-	+	×	/

lubridate

Timezones

```
x1 <- ymd_hms("2015-06-01 12:00:00", tz = "America/New_York")
x2 <- ymd_hms("2015-06-01 18:00:00", tz = "Europe/Copenhagen")
x3 <- ymd_hms("2015-06-02 04:00:00", tz = "Pacific/Auckland")

x1 - x2
#> Time difference of 0 secs
x1 - x3
#> Time difference of 0 secs
```