# Data Import and Tidying

Samuel Lippl

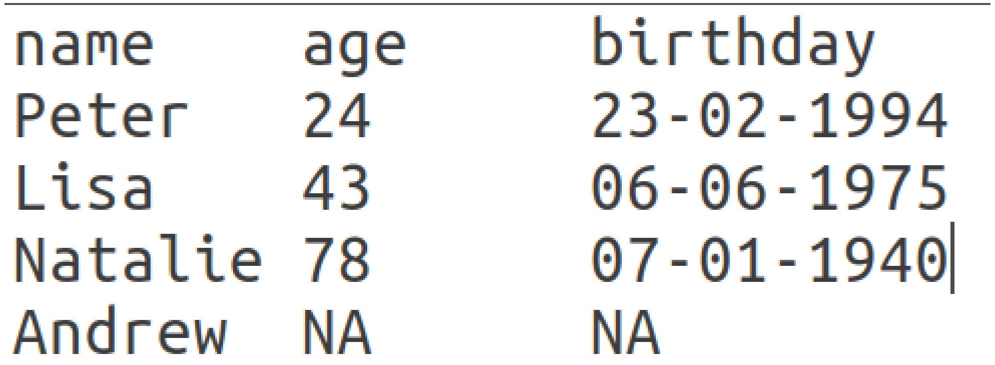
07 November 2018

# Agenda

- Data import
- Principles of R
- Data tidying
- Join tables

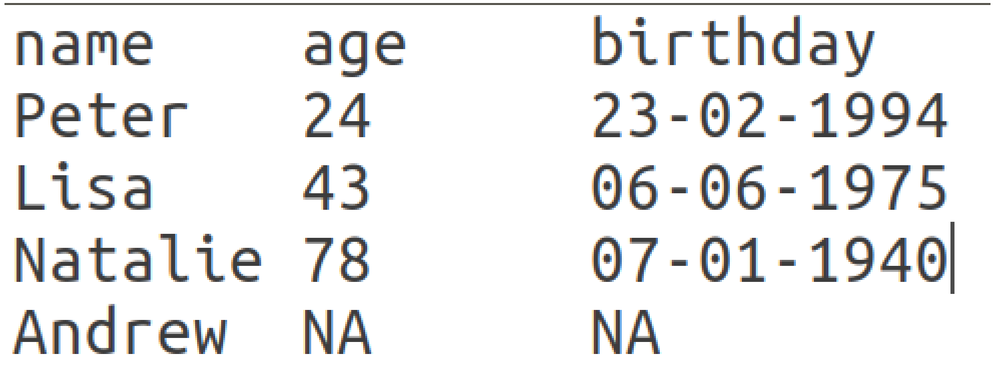
# Data import

# Sharing data

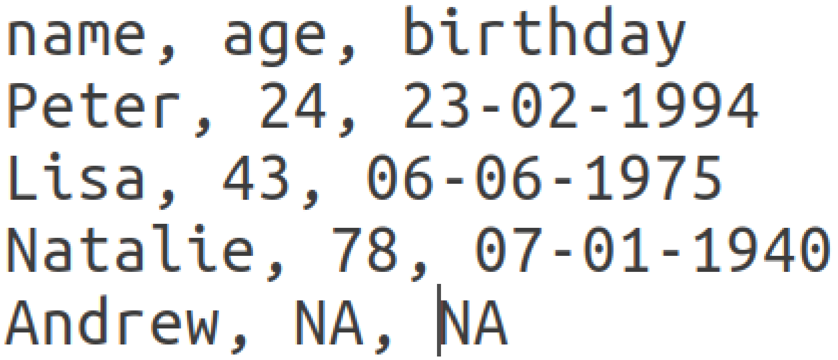


```
name     age       birthday
Peter    24        23-02-1994
Lisa     43        06-06-1975
Natalie  78        07-01-1940
Andrew   NA        NA
```

# Sharing data



```
name     age      birthday
Peter    24       23-02-1994
Lisa     43       06-06-1975
Natalie  78       07-01-1940
Andrew   NA       NA
```



```
name, age, birthday
Peter, 24, 23-02-1994
Lisa, 43, 06-06-1975
Natalie, 78, 07-01-1940
Andrew, NA, NA
```

# Exercise 1: Learn how to read in delimited files

1. Read sections 11.1 and 11.2
2. Complete the exercises in 11.2.2 in [r4ds.had.co.nz]

# Parsing files

- `read_csv` takes the first 1000 lines of a table to guess the column type.

```
read_csv("a\na")
```

```
## # A tibble: 1 x 1
##   a
##   <chr>
## 1 a
```

# Parsing files

- `read_csv` takes the first 1000 lines of a table to guess the column type.

```r
read_csv("a\n1")
```

```
## # A tibble: 1 x 1
##       a
##   <int>
## 1     1
```

# Parsing files

- `read_csv` takes the first 1000 lines of a table to guess the column type.

```
read_csv("a\n2010-01-01")
```

```
## # A tibble: 1 x 1
##   a
##   <date>
## 1 2010-01-01
```

# Parsing files: problems

Problems occur if:

- the first 1000 lines are not particularly informative
    - only NA values
    - only integers even though non-integers occur later
- the way in which the values are entered is not entirely clear
    - dates
    - factors

# Other formats

- Binary files
  - feather implements a binary format to share across languages
  - RDS is a format specific to R
- SPSS, SAS, Stata: `haven`
- Matlab, Excel etc.

# Principles of R

# Make common tasks more accessible

```
df <-
  file %>%
  read_delim(delim = ",", na = c("", "NA"),
             skip = 0, ...)
```

becomes …

```
df <-
  file %>%
  read_csv()
```

by:

- special functions
- default values

# Make generalized functions consistent

- `read_delim` has the same interface as `read_csv`, with additional parameters

# Data tidying

# Reminder: Tidy Data

1. Each variable forms a column.
2. Each observation forms a row.
3. Each type of observational unit forms a table.

# Data Tidying

- Data often arrives in an untidy state
- refers to the procedure by which we make this data tidy
- Task of the package `tidyr` (part of the tidyverse)
    - `gather` summarizes column names as a new column
    - `spread` spreads variable levels as new column names

# spread and gather

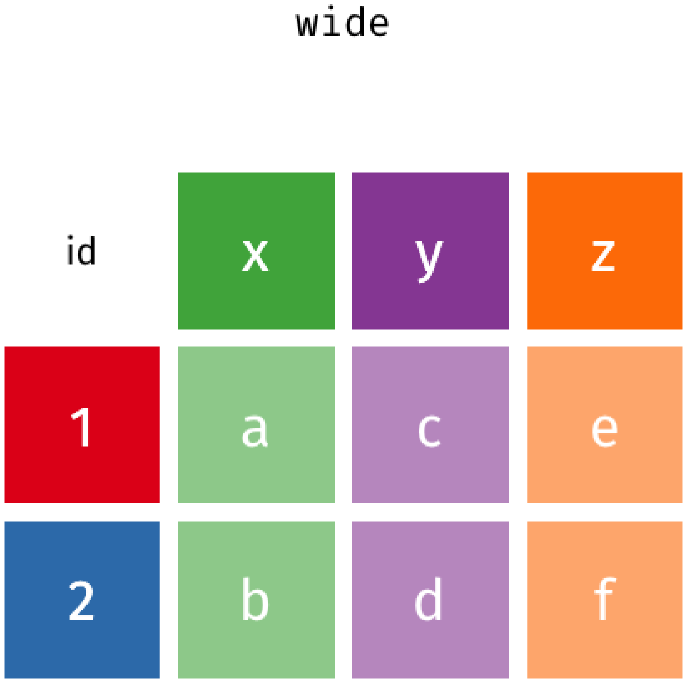
wide

| id | x | y | z |
|---|---|---|---|
| 1 | a | c | e |
| 2 | b | d | f |

long

| id | key | val |
|---|---|---|
| 1 | x | a |
| 2 | x | b |
| 1 | y | c |
| 2 | y | d |
| 1 | z | e |
| 2 | z | f |

# `spread` and `gather`



`wide`

| id | x | y | z |
|---|---|---|---|
| 1 | a | c | e |
| 2 | b | d | f |

# `gather`: example

```
table4a
```

```
## # A tibble: 3 x 3
##   country     `1999` `2000`
## * <chr>        <int>  <int>
## 1 Afghanistan    745   2666
## 2 Brazil       37737  80488
## 3 China       212258 213766
```

# Exercise 2

Fill in the corresponding lines in the code to tidy the table

```
table4a %>%
  gather(
    # Specify the name of the key, i. e. the variable which has the
    # different columns as values
    key = ,
    # What name should the column have in which the former cell values
    # are notated
    value = ,
    # Specify the columns which are variable levels
    ...)
```

# `spread`: example

```
table2
```

```
## # A tibble: 12 x 4
##    country      year type            count
##    <chr>       <int> <chr>           <int>
##  1 Afghanistan  1999 cases             745
##  2 Afghanistan  1999 population   19987071
##  3 Afghanistan  2000 cases            2666
##  4 Afghanistan  2000 population   20595360
##  5 Brazil       1999 cases           37737
##  6 Brazil       1999 population  172006362
##  7 Brazil       2000 cases           80488
##  8 Brazil       2000 population  174504898
##  9 China        1999 cases          212258
## 10 China        1999 population 1272915272
## 11 China        2000 cases          213766
## 12 China        2000 population 1280428583
```

# Exercise 3

Fill in the corresponding lines in the code to tidy the table

```
table2 %>%
  spread(
    # the variable which gives the new column names
    key = ,
    # the variable which gives the new column cell values
    value =
  )
```

# Join tables

# Joining two data frames

- combine information from two data frames
- covered by the `*_join` functions in `dplyr`
- join      certain columns

# Joining two data frames

- combine information from two data frames
- covered by the `*_join` functions in `dplyr`
- join certain columns

```
table1 %>%
  *_join(table2, by = "column(s)")
```

# Joins: example

```
band_instruments
```

```
## # A tibble: 3 x 2
##   name  plays 
##   <chr> <chr> 
## 1 John  guitar
## 2 Paul  bass  
## 3 Keith guitar
```

# Joins: example

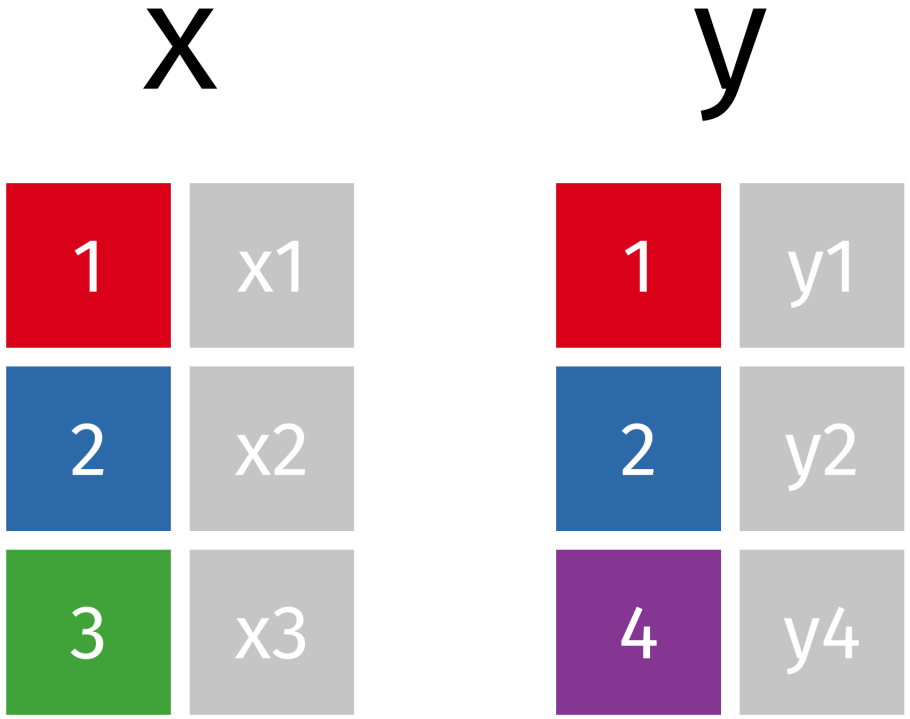
```
band_members
```

```
## # A tibble: 3 x 2
##   name  band
##   <chr> <chr>
## 1 Mick  Stones
## 2 John  Beatles
## 3 Paul  Beatles
```

# Left, right, inner, full

- generate a table with columns from both tables
- differ in the observations which they keep

# Joining tables: animations



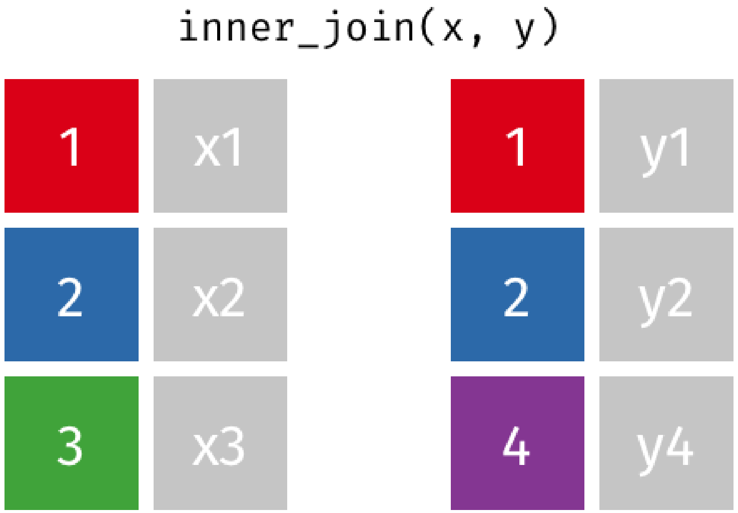
x

| | |
|---|---|
| 1 | x1 |
| 2 | x2 |
| 3 | x3 |

y

| | |
|---|---|
| 1 | y1 |
| 2 | y2 |
| 4 | y4 |

# inner_join

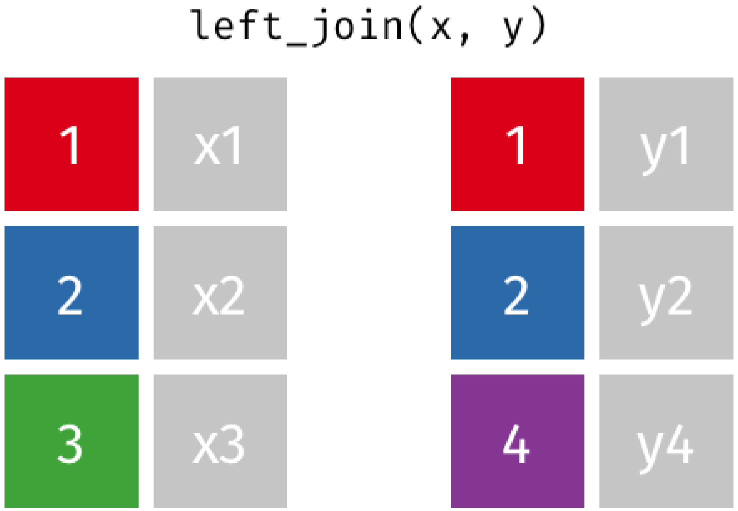


inner_join(x, y)

| | |
|---|---|
| 1 | x1 |
| 2 | x2 |
| 3 | x3 |

| | |
|---|---|
| 1 | y1 |
| 2 | y2 |
| 4 | y4 |

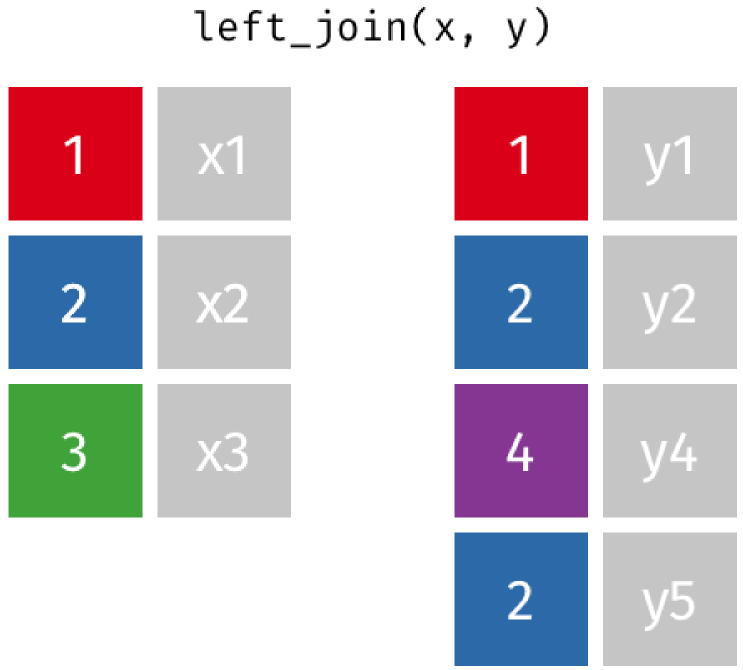
# `left_join`



| | |
|---|---|
| 1 | x1 |
| 2 | x2 |
| 3 | x3 |

| | |
|---|---|
| 1 | y1 |
| 2 | y2 |
| 4 | y4 |

# `left_join`

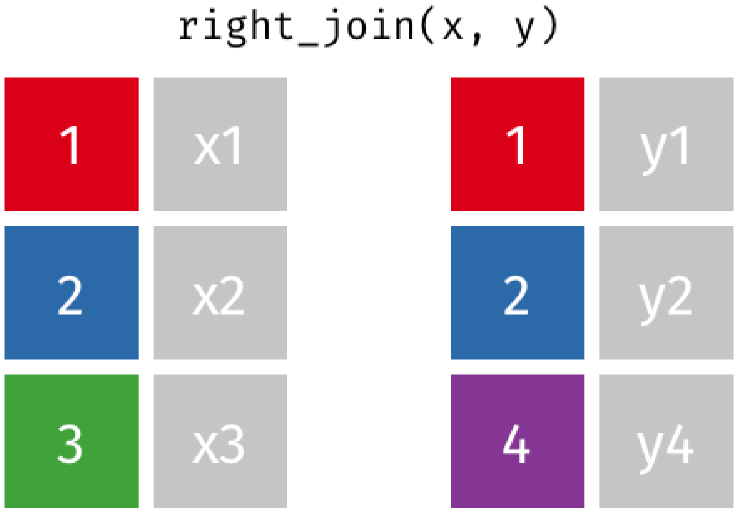


```
left_join(x, y)
```

| | |
|---|---|
| 1 | x1 |
| 2 | x2 |
| 3 | x3 |

| | |
|---|---|
| 1 | y1 |
| 2 | y2 |
| 4 | y4 |
| 2 | y5 |

# right_join

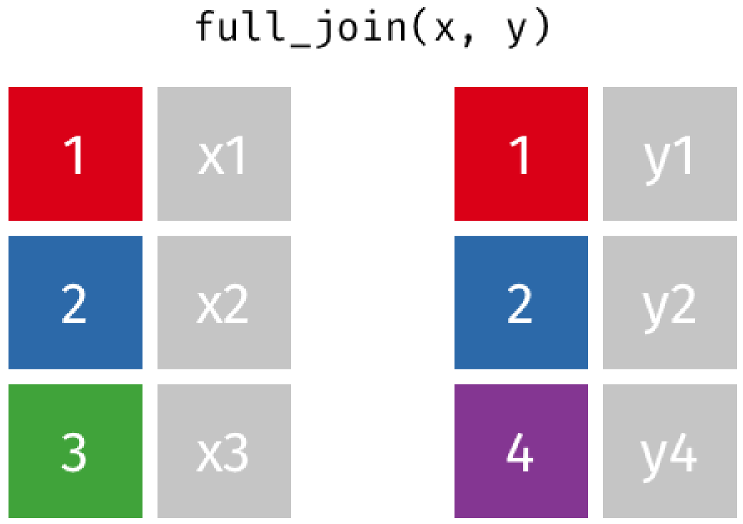


`right_join(x, y)`

| | |
|---|---|
| 1 | x1 |
| 2 | x2 |
| 3 | x3 |

| | |
|---|---|
| 1 | y1 |
| 2 | y2 |
| 4 | y4 |

# full_join



```
full_join(x, y)
```

| | |
|---|---|
| 1 | x1 |
| 2 | x2 |
| 3 | x3 |

| | |
|---|---|
| 1 | y1 |
| 2 | y2 |
| 4 | y4 |

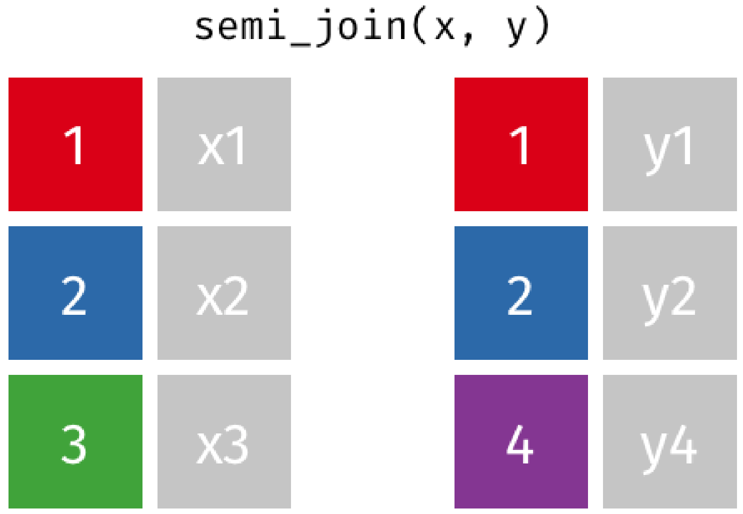
# Exercise 4

1. Generate a table with all band members from `band_members` and the instruments which they are playing.
2. Generate a table which gives you information all players where you know in which band they are playing which instrument

# Filtering joins

- filter `table1`
- `semi_join` keeps the observation which can be found in `table2`
- `anti_join` keeps all other observations

# semi_join

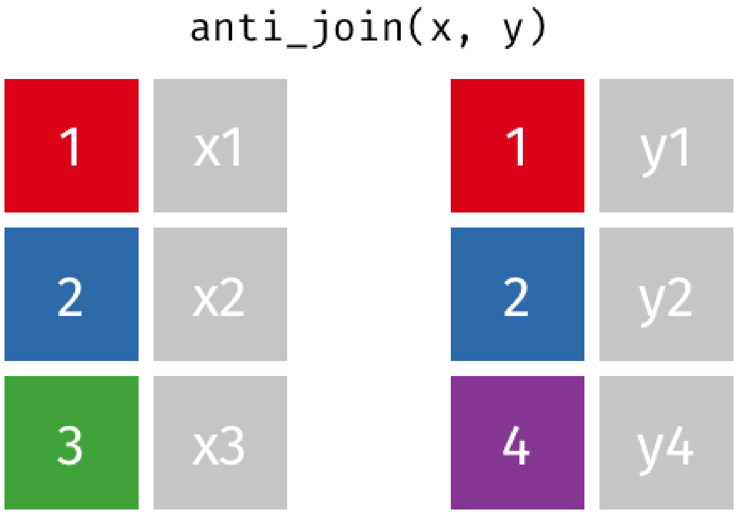


```
semi_join(x, y)
```

| | |
|---|---|
| 1 | x1 |
| 2 | x2 |
| 3 | x3 |

| | |
|---|---|
| 1 | y1 |
| 2 | y2 |
| 4 | y4 |

# anti_join



```
anti_join(x, y)
```

| | |
|---|---|
| 1 | x1 |
| 2 | x2 |
| 3 | x3 |

| | |
|---|---|
| 1 | y1 |
| 2 | y2 |
| 4 | y4 |

# Exercise 5

1. Generate a table with all band members whose instruments is not known

# Further reading

- R4DS, ch. 11-13
- Data transformation cheat sheet
- Data import cheat sheet (includes `tidyr` functions)
- Animated explanations of R functions