Tidy data

Peter Ralph

13 October – Advanced Biological Statistics

Tidy data

Rules of thumb for data tidiness

- Store a copy of data in a nonproprietary format, such as plain ASCII text
- Leave an uncorrected file when doing analyses
- Use descriptive names for your data files and variables
- Include a header line with descriptive variable names
- Maintain effective metadata about the data (a README)
- Add new observations to a dataset by *row*
- Add new variables to a dataset by *column*
- A column of data should contain only one data type
- All measurements of the same type should be in the same column

images of lab notebooks pasted into an Excel document

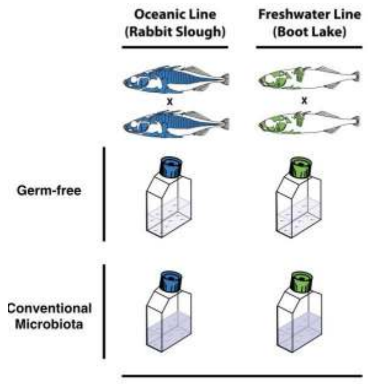
Number of eggs laid by some chickens

breed	1	2	3	4	5	6
rhode island red	5	6	NA	NA	NA	NA
white leghorn	7	5	6	8	NA	NA
barred rock	3	2	4	4	3	4
jersey giant	5	2	8	NA	NA	NA
australorp	4	NA	NA	NA	NA	NA

	breed	num_eggs
11	rhode island red	5
21	rhode island red	6
12	white leghorn	7
22	white leghorn	5
32	white leghorn	6
42	white leghorn	8
13	barred rock	3
23	barred rock	2
33	barred rock	4
43	barred rock	4
53	barred rock	3
63	barred rock	4
14	jersey giant	5
24	jersey giant	2
34	jersey giant	8
15	australorp	4

Exercise

Design a tidy data format for the stickleback experiment:



Tools for tidy data

Tidying data is *hard*!

Tools for tidy data

Tidying data is *hard*!

... and often requires expert input.

Tools for tidy data

Tidying data is *hard*!

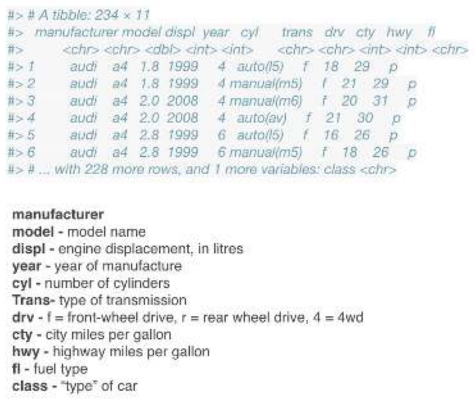
… and often requires expert input.

Many common *data wrangling* operations are made easier by the tidyverse.

The “tidyverse”

- packages that do many of the same things as base functions in R
- designed to do them more “cleanly”
- also includes `ggplot` (for “Grammar of Graphics”)

A tibble is a data frame

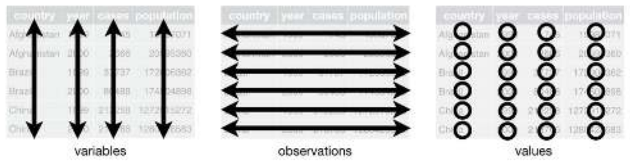


```
#> # A tibble: 234 × 11
#>   manufacturer model displ  year   cyl      trans   drv   cty   hwy    fl
#>          <chr> <chr> <dbl> <int> <int>      <chr> <chr> <int> <int> <chr>
#> 1         audi    a4   1.8  1999     4   auto(l5)     f    18    29     p
#> 2         audi    a4   1.8  1999     4 manual(m5)     f    21    29     p
#> 3         audi    a4   2.0  2008     4 manual(m6)     f    20    31     p
#> 4         audi    a4   2.0  2008     4   auto(av)     f    21    30     p
#> 5         audi    a4   2.8  1999     6   auto(l5)     f    16    26     p
#> 6         audi    a4   2.8  1999     6 manual(m5)     f    18    26     p
#> # ... with 228 more rows, and 1 more variables: class <chr>
```

manufacturer
model - model name
displ - engine displacement, in litres
year - year of manufacture
cyl - number of cylinders
Trans- type of transmission
drv - f = front-wheel drive, r = rear wheel drive, 4 = 4wd
cty - city miles per gallon
hwy - highway miles per gallon
fl - fuel type
class - "type" of car

A tibble is a data frame

```
#> # A tibble: 234 × 11
#>   manufacturer model displ  year   cyl     trans  drv   cty   hwy    fl
#>          <chr> <chr> <dbl> <int> <int>     <chr> <chr> <int> <int> <chr>
#> 1         audi    a4   1.8  1999     4   auto(l5)     f    18    29     p
#> 2         audi    a4   1.8  1999     4 manual(m5)     f    21    29     p
#> 3         audi    a4   2.0  2008     4 manual(m6)     f    20    31     p
#> 4         audi    a4   2.0  2008     4   auto(av)     f    21    30     p
#> 5         audi    a4   2.8  1999     6   auto(l5)     f    16    26     p
#> 6         audi    a4   2.8  1999     6 manual(m5)     f    18    26     p
#> # ... with 228 more rows, and 1 more variables: class <chr>
```



Key functions in dplyr

- Pick observations by their values with `filter()`.
- Reorder the rows with `arrange()`.
- Pick variables by their names with `select()`.
- Create new variables with functions of existing variables with `mutate()`.
- Collapse many values down to a single summary with `summarise()`.

`filter()`, `arrange()` and `select()`

```
a1 <- select(airbnb, neighbourhood, price, host_id, beds, bathrooms)
```

```
a2 <- filter(a1, neighbourhood == "Richmond"
                 | neighbourhood == "Woodlawn"
                 | neighbourhood == "Downtown")
```

```
a3 <- arrange(a2, price, neighbourhood)
```

`mutate()` and `transmutate()`

Add new variables:

```
mutate(a3,
    price_per_bed = price / beds,
    price_per_bath = price / bathrooms)
```

Or, make an entirely new data frame:

```
transmute(airbnb,
    price = price,
    price_per_bed = price / beds,
    price_per_bath = price / bathrooms)
```

`group_by()` and `summarize()`

`group_by()` aggregates data by category, e.g.:

```
by_hood <- group_by(a3, neighbourhood)
```

Now, you can calculate *summaries* of other variables *within* each group, e.g.:

```
summarise(by_hood, price = mean(price, na.rm = TRUE))
```

Your turn

1. Make a data frame only including rooms in the top ten neighbourhoods. Then, using only these neighbourhoods…
2. Find the mean `price`, `cleaning_fee`, and ratio of cleaning fee to price, by neighbourhood.
3. Edit your code in (2) to add variables for the 25% and 75% quantile of `price` (use `quantile( )`).
4. Do as in (2) and (3) but splitting by both `neighbourhood` and `room_type` (e.g., finding the mean price of private rooms in Woodlawn).
5. Edit your code in (1) to add a new variable giving the number of characters in the `house_rules` (use `nchar( )`).

// reveal.js plugins