data wrangling with dplyr

Andrew Jones | Strategy Unit

Wrangling

Reshaping or transforming data into a format which is easier to work with

(…for later visualisation, modelling, or computing of statistics…)

A note on “tidy” data

Tidyverse functions work best with tidy data:

1. Each variable forms a column.
2. Each observation forms a row.

(Broadly, this means long rather than wide tables)

The tool: dplyr package

(dee-ply-r)

dplyr is a language for data manipulation

Most wrangling puzzles can be solved with knowledge of just 5 dplyr verbs (5 functions).

These verbs will be the subject of this session.

Project 2:

Project 2:
Exploring Mental Health (MH) Inpatient Capacity

Project 2:

We have been asked to conduct an analysis of Mental Health inpatient capacity in England.

As part of this, we will be looking at the changes in the number (and occupancy) of MH beds available in recent years.

Background

Maintaining clinical effectiveness and safety when a ward is fully occupied is a serious challenge for staff.

Inappropriate out of area placements mean individuals are separated from their social networks for the duration of their inpatient care.

The Data:

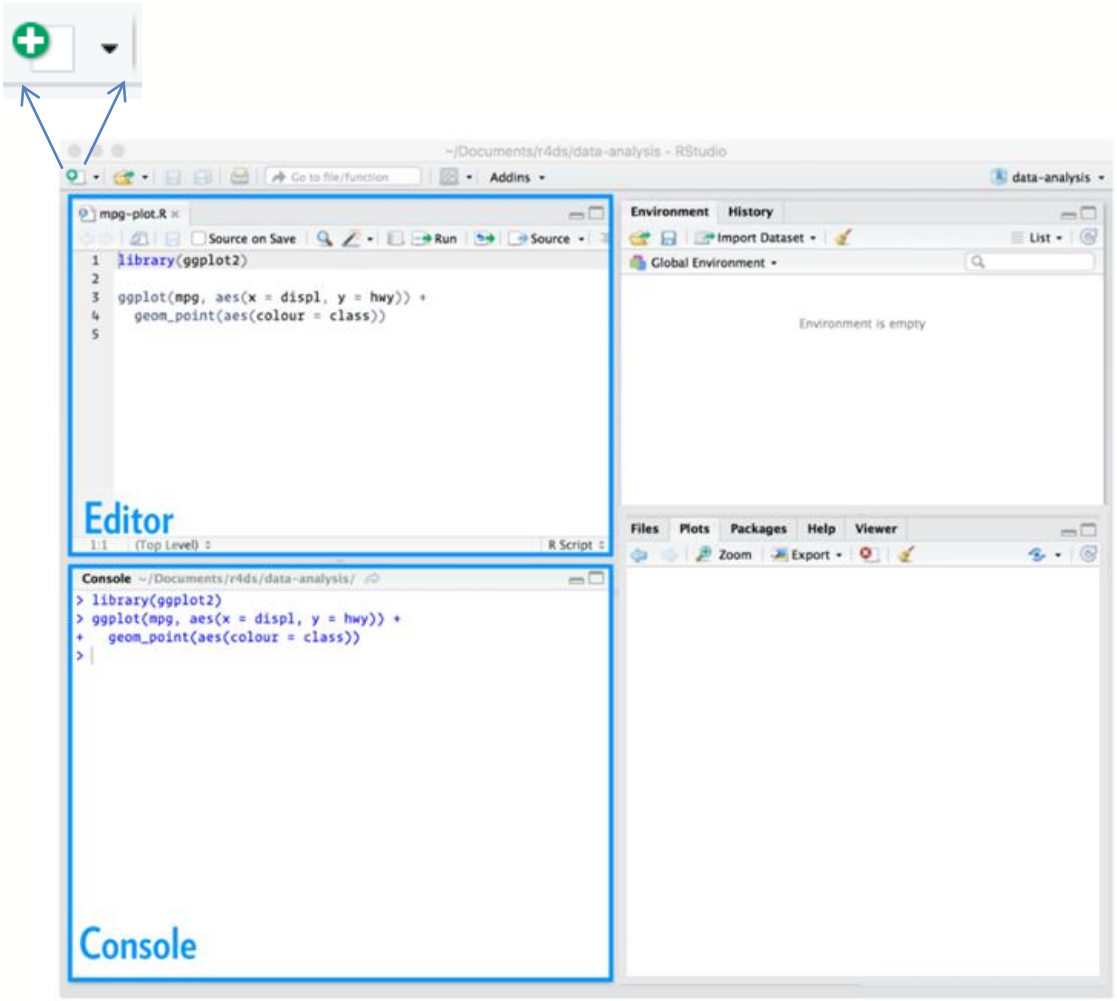
KH03 returns (bed numbers and occupancy) by organisation, published by NHS England.

Scraped from the NHSE statistics website*:

https://www.england.nhs.uk/statistics/statistical-work-areas/bed-availability-and-occupancy/bed-data-overnight/

** and partially cleaned*

Start a new script

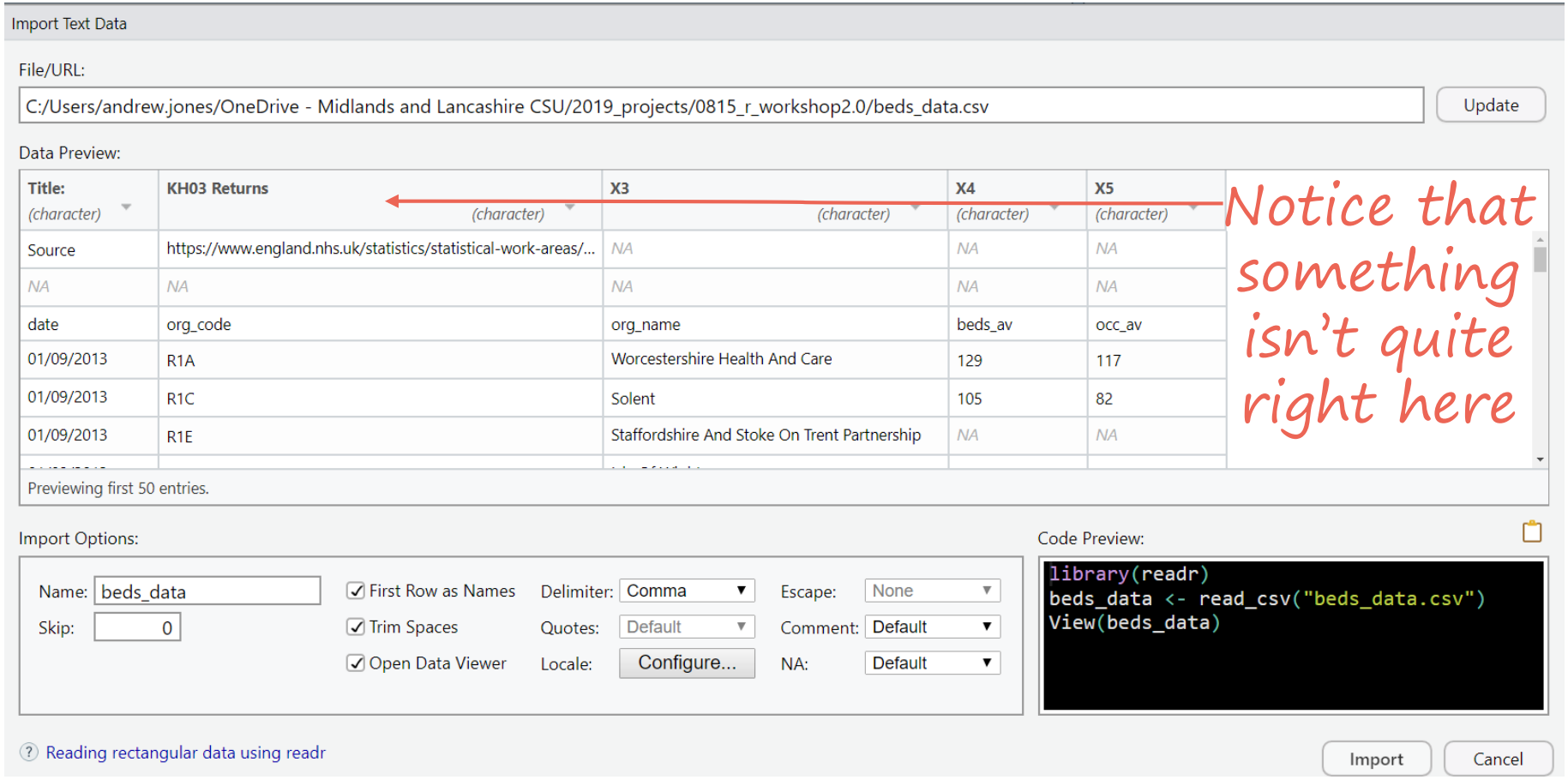


Load the data:

beds_data.csv

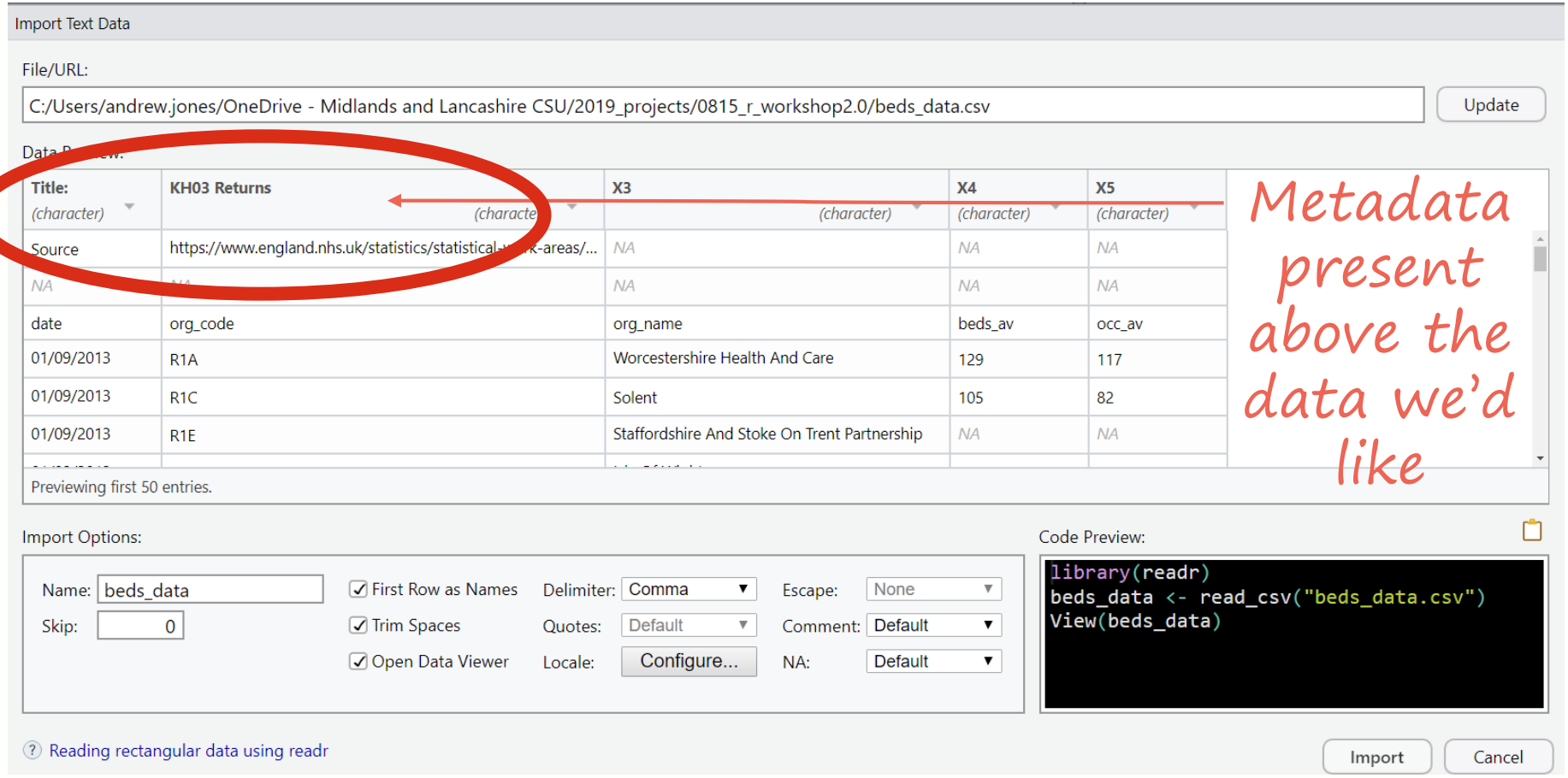
Less friendly csvs

Load beds_data.csv



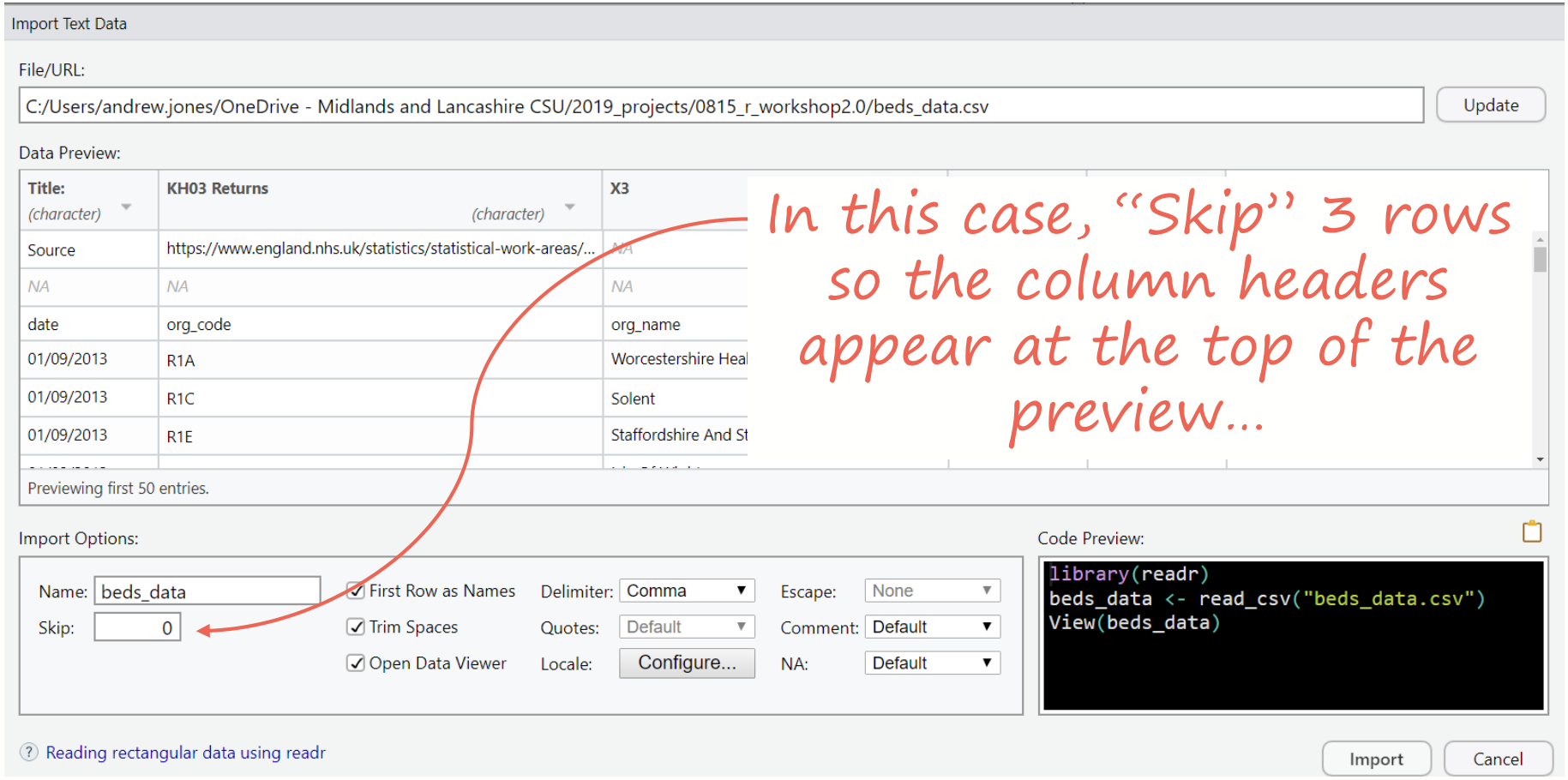
Less friendly csvs

Load beds_data.csv

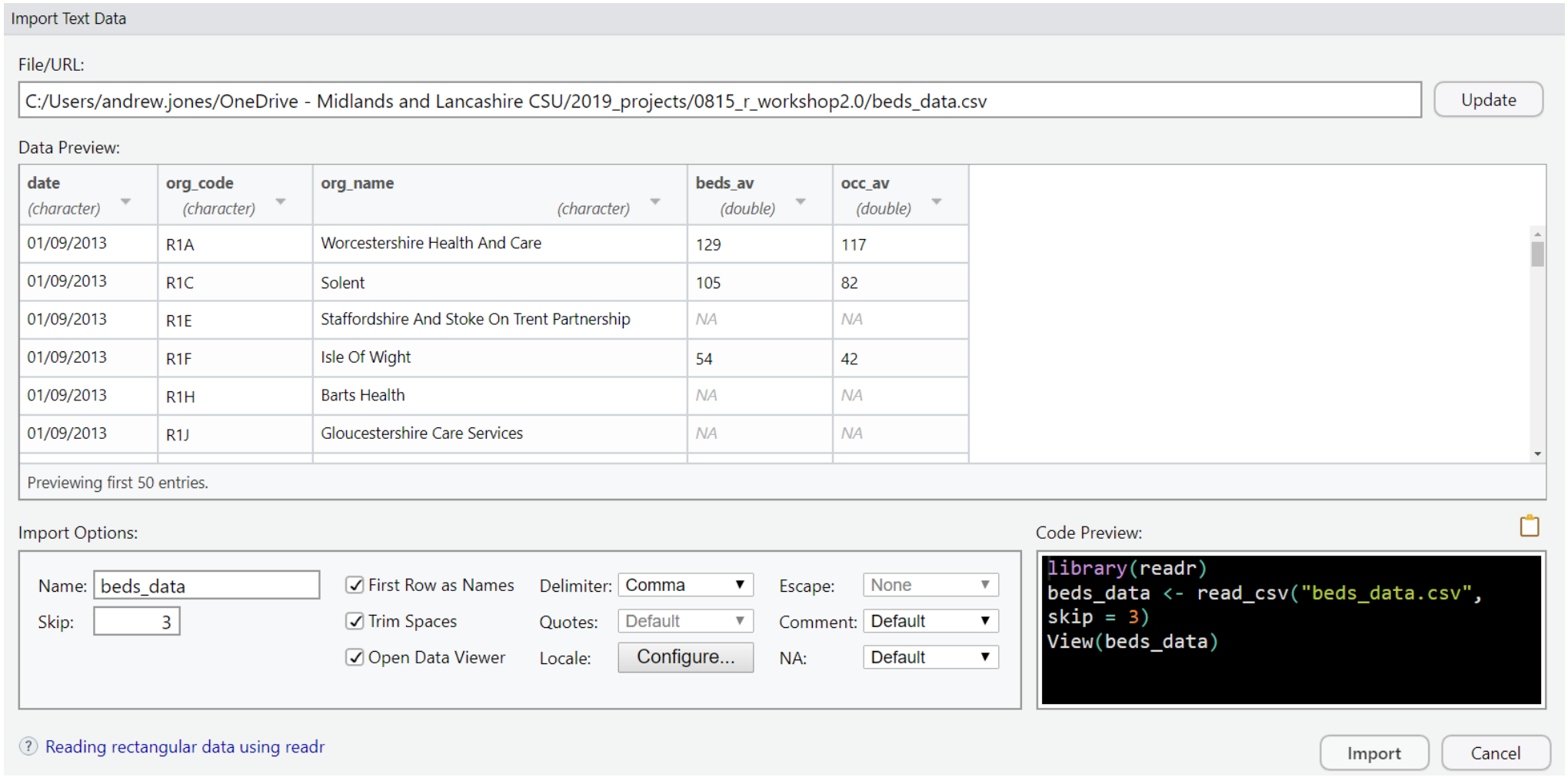


Less friendly csvs

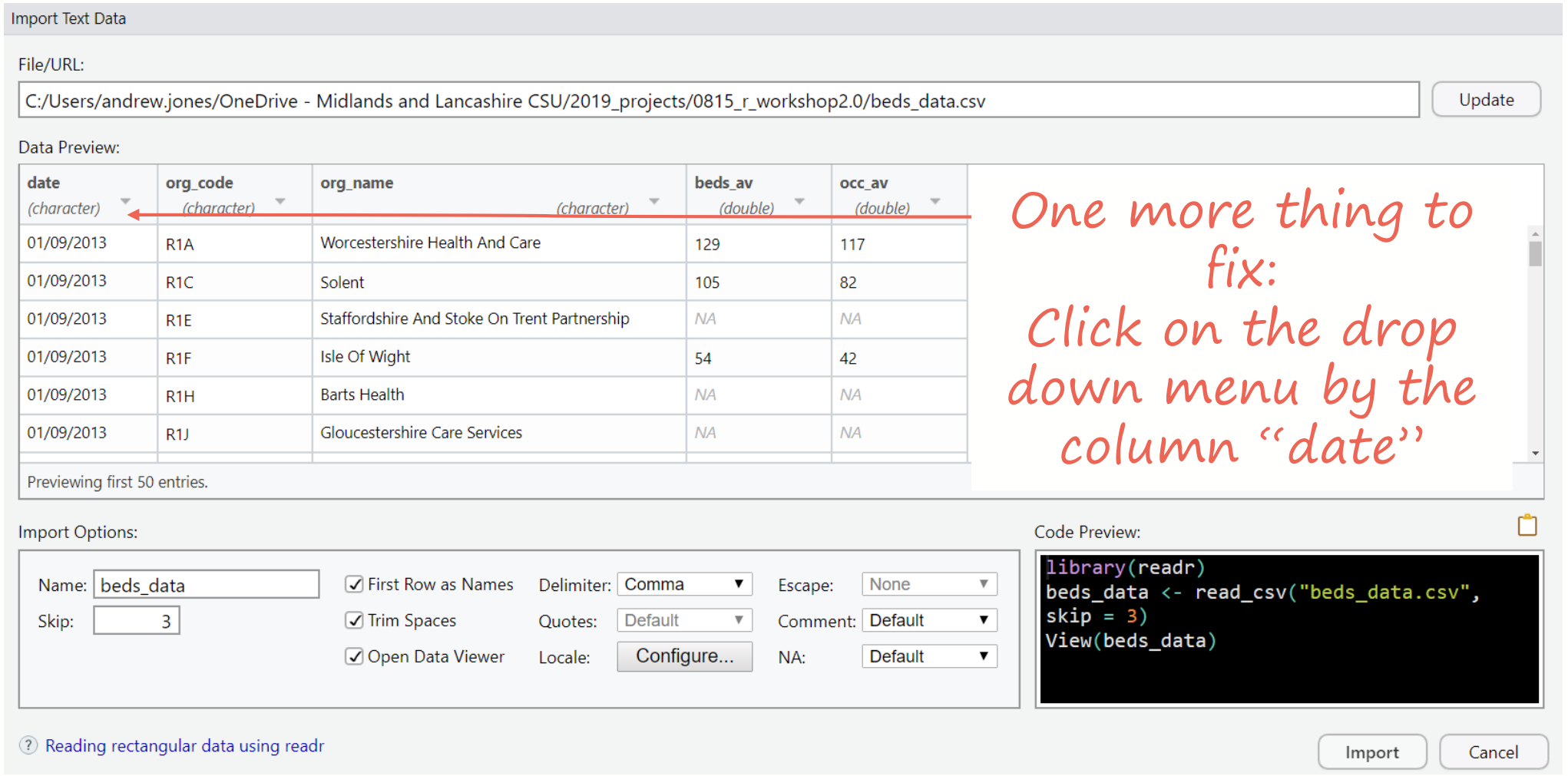
Load beds_data.csv



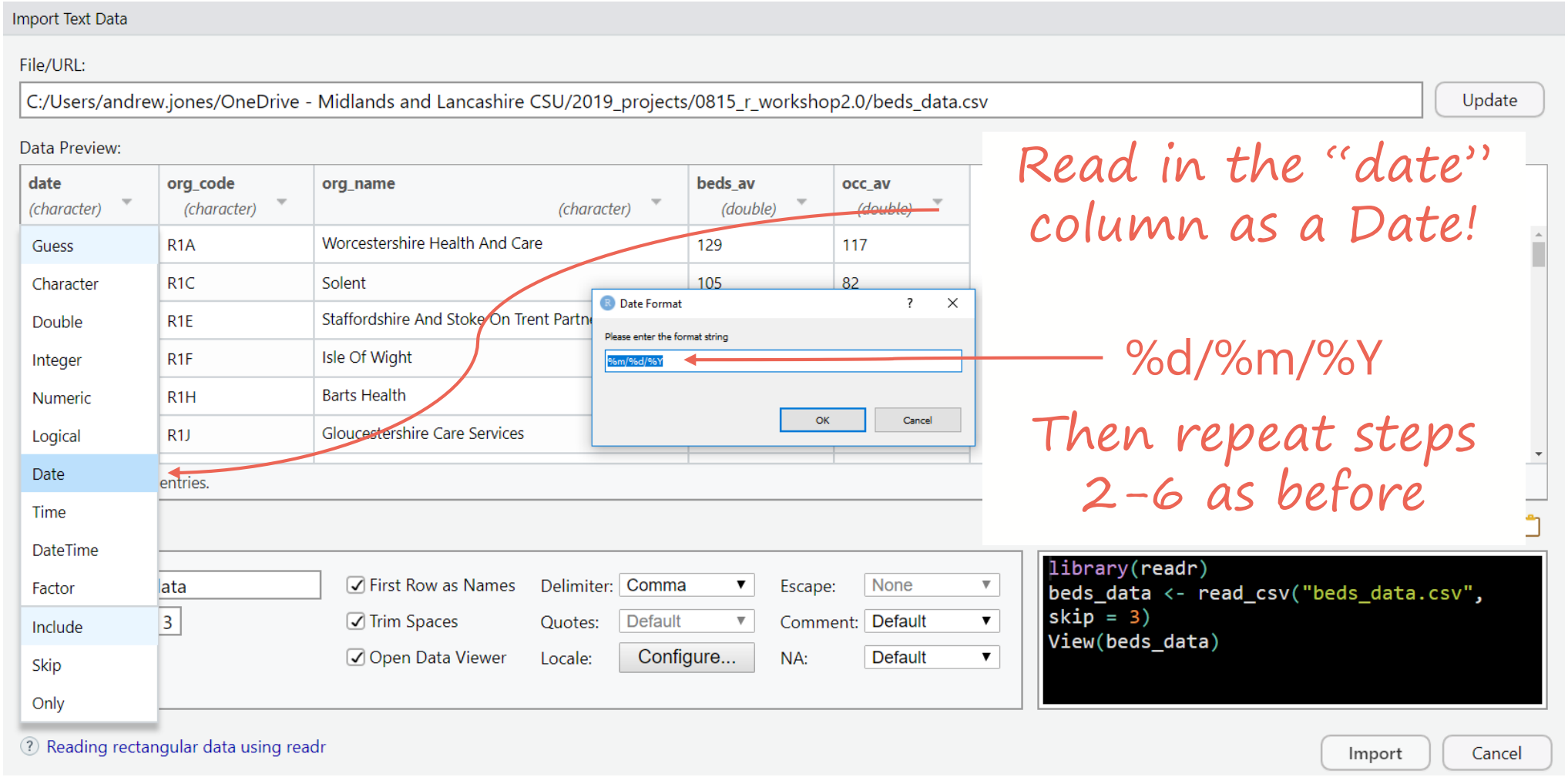
Less friendly csvs



Less friendly csvs



Less friendly csvs



Preview the data

The data:

Observations quarterly

Average number of beds available / occupied at midnight over the 3-month period.

date	org_code	org_name	beds_av	occ_av
-	-	-	NA	NA
-	-	-	48	33

This is real data – so there are real issues (which we'll work with)

The tool: dplyr

5 key verbs:

arrange
filter
mutate
group_by
summarise

will help us gain a deeper understanding of our data sets.

An aside:

Very soon we will want to use a series of these dplyr commands…

Series of commands = Recipe

Imagine a recipe for mashed potato:

Start with a...

potato then
peel then
slice into medium sized pieces then
boil for 25 minutes then
mash

An aside:

Imagine a recipe for mashed potato:

Start with an R object

`potato` then
peel then
slice into medium sized pieces then
boil for 25 minutes then
mash

An aside:

Imagine a recipe for mashed potato:

Start with an R object

`potato` then
`peel()` then
slice into medium sized pieces then
boil for 25 minutes then
mash

An aside:

Imagine a recipe for mashed potato:

potato then
peel() then
slice(size = “medium”) then
boil for 25 minutes then
mash

An aside:

Imagine a recipe for mashed potato:

potato then
peel() then
slice(size = “medium”) then
boil(t = 25) then
mash

An aside:

```
potato %>%
  peel() %>%
  slice(size = “medium”) %>%
  boil(t = 25)
```

An aside:

Imagine a recipe for mashed potato:

Input object

```
potato %>%
  peel() %>%
  slice(size = “medium”) %>%
  boil(t = 25)
```

Output = hot chopped potato

An aside:

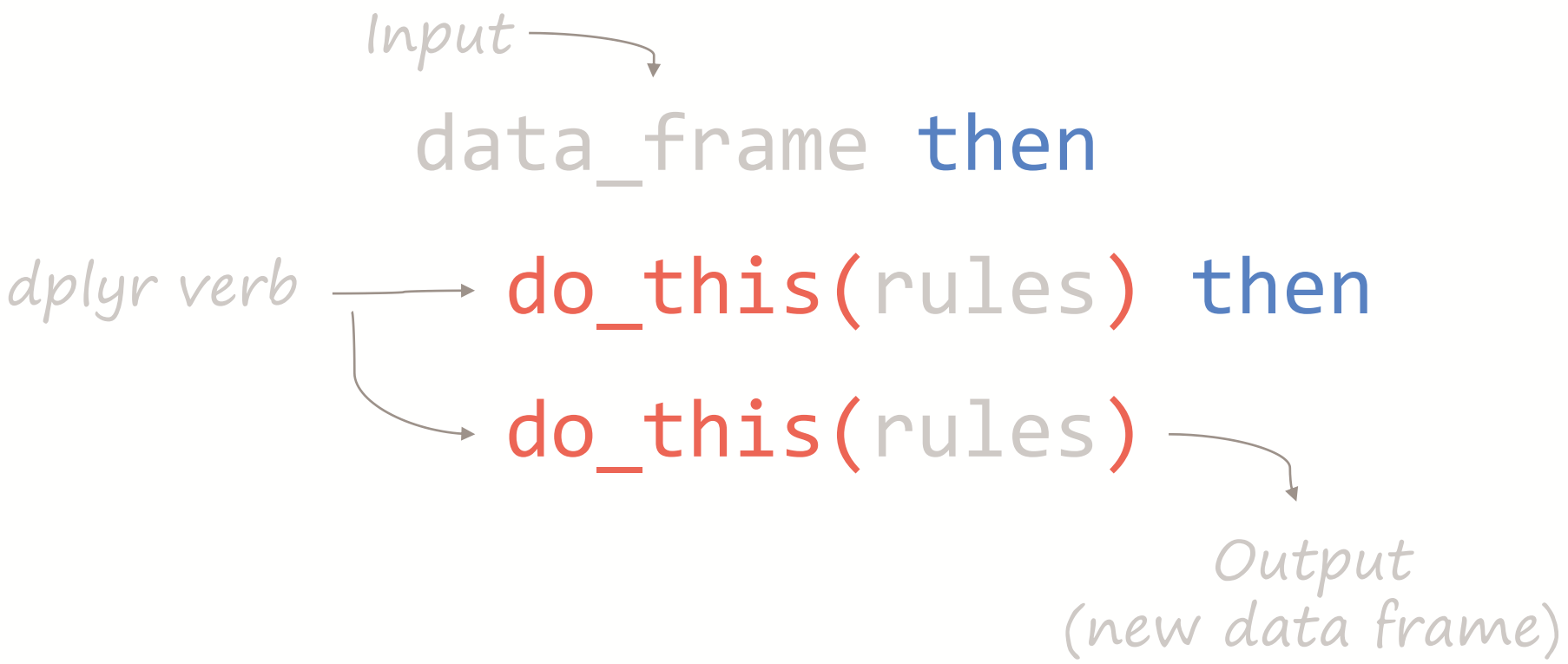
Imagine a recipe for mashed potato:

```
potato %>%
  peel() %>%
  slice(size = "medium") %>%
  boil(t = 25)
```

What is the output after this step?

Each step builds on the previous one

Tidyverse syntax

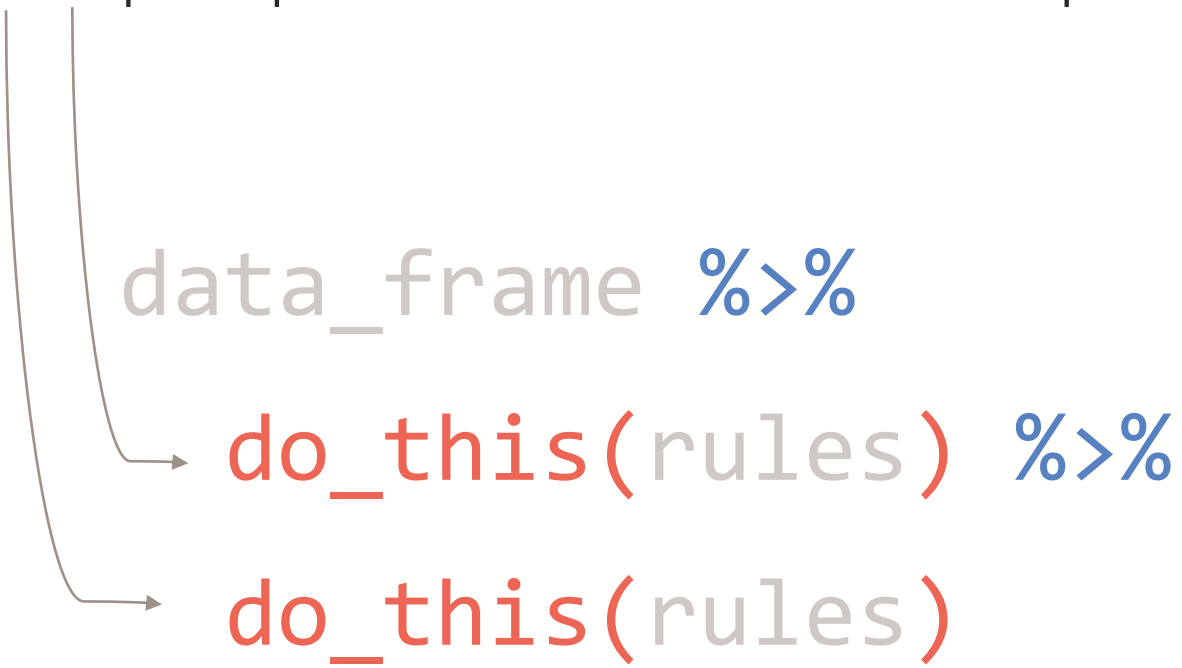


Tidyverse syntax

```
data_frame %>%
  do_this(rules) %>%
  do_this(rules)
```

The tidyverse

Combine simple pieces to solve complex puzzles

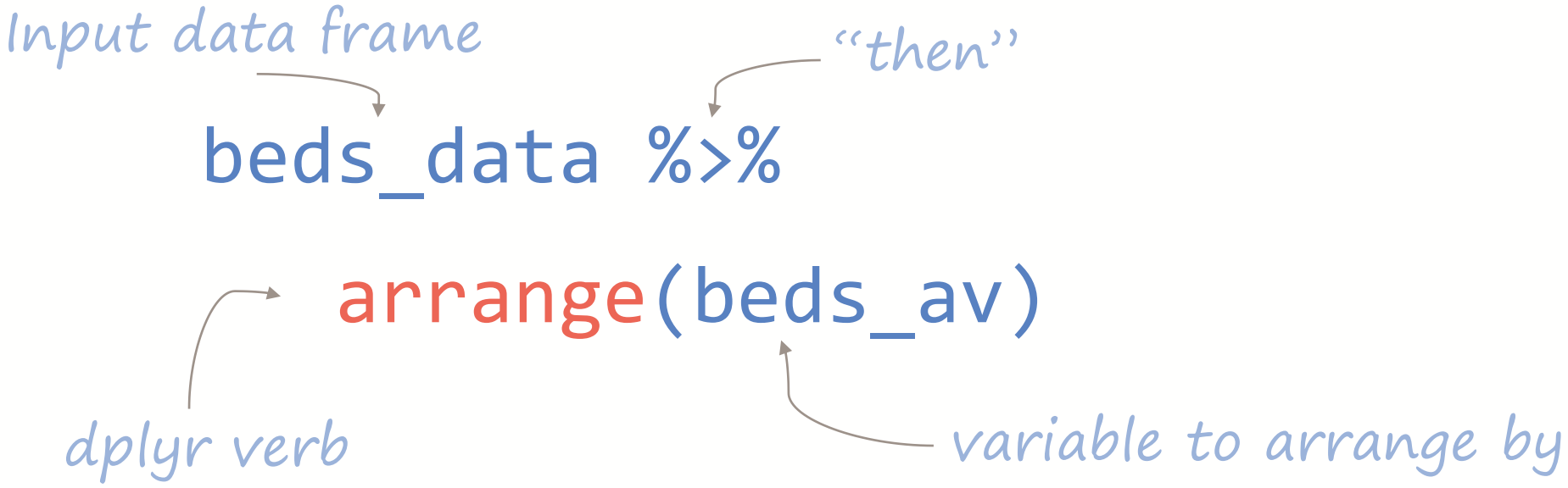


Using dplyr

Q1. Which organisation provided the highest number of Mental Health (MH) beds?

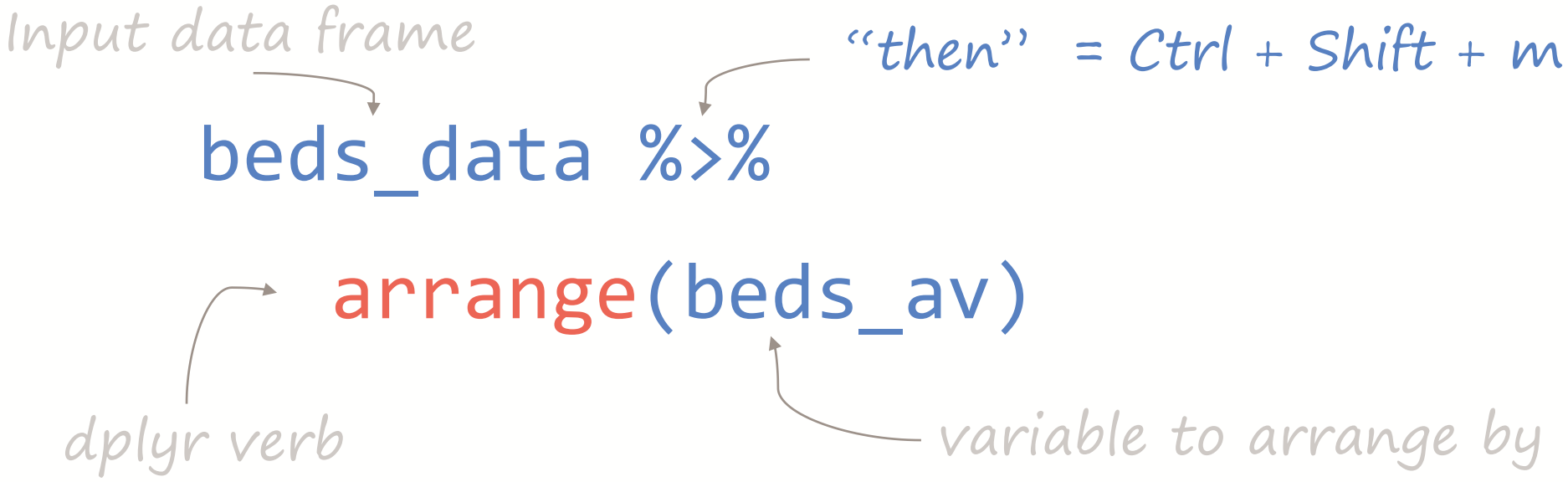
1. arrange

Reorder rows based on selected variable



1. arrange

Reorder rows based on selected variable



1. arrange

Reorder rows based on selected variable

```
beds_data %>%
    arrange(beds_av)
```

defaults to ascending order

1. arrange

As we'd like descending order:

```
beds_data %>%
  arrange(desc(beds_av))
```

for text and numeric variables

Q2. Which 2 organisations provided the highest number of MH beds in Sept. 2018?

We'll use **arrange()** as before

Q2. Which 2 organisations provided the highest number of MH beds in Sept. 2018?

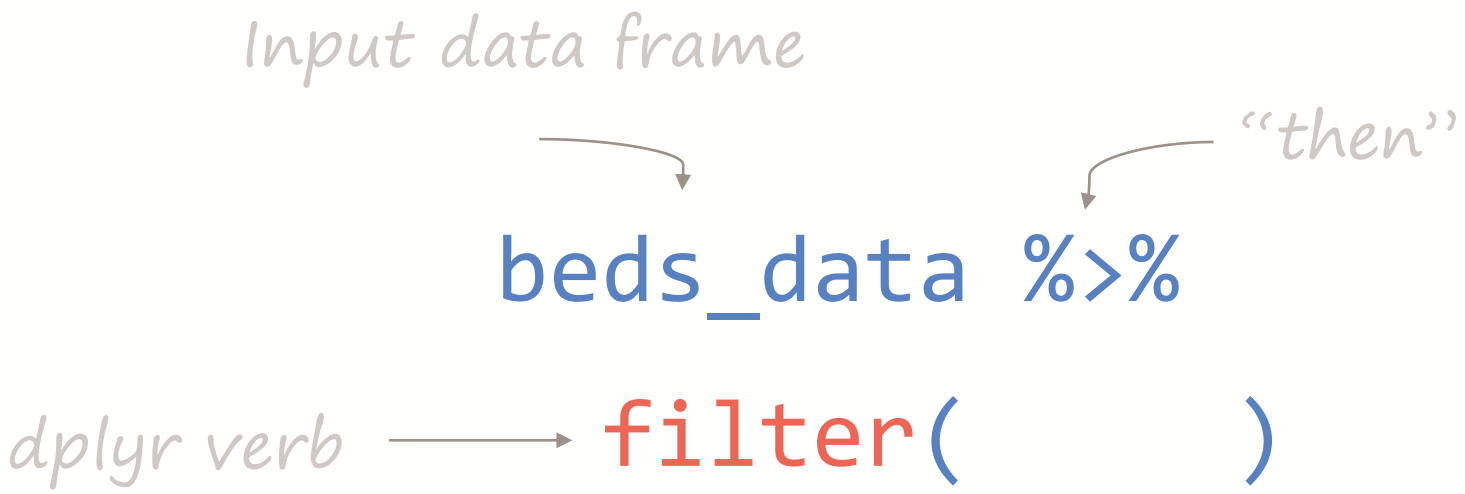
We'll use **arrange()** as before

Q2. Which 2 organisations provided the highest number of MH beds in Sept. 2018?

But we require only observations with this date

2. filter

pick observations by their value



2. filter

pick observations by their value

```
... %>%
  filter(date == "2018-09-01")
```

The expression inside brackets should return TRUE or FALSE

We are testing equality so ==

we are choosing rows where this expression is TRUE

2. filter

pick observations by their value

```
beds_data %>%
  arrange(desc(beds_av)) %>%
  filter(date == “2018-09-01”)
```

Q3. Which 5 organisations had the highest percentage bed occupancy in Sept. 2018?

We'll use `arrange()` as before

Q3. Which 5 organisations had the highest percentage bed occupancy in Sept. 2018?

We'll use `arrange()` as before

Q3. Which 5 organisations had the highest percentage bed occupancy in Sept. 2018?

`filter()` as before

We'll use `arrange()` as before

Q3. Which 5 organisations had the highest percentage bed occupancy in Sept. 2018?

We don't have this variable...

`filter()` as before

We'll use `arrange()` as before

Q3. Which 5 organisations had the highest percentage bed occupancy in Sept. 2018?

We don't have this variable...

but we can create it:

`filter()` as before

3. mutate

create new variables from existing ones


```
beds_data %>%
  mutate(perc_occ = occ_av / beds_av)
```

3. mutate

```
beds_data %>%
    mutate(perc_occ = occ_av / beds_av)
```

RHS usually a function of existing variable(s)

new column will be named

NOT a test of equality, so =

3. mutate

```
beds_data %>%
  mutate(perc_occ = occ_av / beds_av) %>%
  filter(date == "2018-09-01") %>%
        arrange(desc(perc_occ))
```

We can refer to variables we've just created above

Q4. What was the mean number of beds, (across all trusts) for each value of date?

Q4. What was the mean number of beds (across all trusts) for each value of date?

Let's first look at how we'd produce summary stats like a mean...

4. summarise

collapse many values into a single summary value

```
beds_data %>%
    summarise(mean_beds = mean(beds_av))
```

Similar syntax to **mutate**:

new column name

(summary) function using existing columns

4. summarise

collapse many values into a single summary value

```
beds_data %>%
    summarise(mean_beds = mean(beds_av))
```

but in this case our code returns **NA**

Similar syntax to **mutate**:

new column name

(summary) function using existing columns

4. summarise

collapse many values into a single summary value

```
beds_data %>%
summarise(mean_beds =
            mean(beds_av, na.rm = T))
```

We'll need to remove **NA** values to get a suitable mean

T is short for **TRUE**

4. summarise

collapse many values into a single summary value

```
beds_data %>%
summarise(mean_beds =
                  mean(beds_av, na.rm = T))
```

This code produces a single summary value for the whole dataset

Q4. What was the mean number of beds, (across all trusts) for each value of date?

Q4. What was the mean number of beds, (across all trusts) for each value of date?

Now we know how to use summarise...

Q4. What was the mean number of beds, (across all trusts) for each value of date?

Now we know how to use summarise...

We'll produce a summary value for each value of date

5. group_by

For each group...

```
beds_data %>%
  group_by(date) %>% ...
```

For each value of date...

(do something):
***group_by** does nothing to the change the data frame...*

But it does change a setting behind the scenes

5. group_by

For each group… summarise (produce single summary value)

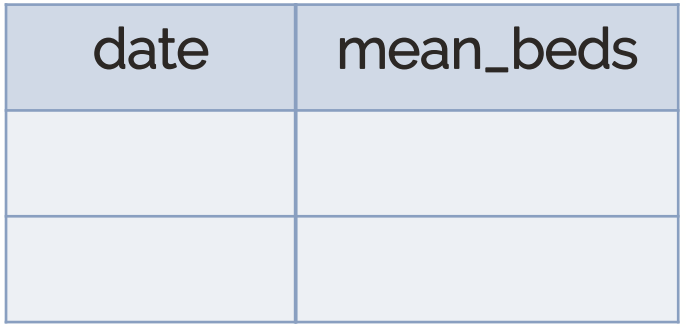
```
beds_data %>%
    group_by(date) %>%
     summarise(
        mean_beds = mean(beds_av, na.rm =T)
                )
```

a very common pattern:

group_by *and* summarise

A column is created for each grouping variable

A column for the summary




```
beds_data %>%
    group_by(date) %>%
     summarise(
        mean_beds = mean(beds_av, na.rm = T)
              )
```

a very common pattern:

group_by *and* summarise

A row for each unique value of date

date	mean_beds
"2010-06-01"	
"2010-09-01"	

```
beds_data %>%
    group_by(date) %>%
     summarise(
        mean_beds = mean(beds_av, na.rm = T)
                )
```

5a. ungroup

Often it's safest to remove the grouping after you've performed the required operation

```
beds_data %>%
 group_by(date) %>%
 summarise(
    mean_beds = mean(beds_av, na.rm =T)) %>%
 ungroup()
```

Which 5 organisations have the highest mean % bed occupancy?

Which 5 organisations have the highest mean % bed occupancy?

(over the 5 year period)

Which 5 organisations have the highest mean % bed occupancy?

1. *Create a new variable*

`mutate()`

2. Then, for each of the...

`group_by()`

Which 5 organisations have the highest mean % bed occupancy?

1. Create a new variable

`mutate()`

2. Then, for each of the...

`group_by()`

3. Summary stat using

`summarise`

Which 5 organisations have the highest mean % bed occupancy?

1. Create a new variable

`mutate()`

2. Then, for each of the...

`group_by()`

3. Summary stat using

`summarise`

Which 5 organisations have the highest mean % bed occupancy?

1. Create a new variable

`mutate()`

4. As before, we'll use

`arrange`

2. Then, for each of the...

`group_by()`

3. Summary stat using

`summarise`

Which 5 organisations have the highest mean % bed occupancy?

1. Create a new variable

`mutate()`

4. As before, we'll use

`arrange`

Tip: Run the code after each new line to check it returns the output you'd expect.

Over to you…

2. Then, for each of the...

`group_by()`

3. Summary stat using `summarise`

Which 5 organisations have the highest mean % bed occupancy?

1. Create a new variable `mutate()`

4. As before, we'll use `arrange`

```
beds_data %>%
  mutate(_ = _)%>%
  group_by() %>%
  summarise(_ = _ ) %>%
  arrange()
```

Tip: Run the code after each new line to check it returns the output you'd expect.

Solution

```
beds_data %>%
  mutate(perc_occ = occ_av / beds_av) %>%
  group_by(org_name) %>%
  summarise(mean_pocc =
              mean(perc_occ, na.rm = T)) %>%
  arrange(desc(mean_pocc))
```

Extension:

How many columns associated with each observation?

```
summarise(number = n())
```

This is a common pattern – it will count the number of rows associated with each group

Solution

```
beds_data %>%
  mutate(perc_occ = occ_av / beds_av) %>%
  group_by(org_name) %>%
  summarise(mean_pocc= mean(perc_occ),
            number = n())%>%
  arrange(desc(mean_pocc))
```

Adding another summary column

6. select

select a subset of variables from existing data set

```
beds_data %>%
        select(org_code, org_name)
```

6. select

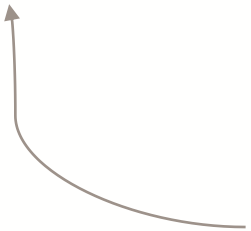
select a subset of variables from existing data set

```
beds_data %>%
        select(-org_code)
```

6. select

select a subset of variables from existing data set

```
beds_data %>%
        select(1:3)
```

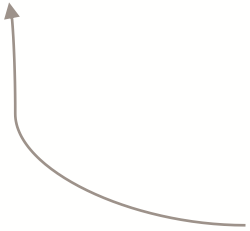


You can also refer to columns by number. Here 1:5 saves having to type: 1,2,3,4,5

6. select

select a subset of variables from existing data set

```
beds_data %>%
        select(org_name, everything())
```



If you want this column at the start of your data frame

This work is licensed as

End

Addendum

An aside: Data frames part II

With ggplot and dplyr we have been working with data frames.

Variables in columns

date	org	beds_av
"2010-03-01"	"5AA"	100
"2010-03-01"	"5BB"	80
"2010-03-01"	"5CC"	200

An aside: Data frames part II

We can think of a data frame as a series of columns, bound together by an invisible "data frame" structure.

date	org	beds_av
"2010-03-01"	"5AA"	100
"2010-03-01"	"5BB"	80
"2010-03-01"	"5CC"	200

An aside: Data frames part II

If we remove the structure we can look at the columns in isolation.

Data frame structure removed →

beds_av
100
80
200

An aside: Data frames part II

These isolated columns can be thought of as vectors.

beds_av
100
80
200

An aside: Data frames part II

Vector types include integer and character

beds_av
100
80
200

An aside: Data frames part II

You can create a vector with **`c()`**

```
c(100, 80, 200)
```

c stands for "combine"

beds_av
100
80
200