

# data wrangling with dplyr

Credit: JustToolazy. Licensed under CC BY 2.0.

# 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:

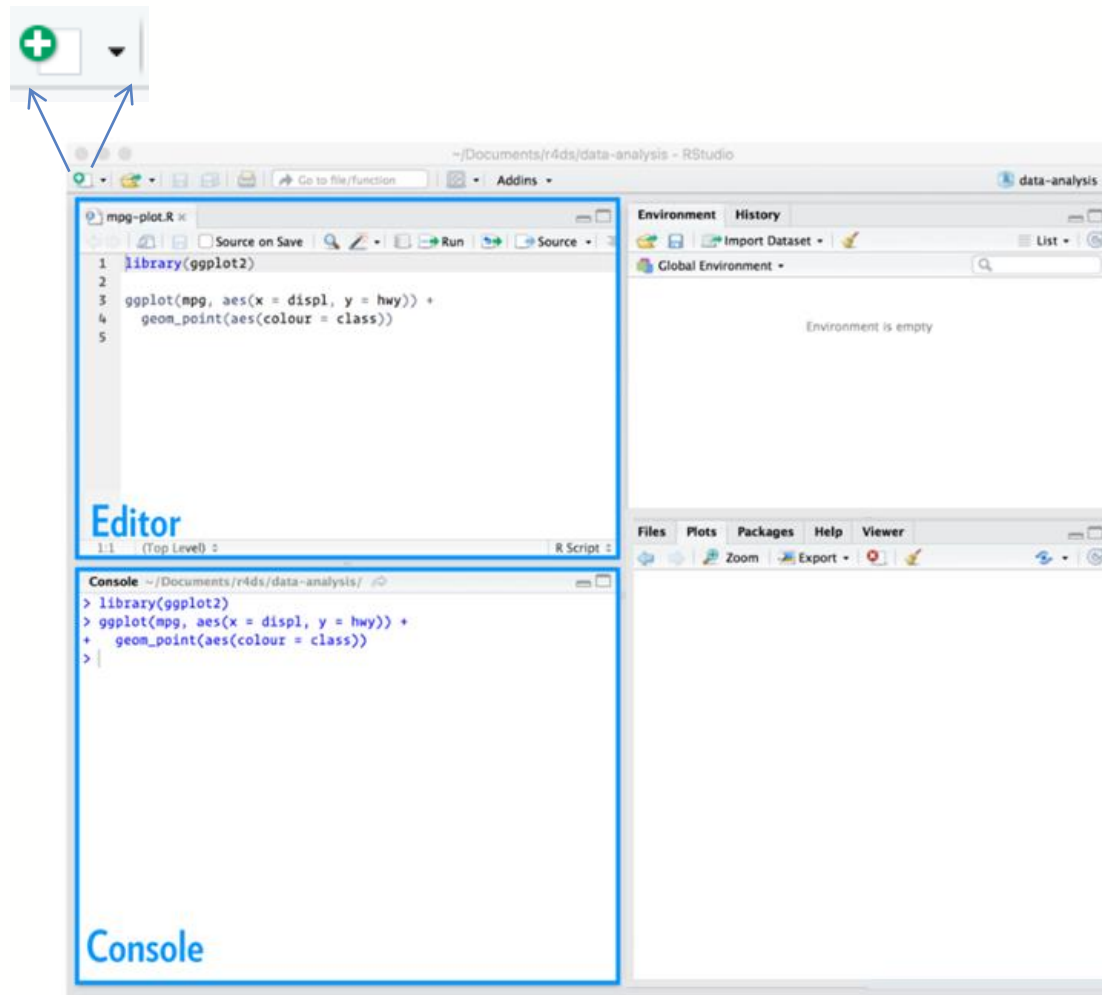
KH03 returns (bed numbers and occupancy) by organisation, published by NHS England.

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`

Import Text Data

File/URL:  
C:/Users/andrew.jones/OneDrive - Midlands and Lancashire CSU/2019\_projects/0815\_r\_workshop2.0/beds\_data.csv Update

Data Preview:

Title: (character)	KH03 Returns (character)	X3 (character)	X4 (character)	X5 (character)
Source	https://www.england.nhs.uk/statistics/statistical-work-areas/...	NA	NA	NA
NA	NA	NA	NA	NA
date	org_code	org_name	beds_av	occ_av
01/09/2013	R1A	Worcestershire Health And Care	129	117
01/09/2013	R1C	Solent	105	82
01/09/2013	R1E	Staffordshire And Stoke On Trent Partnership	NA	NA

Notice that something isn't quite right here

Import Options:

Name:  ☒ First Row as Names Delimiter:  Escape:

Skip:  ☒ Trim Spaces Quotes:  Comment:

☒ Open Data Viewer Locale:  NA:

Code Preview:

```
library(readr)
beds_data <- read_csv("beds_data.csv")
View(beds_data)
```

Import Cancel

? Reading rectangular data using readr

# Less friendly csvs

Load `beds_data.csv`

Import Text Data

File/URL:  
C:/Users/andrew.jones/OneDrive - Midlands and Lancashire CSU/2019\_projects/0815\_r\_workshop2.0/beds\_data.csv Update

Data Preview:

Title: (character)	KH03 Returns (character)	X3 (character)	X4 (character)	X5 (character)
Source	https://www.england.nhs.uk/statistics/statistical-work-areas/...	NA	NA	NA
NA	NA	NA	NA	NA
date	org_code	org_name	beds_av	occ_av
01/09/2013	R1A	Worcestershire Health And Care	129	117
01/09/2013	R1C	Solent	105	82
01/09/2013	R1E	Staffordshire And Stoke On Trent Partnership	NA	NA

Previewing first 50 entries.

Import Options:

Name:  ☒ First Row as Names Delimiter:  Escape:

Skip:  ☒ Trim Spaces Quotes:  Comment:

☒ Open Data Viewer Locale:  NA:

Code Preview:

```
library(readr)
beds_data <- read_csv("beds_data.csv")
View(beds_data)
```

Import Cancel

? Reading rectangular data using readr

*Metadata present above the data we'd like*

# Less friendly csvs

## Load `beds_data.csv`

Import Text Data

File/URL:  
C:/Users/andrew.jones/OneDrive - Midlands and Lancashire CSU/2019\_projects/0815\_r\_workshop2.0/beds\_data.csv Update

Data Preview:

Title: (character)	KH03 Returns (character)	X3
Source	https://www.england.nhs.uk/statistics/statistical-work-areas/...	NA
NA	NA	NA
date	org_code	org_name
01/09/2013	R1A	Worcestershire Heal
01/09/2013	R1C	Solent
01/09/2013	R1E	Staffordshire And St

Previewing first 50 entries.

Import Options:

Name: `beds_data` ☒ First Row as Names Delimiter: `Comma` Escape: `None`

Skip: `0` ☒ Trim Spaces Quotes: `Default` Comment: `Default`

☒ Open Data Viewer Locale: `Configure...` NA: `Default`

Code Preview:

```
library(readr)
beds_data <- read_csv("beds_data.csv")
View(beds_data)
```

Import Cancel

? Reading rectangular data using readr

In this case, "Skip" 3 rows  
so the column headers  
appear at the top of the  
preview...



# Less friendly csvs

Import Text Data

File/URL:  
C:/Users/andrew.jones/OneDrive - Midlands and Lancashire CSU/2019\_projects/0815\_r\_workshop2.0/beds\_data.csv

Update

Data Preview:

date (character)	org_code (character)	org_name (character)	beds_av (double)	occ_av (double)
01/09/2013	R1A	Worcestershire Health And Care	129	117
01/09/2013	R1C	Solent	105	82
01/09/2013	R1E	Staffordshire And Stoke On Trent Partnership	NA	NA
01/09/2013	R1F	Isle Of Wight	54	42
01/09/2013	R1H	Barts Health	NA	NA
01/09/2013	R1J	Gloucestershire Care Services	NA	NA

Previewing first 50 entries.

Import Options:

Name: beds\_data

Skip: 3

☒ First Row as Names

☒ Trim Spaces

☒ Open Data Viewer

Delimiter: Comma

Quotes: Default

Locale: Configure...

Escape: None

Comment: Default

NA: Default

Code Preview:

```
library(readr)
beds_data <- read_csv("beds_data.csv",
skip = 3)
View(beds_data)
```

? Reading rectangular data using readr

Import

Cancel

15

# Less friendly csvs

Import Text Data

File/URL:  
C:/Users/andrew.jones/OneDrive - Midlands and Lancashire CSU/2019\_projects/0815\_r\_workshop2.0/beds\_data.csv Update

Data Preview:

date (character)	org_code (character)	org_name (character)	beds_av (double)	occ_av (double)
01/09/2013	R1A	Worcestershire Health And Care	129	117
01/09/2013	R1C	Solent	105	82
01/09/2013	R1E	Staffordshire And Stoke On Trent Partnership	NA	NA
01/09/2013	R1F	Isle Of Wight	54	42
01/09/2013	R1H	Barts Health	NA	NA
01/09/2013	R1J	Gloucestershire Care Services	NA	NA

Previewing first 50 entries.

Import Options:

Name:  ☒ First Row as Names Delimiter:  Escape:   
Skip:  ☒ Trim Spaces Quotes:  Comment:   
☒ Open Data Viewer Locale:  NA:

Code Preview:

```
library(readr)
beds_data <- read_csv("beds_data.csv",
skip = 3)
View(beds_data)
```

? Reading rectangular data using readr Import Cancel

One more thing to fix:  
Click on the drop down menu by the column "date"

# Less friendly csvs

Import Text Data

File/URL:  
C:/Users/andrew.jones/OneDrive - Midlands and Lancashire CSU/2019\_projects/0815\_r\_workshop2.0/beds\_data.csv Update

Data Preview:

date (character)	org_code (character)	org_name (character)	beds_av (double)	occ_av (double)
Guess	R1A	Worcestershire Health And Care	129	117
Character	R1C	Solent	105	82
Double	R1E	Staffordshire And Stoke On Trent Partn		
Integer	R1F	Isle Of Wight		
Numeric	R1H	Barts Health		
Logical	R1J	Gloucestershire Care Services		
Date				
Time				
DateTime				
Factor				
Include				
Skip				
Only				

entries.

data ☐ First Row as Names ☒ Trim Spaces ☒ Open Data Viewer Delimiter: Comma Escape: None Quotes: Default Comment: Default NA: Default

3

Locale: Configure...

Reading rectangular data using readr

Import Cancel

*Read in the "date" column as a Date!*

*%d/%m/%Y*

*Then repeat steps 2-6 as before*

```
library(readr)
beds_data <- read_csv("beds_data.csv",
skip = 3)
View(beds_data)
```

# 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()** %>%

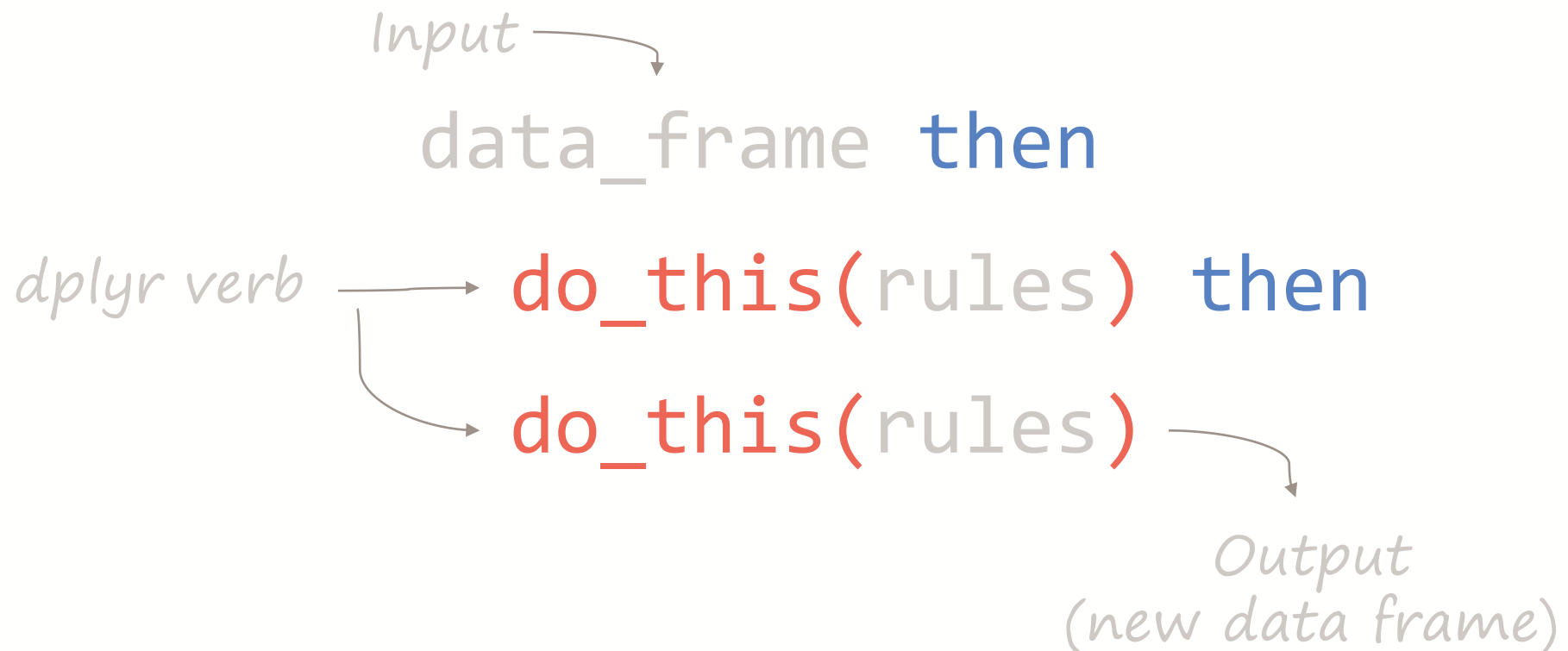
**slice(size = "medium")** %>%

**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



data\_frame %>%

do\_this(rules) %>%

do\_this(rules)



# Using dplyr

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

# 1. arrange

Reorder rows based on selected variable

*Input data frame*

`beds_data %>%`

*“then”*

`arrange(beds_av)`

*dplyr verb*

*variable to arrange by*

# 1. arrange

Reorder rows based on selected variable

*Input data frame*

`beds_data %>%`

*“then” = Ctrl + Shift + m*

`arrange(beds_av)`

*dplyr verb*

*variable to arrange by*

# 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

*Input data frame*

*beds\_data* %>%

*“then”*

*dplyr verb* → **filter**( )

## 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)

new column will  
be named

NOT a test of  
equality, so =

RHS usually a  
function of  
existing  
variable(s)

# 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))
```

*Similar syntax to  
mutate:*

*new column  
name*

*(summary)  
function using  
existing  
columns*

*but in this  
case our code  
returns NA*



# 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*

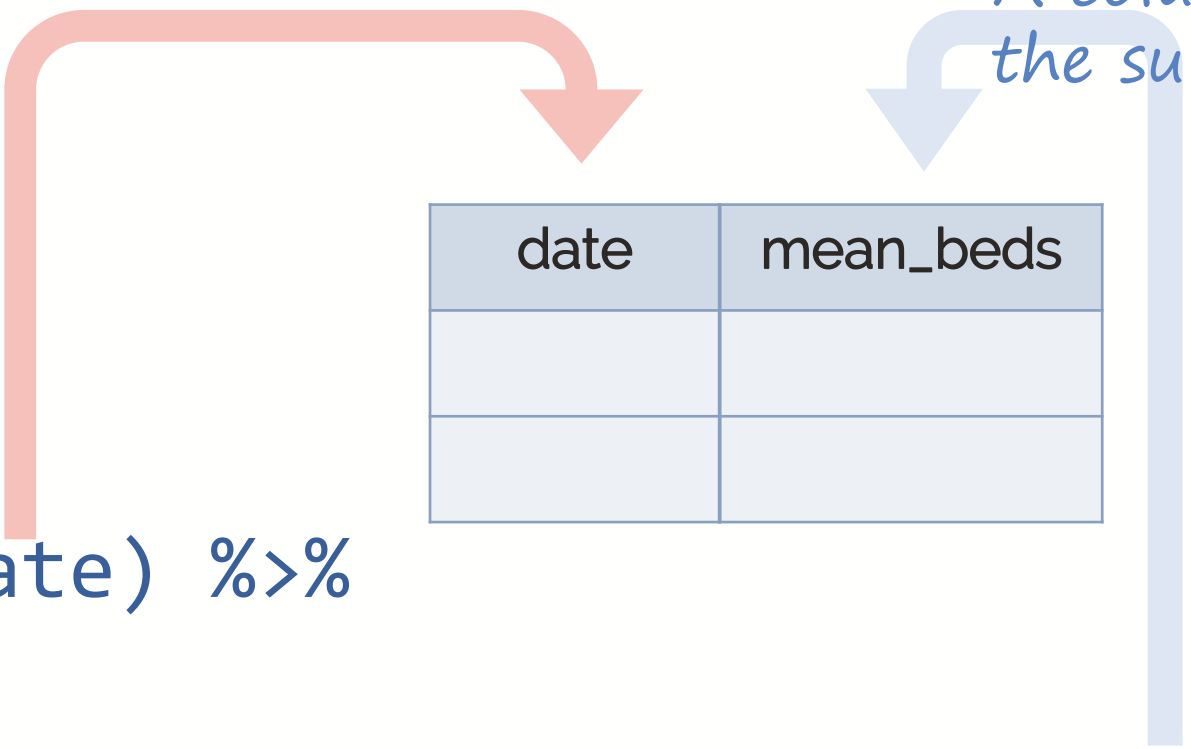
*A column for  
the summary*

```
beds_data %>%
```

```
  group_by(date) %>%
```

```
  summarise(
```

```
    mean_beds = mean(beds_av, na.rm = T)  
  )
```



date	mean_beds



*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()

Which 5 organisations  
have the highest **mean**  
% bed occupancy?

2. Then, for  
each of the...  
↓ group\_by()

3. Summary  
stat using  
**summarise**

1. Create a new variable  
mutate()

Which 5 organisations  
have the **highest** mean  
% bed occupancy?

```
graph TD; Q[Which 5 organisations have the highest mean % bed occupancy?]; Q --> S1[1. Create a new variable mutate()]; Q --> S2[2. Then, for each of the... group_by()]; Q --> S3[3. Summary stat using summarise]; Q --> S4[4. As before, we'll use arrange];
```

2. Then, for each of the... `group_by()`

3. Summary stat using `summarise`

1. Create a new variable  
`mutate()`

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



Which 5 organisations  
have the highest mean  
% bed occupancy?

```
graph TD; S1[1. Create a new variable  
mutate()] --> Q[Which 5 organisations  
have the highest mean  
% bed occupancy?]; S2[2. Then, for  
each of the...  
group_by()] --> Q; S3[3. Summary  
stat using  
summarise] --> Q; S4[4. As before,  
we'll use  
arrange] --> Q;
```

1. Create a new variable  
`mutate()`

2. Then, for  
each of the...

`group_by()`

3. Summary  
stat using  
`summarise`

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

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

Over to you...

Which 5 organisations  
have the highest mean  
% bed occupancy?

1. Create a new variable  
`mutate()`

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

2. Then, for  
each of the...

`group_by()`

3. Summary  
stat using  
`summarise`

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

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

# 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 %>%
```

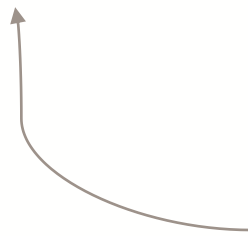
```
  select(org_code, org_name)
```

# 6. select

select a subset of variables from existing data set

```
beds_data %>%
```

```
  select(-org_code)
```



*To remove a column*

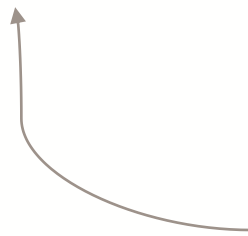


# 6. select

select a subset of variables from existing data set

```
beds_data %>%
```

```
  select(1:3)
```



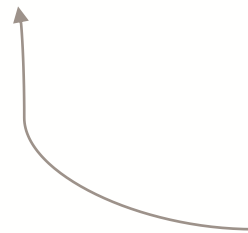
*You can also refer to columns by number. Here 1:5 saves having to type: 1,2,3,4,5*

# 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

Creative Commons

Attribution-ShareAlike 4.0

International

To view a copy of this license, visit

<https://creativecommons.org/licenses/by-sa/4.0/>

For title photo:

<https://creativecommons.org/licenses/by/2.0/>

End

# Addendum

# An aside: Data frames part II

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

*Variables in columns*



<b>date</b>	<b>org</b>	<b>beds_av</b>
"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.

<b>date</b>	<b>org</b>	<b>beds_av</b>
"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