Easy data manipulation

{tidyr} // {dplyr}

{tidyr}

Tidying data.

Why should we tidy data?

- Consistent data structure makes your life easier!
- ‘Tidy data’ helps you to:
  - – Aggregate and explore (dplyr)
  - – Visualise (ggplot2)
  - – Model (lm…)
  - – Re-use previous code

What is tidy data?

- Tidy data consists of **key/value** pairs
  - – Key/value pairs are associations between variable names and observations of that variable
  - – A key is a variable name (e.g. CITY)
  - – A value is an observation of that variable (e.g. ‘Glasgow’)
- Columns: variables
- Rows: observations

What is tidy data?

- What does this data set **mean**?
- What are its 3 defining variables?

	Dark morph	Light morph
Male	8	2
Female	4	6

What is tidy data?

Sex	Morph	Count
Female	Dark	4
Female	Light	6
Male	Dark	8
Male	Light	2

Mapping meaning to structure

	Dark morph	Light morph
Male	8	2
Female	4	6

→

Sex	Morph	Count
Female	Dark	4
Female	Light	6
Male	Dark	8
Male	Light	2

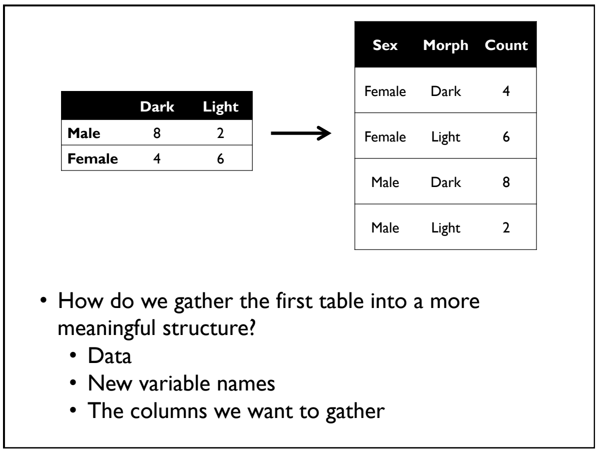
Mapping meaning to structure

Each column is a variable.

Each row is an observation.

A row shows the count for one morph, for one sex.

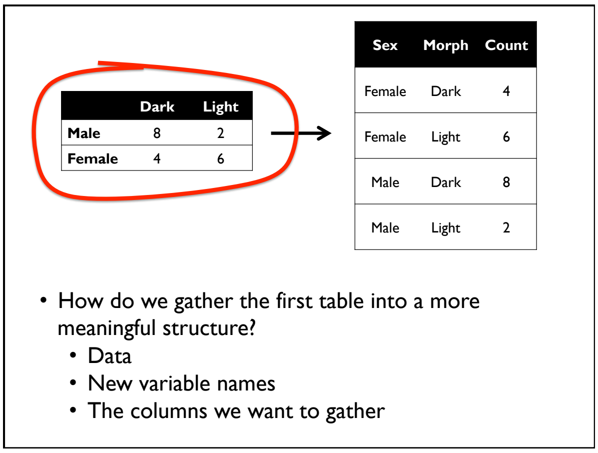
Sex	Morph	Count
Female	Dark	4
Female	Light	6
Male	Dark	8
Male	Light	2



	Dark	Light
Male	8	2
Female	4	6

Sex	Morph	Count
Female	Dark	4
Female	Light	6
Male	Dark	8
Male	Light	2

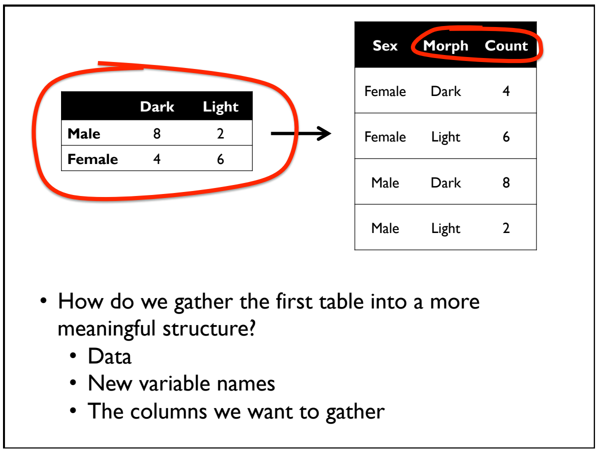
- How do we gather the first table into a more meaningful structure?
    - Data
    - New variable names
    - The columns we want to gather



	Dark	Light
Male	8	2
Female	4	6

Sex	Morph	Count
Female	Dark	4
Female	Light	6
Male	Dark	8
Male	Light	2

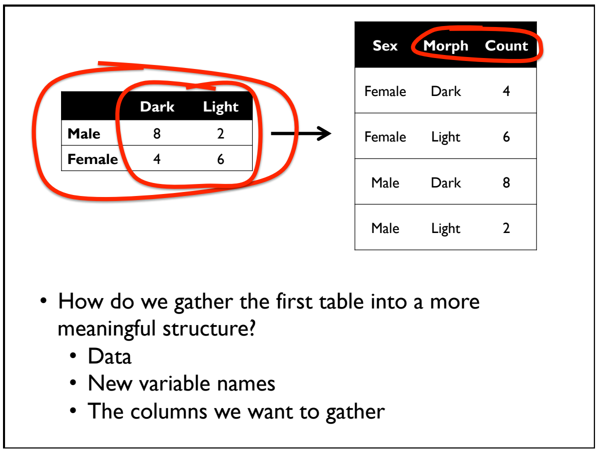
- How do we gather the first table into a more meaningful structure?
    - Data
    - New variable names
    - The columns we want to gather



	Dark	Light
Male	8	2
Female	4	6

Sex	Morph	Count
Female	Dark	4
Female	Light	6
Male	Dark	8
Male	Light	2

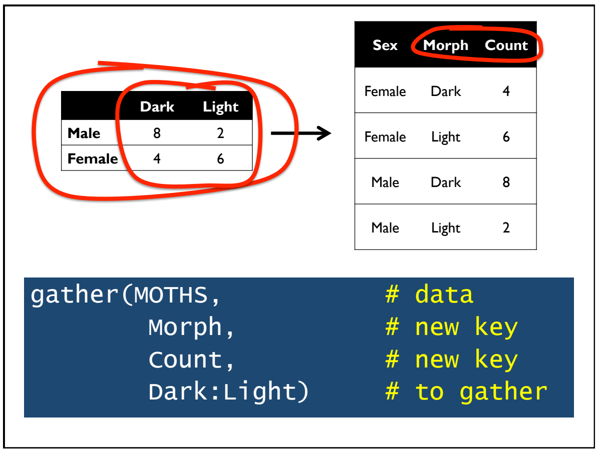
- How do we gather the first table into a more meaningful structure?
    - Data
    - New variable names
    - The columns we want to gather



	Dark	Light
Male	8	2
Female	4	6

Sex	Morph	Count
Female	Dark	4
Female	Light	6
Male	Dark	8
Male	Light	2

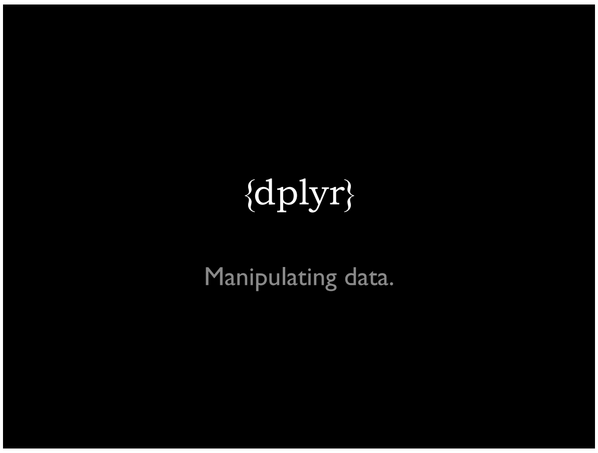
- How do we gather the first table into a more meaningful structure?
    - Data
    - New variable names
    - The columns we want to gather



	Dark	Light
Male	8	2
Female	4	6

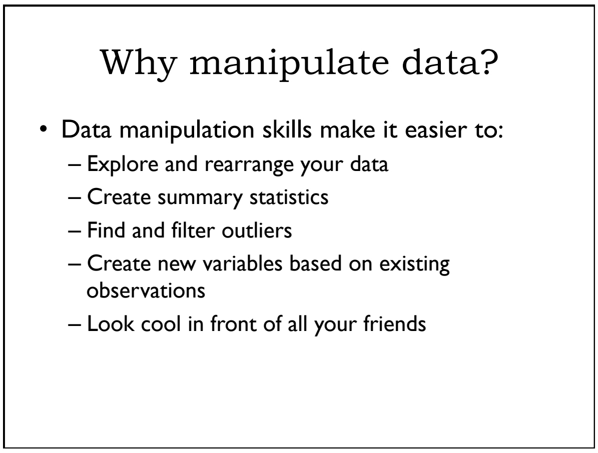
Sex	Morph	Count
Female	Dark	4
Female	Light	6
Male	Dark	8
Male	Light	2

```
gather(MOTHS,                # data
        Morph,               # new key
        Count,               # new key
        Dark:Light)          # to gather
```



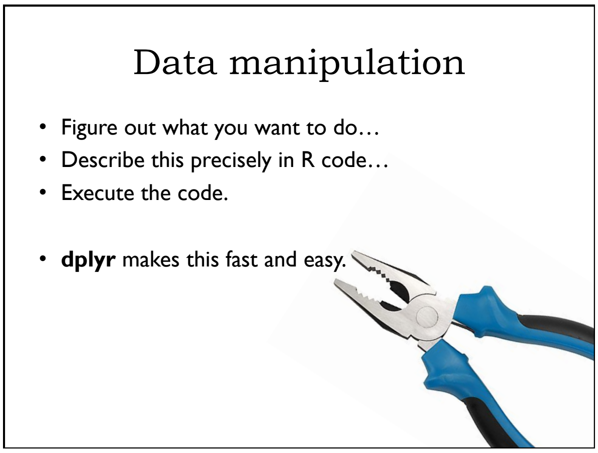
{dplyr}

Manipulating data.



Why manipulate data?

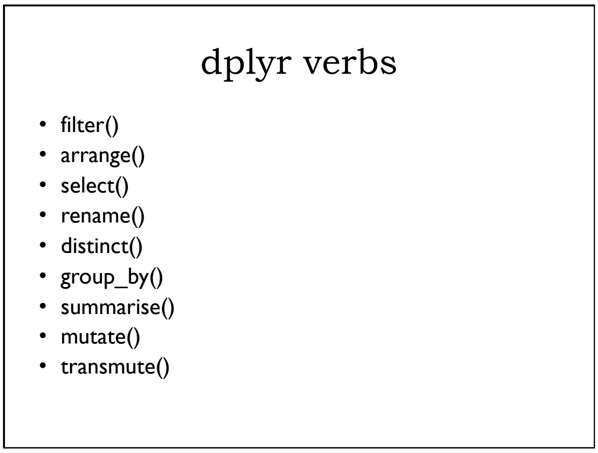
- Data manipulation skills make it easier to:
  - Explore and rearrange your data
  - Create summary statistics
  - Find and filter outliers
  - Create new variables based on existing observations
  - Look cool in front of all your friends



Data manipulation

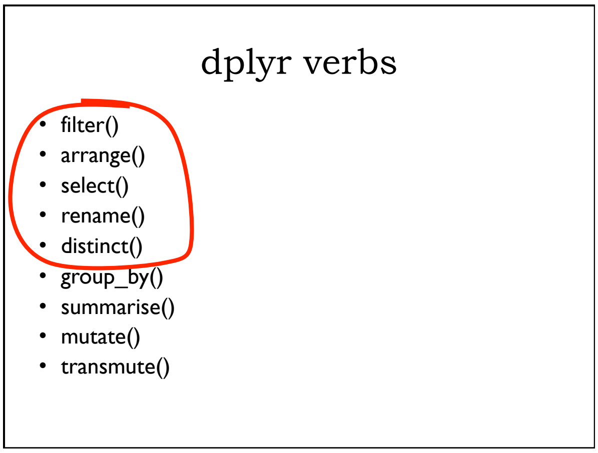
- Figure out what you want to do…
- Describe this precisely in R code…
- Execute the code.

- **dplyr** makes this fast and easy.



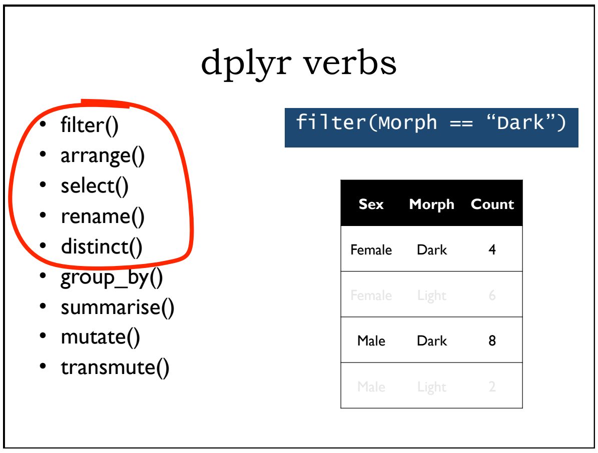
dplyr verbs

- filter()
- arrange()
- select()
- rename()
- distinct()
- group_by()
- summarise()
- mutate()
- transmute()



dplyr verbs

- filter()
- arrange()
- select()
- rename()
- distinct()
- group_by()
- summarise()
- mutate()
- transmute()

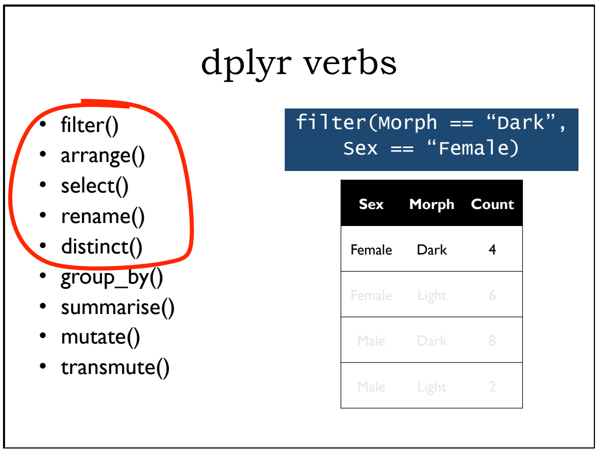


dplyr verbs

- filter()
- arrange()
- select()
- rename()
- distinct()
- group_by()
- summarise()
- mutate()
- transmute()

```
filter(Morph == “Dark”)
```

Sex	Morph	Count
Female	Dark	4
Female	Light	6
Male	Dark	8
Male	Light	2

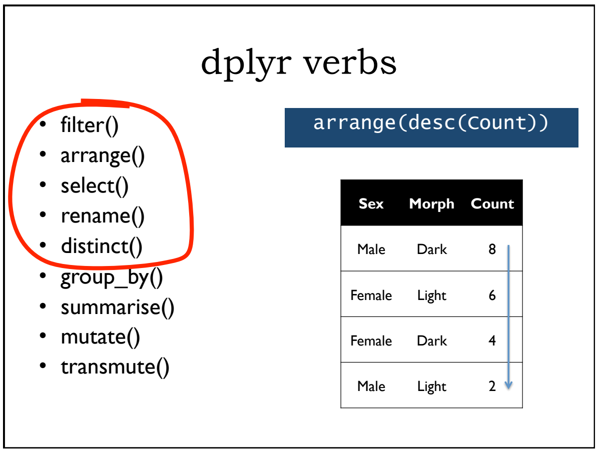


dplyr verbs

- filter()
- arrange()
- select()
- rename()
- distinct()
- group_by()
- summarise()
- mutate()
- transmute()

```
filter(Morph == “Dark”,
  Sex == “Female)
```

Sex	Morph	Count
Female	Dark	4
Female	Light	6
Male	Dark	8
Male	Light	2

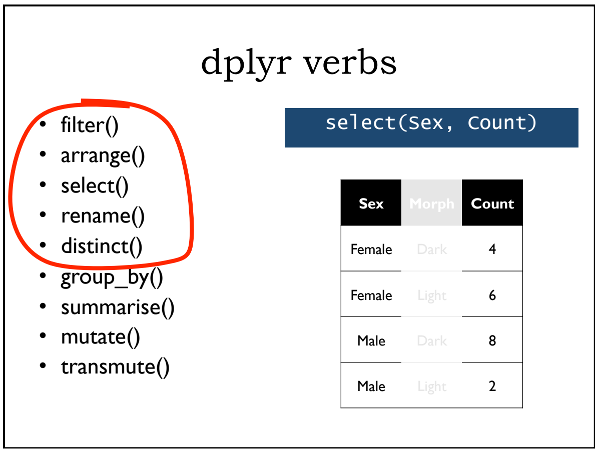


dplyr verbs

- filter()
- arrange()
- select()
- rename()
- distinct()
- group_by()
- summarise()
- mutate()
- transmute()

```
arrange(desc(Count))
```

Sex	Morph	Count
Male	Dark	8
Female	Light	6
Female	Dark	4
Male	Light	2

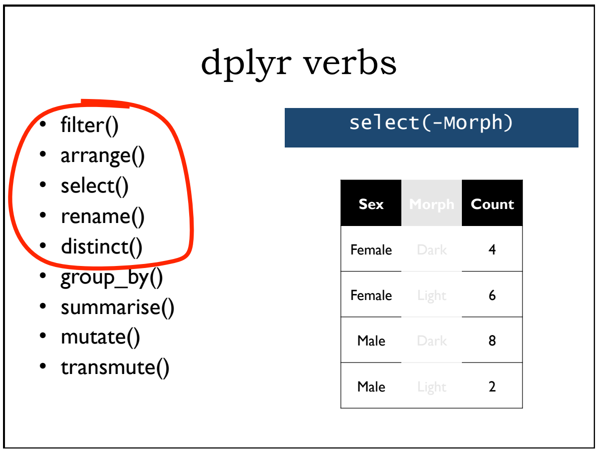


dplyr verbs

- filter()
- arrange()
- select()
- rename()
- distinct()
- group_by()
- summarise()
- mutate()
- transmute()

```
select(Sex, Count)
```

Sex	Morph	Count
Female	Dark	4
Female	Light	6
Male	Dark	8
Male	Light	2

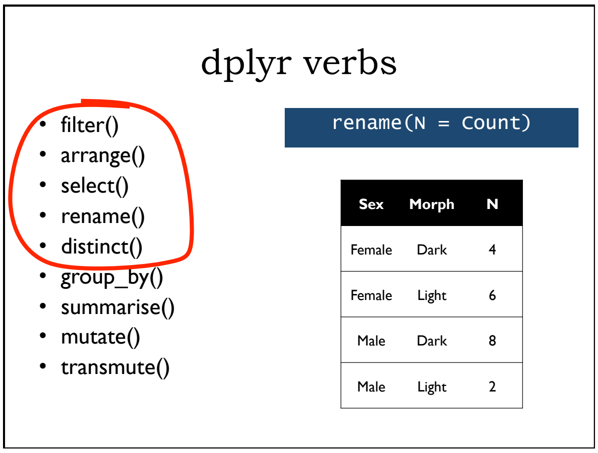


dplyr verbs

- filter()
- arrange()
- select()
- rename()
- distinct()
- group_by()
- summarise()
- mutate()
- transmute()

```
select(-Morph)
```

Sex	Morph	Count
Female	Dark	4
Female	Light	6
Male	Dark	8
Male	Light	2

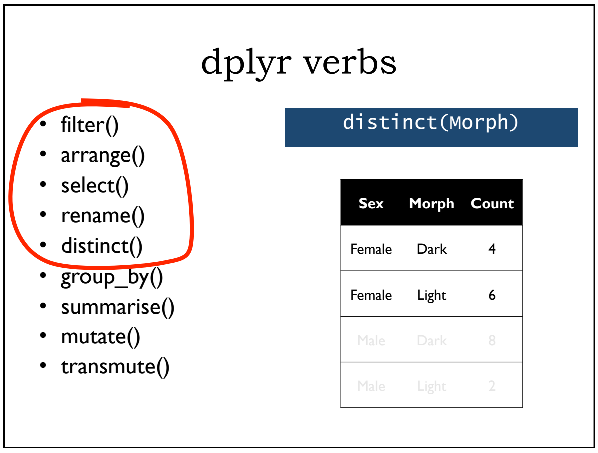


dplyr verbs

- filter()
- arrange()
- select()
- rename()
- distinct()
- group_by()
- summarise()
- mutate()
- transmute()

```
rename(N = Count)
```

Sex	Morph	N
Female	Dark	4
Female	Light	6
Male	Dark	8
Male	Light	2

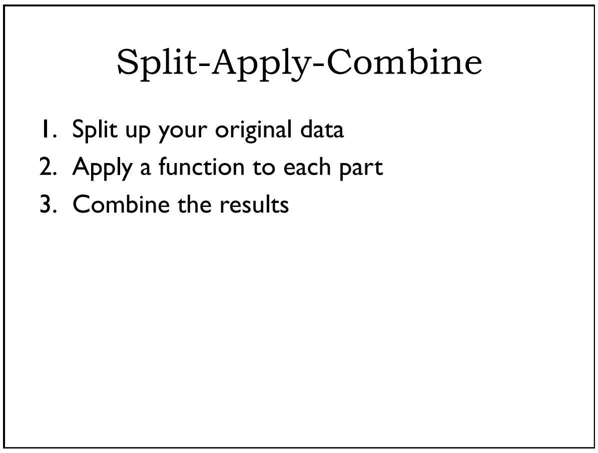


dplyr verbs

- filter()
- arrange()
- select()
- rename()
- distinct()
- group_by()
- summarise()
- mutate()
- transmute()

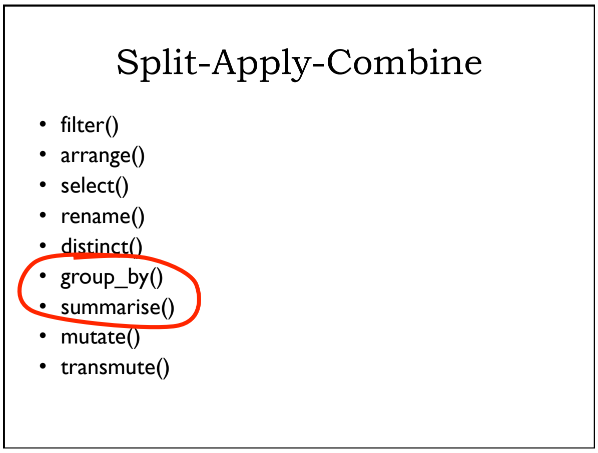
```
distinct(Morph)
```

Sex	Morph	Count
Female	Dark	4
Female	Light	6
Male	Dark	8
Male	Light	2



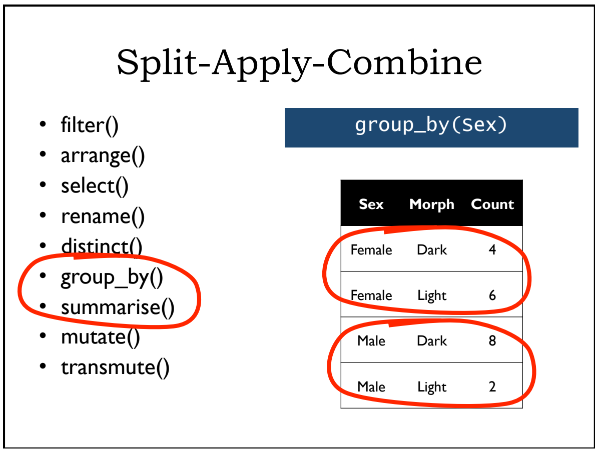
Split-Apply-Combine

1. Split up your original data
2. Apply a function to each part
3. Combine the results



Split-Apply-Combine

- filter()
- arrange()
- select()
- rename()
- distinct()
- group_by()
- summarise()
- mutate()
- transmute()

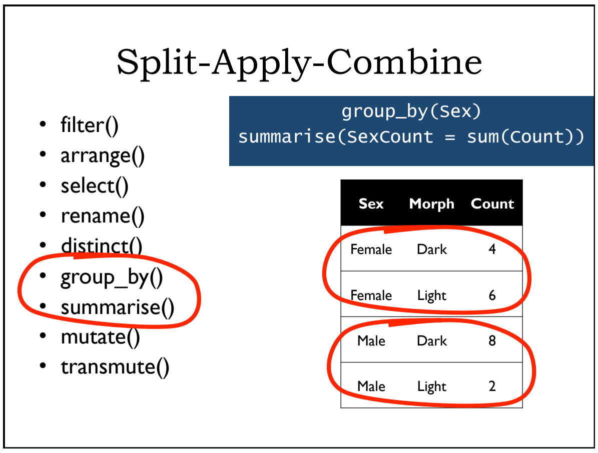


Split-Apply-Combine

- filter()
- arrange()
- select()
- rename()
- distinct()
- group_by()
- summarise()
- mutate()
- transmute()

```
group_by(Sex)
```

Sex	Morph	Count
Female	Dark	4
Female	Light	6
Male	Dark	8
Male	Light	2

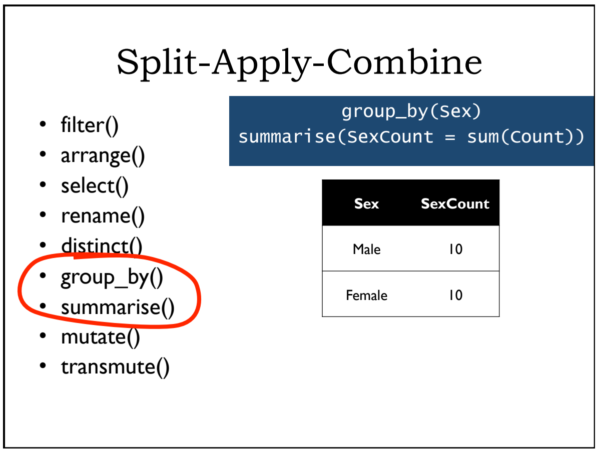


Split-Apply-Combine

- filter()
- arrange()
- select()
- rename()
- distinct()
- group_by()
- summarise()
- mutate()
- transmute()

```
group_by(Sex)
summarise(SexCount = sum(Count))
```

Sex	Morph	Count
Female	Dark	4
Female	Light	6
Male	Dark	8
Male	Light	2

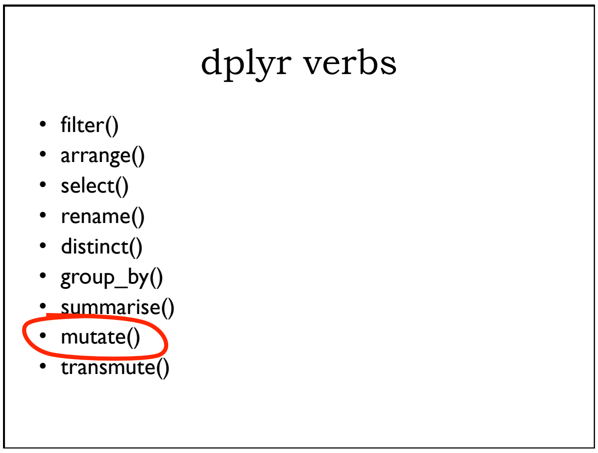


Split-Apply-Combine

- filter()
- arrange()
- select()
- rename()
- distinct()
- group_by()
- summarise()
- mutate()
- transmute()

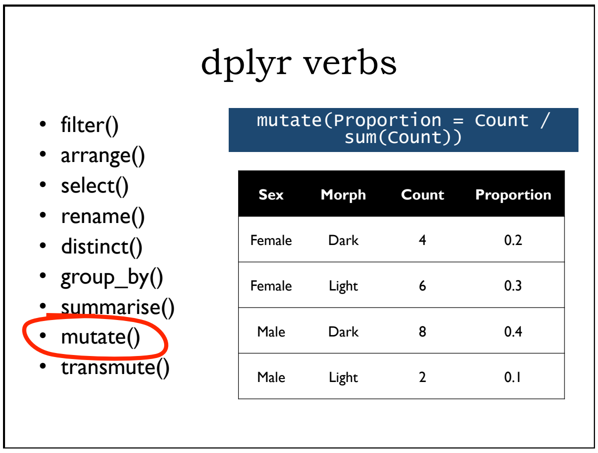
```
group_by(Sex)
summarise(SexCount = sum(Count))
```

Sex	SexCount
Male	10
Female	10



dplyr verbs

- filter()
- arrange()
- select()
- rename()
- distinct()
- group_by()
- summarise()
- mutate()
- transmute()

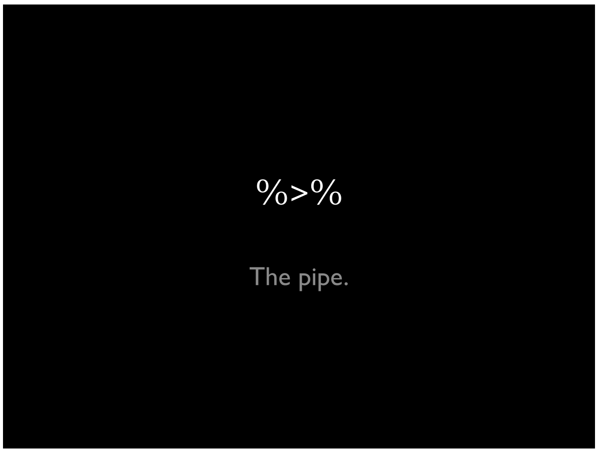


dplyr verbs

- filter()
- arrange()
- select()
- rename()
- distinct()
- group_by()
- summarise()
- mutate()
- transmute()

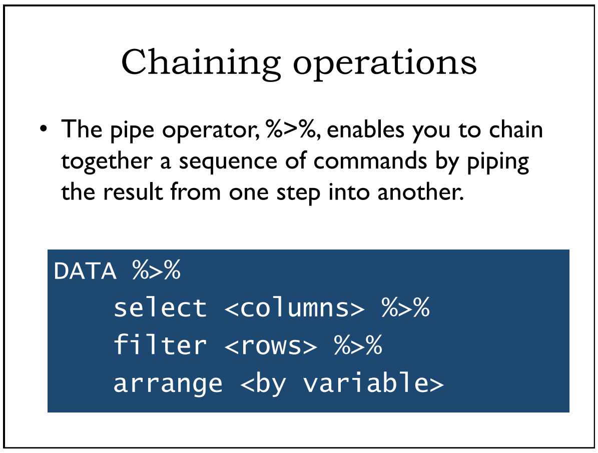
```
mutate(Proportion = Count /
sum(Count))
```

Sex	Morph	Count	Proportion
Female	Dark	4	0.2
Female	Light	6	0.3
Male	Dark	8	0.4
Male	Light	2	0.1



%>%

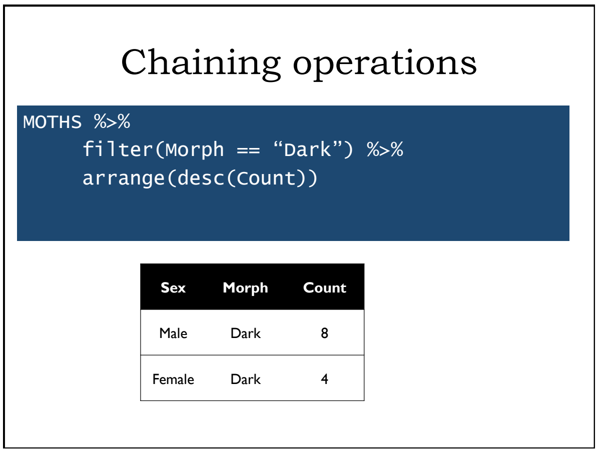
The pipe.



Chaining operations

- The pipe operator, %>%, enables you to chain together a sequence of commands by piping the result from one step into another.

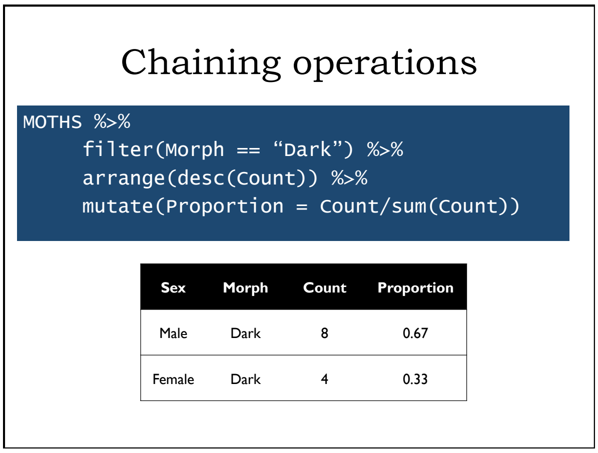
```
DATA %>%
    select <columns> %>%
    filter <rows> %>%
    arrange <by variable>
```



Chaining operations

```
MOTHS %>%
    filter(Morph == “Dark”) %>%
    arrange(desc(Count))
```

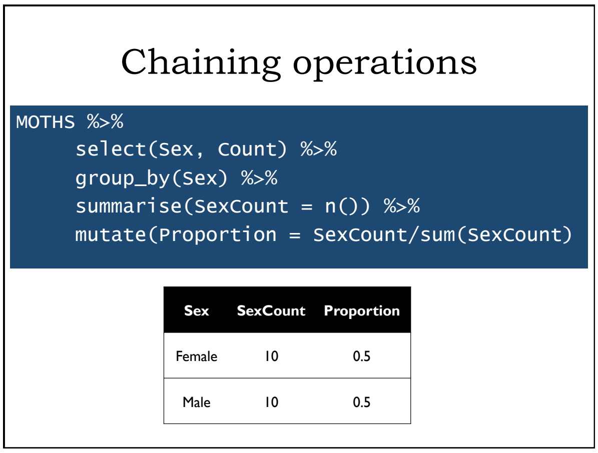
Sex	Morph	Count
Male	Dark	8
Female	Dark	4



Chaining operations

```
MOTHS %>%
    filter(Morph == “Dark”) %>%
    arrange(desc(Count)) %>%
    mutate(Proportion = Count/sum(Count))
```

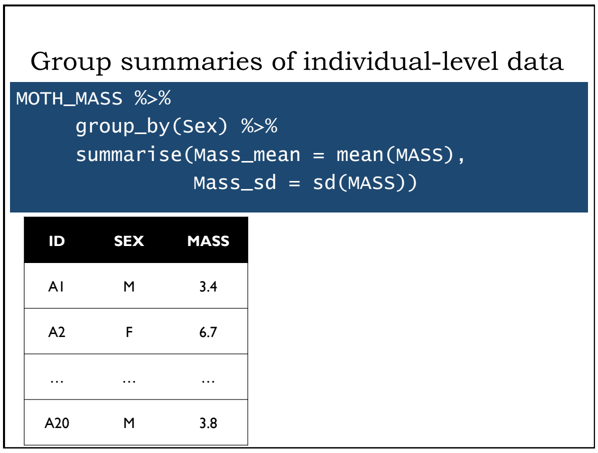
Sex	Morph	Count	Proportion
Male	Dark	8	0.67
Female	Dark	4	0.33



Chaining operations

```
MOTHS %>%
     select(Sex, Count) %>%
     group_by(Sex) %>%
     summarise(SexCount = n()) %>%
     mutate(Proportion = SexCount/sum(SexCount)
```

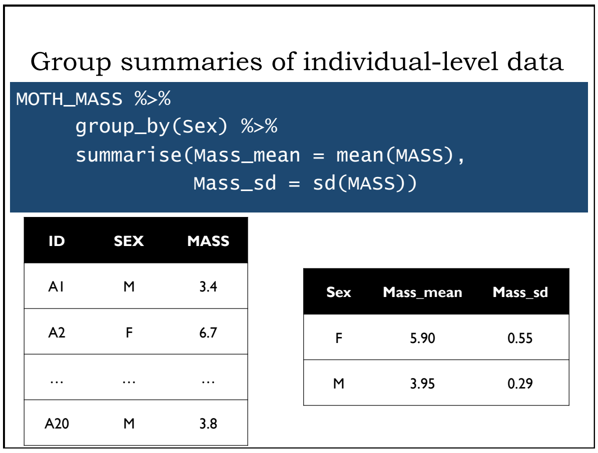
Sex	SexCount	Proportion
Female	10	0.5
Male	10	0.5



Group summaries of individual-level data

```
MOTH_MASS %>%
     group_by(Sex) %>%
     summarise(Mass_mean = mean(MASS),
               Mass_sd = sd(MASS))
```

ID	SEX	MASS
A1	M	3.4
A2	F	6.7
…	…	…
A20	M	3.8



Group summaries of individual-level data

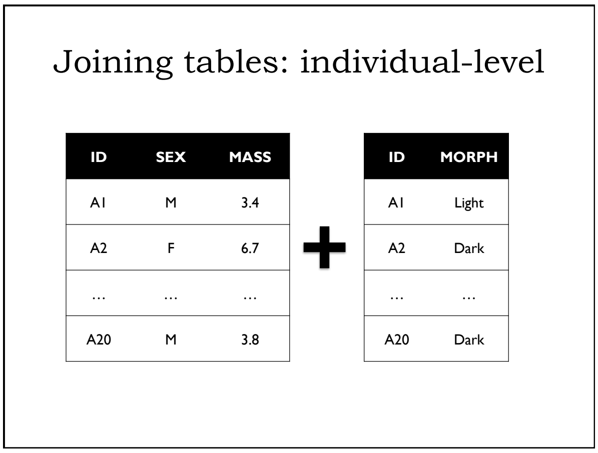
```
MOTH_MASS %>%
     group_by(Sex) %>%
     summarise(Mass_mean = mean(MASS),
               Mass_sd = sd(MASS))
```

ID	SEX	MASS
A1	M	3.4
A2	F	6.7
…	…	…
A20	M	3.8

Sex	Mass_mean	Mass_sd
F	5.90	0.55
M	3.95	0.29

Join

Merging data frames.

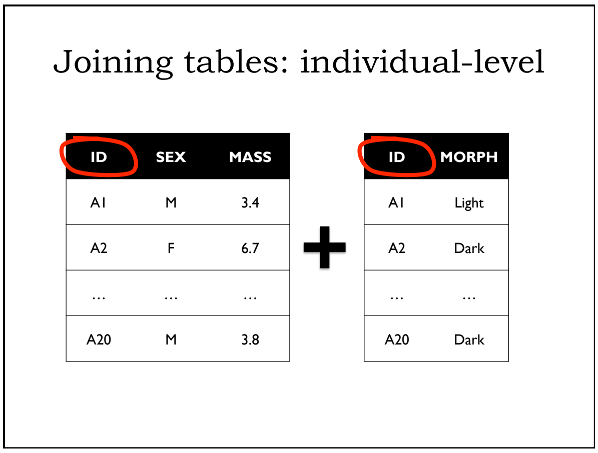


Joining tables: individual-level

ID	SEX	MASS
A1	M	3.4
A2	F	6.7
…	…	…
A20	M	3.8

+

ID	MORPH
A1	Light
A2	Dark
…	…
A20	Dark

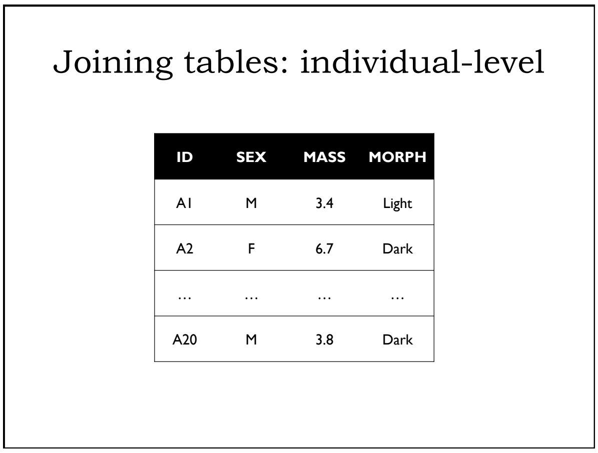


Joining tables: individual-level

ID	SEX	MASS
A1	M	3.4
A2	F	6.7
…	…	…
A20	M	3.8

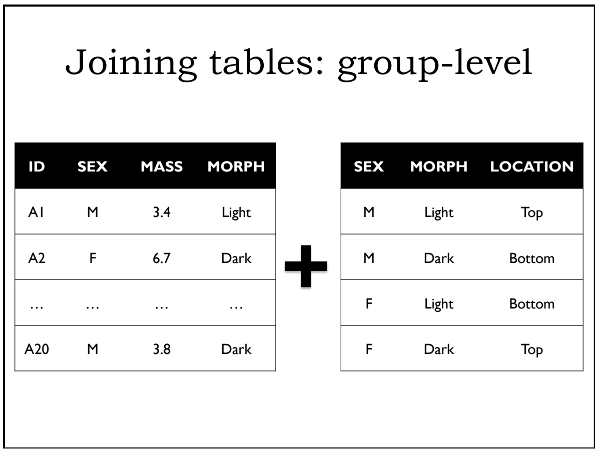
+

ID	MORPH
A1	Light
A2	Dark
…	…
A20	Dark



Joining tables: individual-level

ID	SEX	MASS	MORPH
A1	M	3.4	Light
A2	F	6.7	Dark
…	…	…	…
A20	M	3.8	Dark

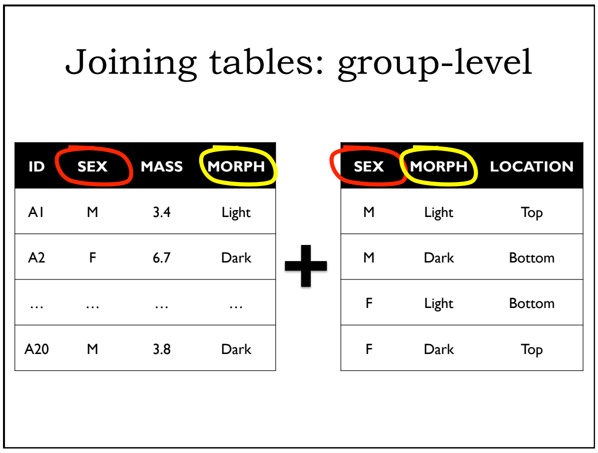


Joining tables: group-level

ID	SEX	MASS	MORPH
A1	M	3.4	Light
A2	F	6.7	Dark
…	…	…	…
A20	M	3.8	Dark

+

SEX	MORPH	LOCATION
M	Light	Top
M	Dark	Bottom
F	Light	Bottom
F	Dark	Top

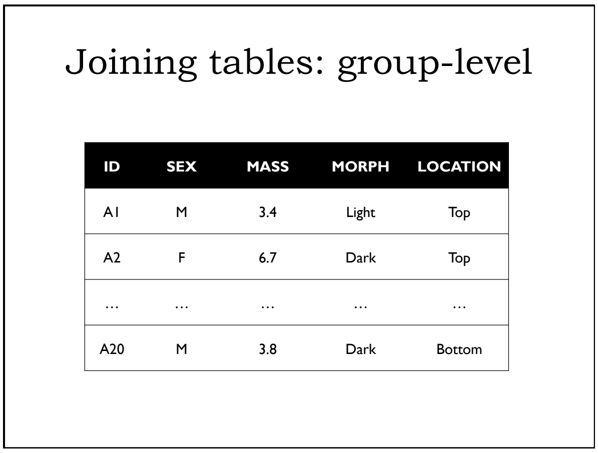


Joining tables: group-level

ID	SEX	MASS	MORPH
A1	M	3.4	Light
A2	F	6.7	Dark
…	…	…	…
A20	M	3.8	Dark

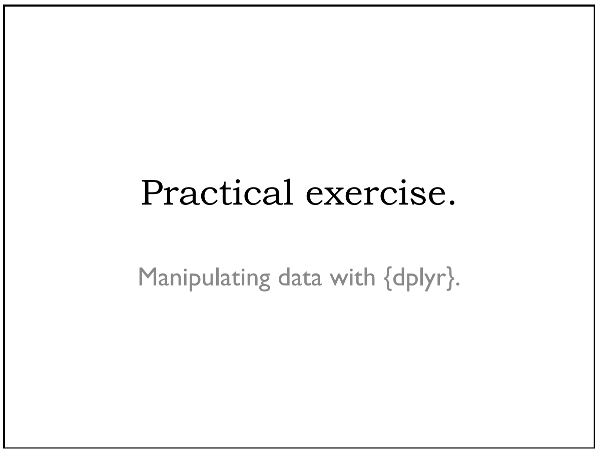
+

SEX	MORPH	LOCATION
M	Light	Top
M	Dark	Bottom
F	Light	Bottom
F	Dark	Top



Joining tables: group-level

ID	SEX	MASS	MORPH	LOCATION
A1	M	3.4	Light	Top
A2	F	6.7	Dark	Top
…	…	…	…	…
A20	M	3.8	Dark	Bottom



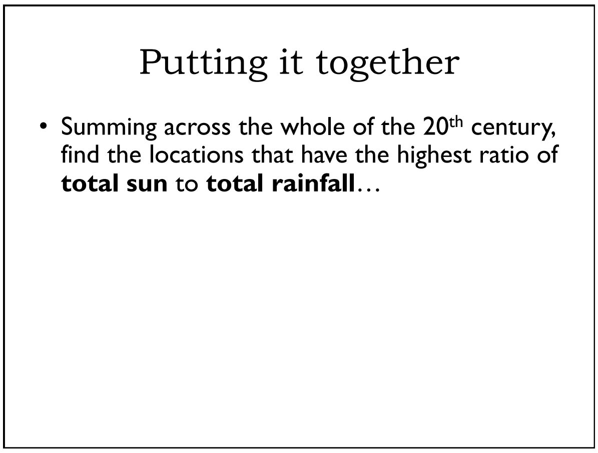
Practical exercise.

Manipulating data with {dplyr}.

Tidying & manipulating data

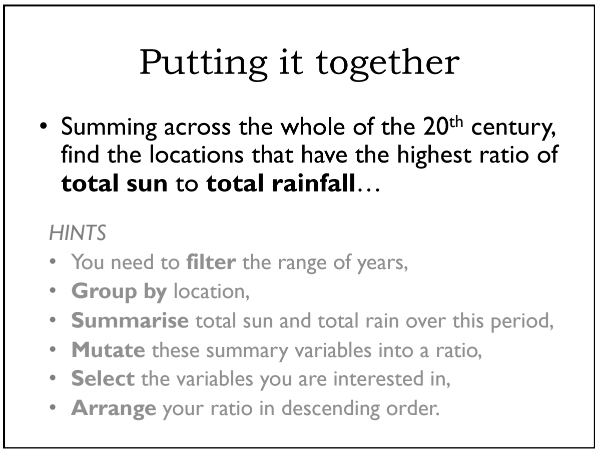
- Tidy some 'messy' data
- Interrogate historical data from UK weather stations

© 2014 TomHouslay.com



Putting it together

- Summing across the whole of the 20th century, find the locations that have the highest ratio of **total sun** to **total rainfall**…

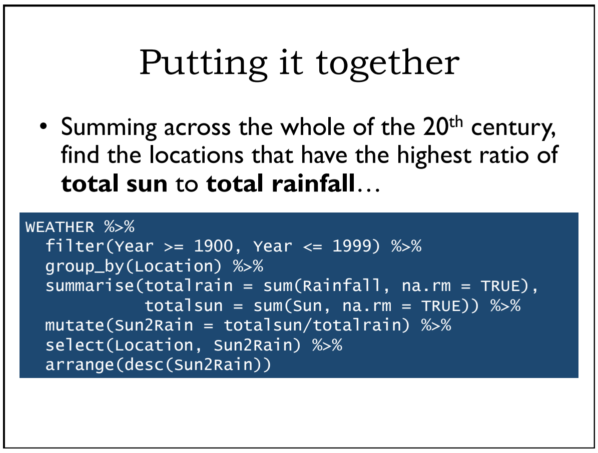


Putting it together

- Summing across the whole of the 20th century, find the locations that have the highest ratio of **total sun** to **total rainfall**…

HINTS

- You need to **filter** the range of years,
- **Group by** location,
- **Summarise** total sun and total rain over this period,
- **Mutate** these summary variables into a ratio,
- **Select** the variables you are interested in,
- **Arrange** your ratio in descending order.



Putting it together

- Summing across the whole of the 20th century, find the locations that have the highest ratio of **total sun** to **total rainfall**…

```
WEATHER %>%
  filter(Year >= 1900, Year <= 1999) %>%
  group_by(Location) %>%
  summarise(totalrain = sum(Rainfall, na.rm = TRUE),
            totalsun = sum(Sun, na.rm = TRUE)) %>%
  mutate(Sun2Rain = totalsun/totalrain) %>%
  select(Location, Sun2Rain) %>%
  arrange(desc(Sun2Rain))
```

{end}

You can now manipulate data.