An introduction to analysing data

Owen Ward

2020-10-22

Where to get data?

- If we want to analyse data in `R`, we need to find a way to read it and create a `data.frame`, or object we can perform calculations on.

Where to get data?

- If we want to analyse data in `R`, we need to find a way to read it and create a `data.frame`, or object we can perform calculations on.
- Several ways of getting `data`.

Where to get data?

- If we want to analyse data in `R`, we need to find a way to read it and create a `data.frame`, or object we can perform calculations on.
- Several ways of getting data.
- Included in `R`, or in an `R` package.

Where to get data?

- If we want to analyse data in `R`, we need to find a way to read it and create a `data.frame`, or object we can perform calculations on.
- Several ways of getting `data`.
- Included in `R`, or in an `R` package.
- Read a text/csv/etc file.

Where to get data?

- If we want to analyse data in `R`, we need to find a way to read it and create a `data.frame`, or object we can perform calculations on.
- Several ways of getting data.
- Included in `R`, or in an `R` package.
- Read a text/csv/etc file.
- Scrape it from a website/API.

Data in base R

- Several datasets are included when you install R.
- Can see these by running `data()`.

```r
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Spec
## 1          5.1         3.5          1.4         0.2  set
## 2          4.9         3.0          1.4         0.2  set
## 3          4.7         3.2          1.3         0.2  set
## 4          4.6         3.1          1.5         0.2  set
## 5          5.0         3.6          1.4         0.2  set
## 6          5.4         3.9          1.7         0.4  set
```

```r
class(iris)
```

```
## [1] "data.frame"
```

Data in packages

- Similarly, can load a package and the data.

```r
library(palmerpenguins)
# look at data() now
head(penguins)
```

```
## # A tibble: 6 x 8
##   species island bill_length_mm bill_depth_mm flipper_le
##   <fct>   <fct>           <dbl>         <dbl>
## 1 Adelie  Torge~           39.1          18.7
## 2 Adelie  Torge~           39.5          17.4
## 3 Adelie  Torge~           40.3          18
## 4 Adelie  Torge~           NA            NA
## 5 Adelie  Torge~           36.7          19.3
## 6 Adelie  Torge~           39.3          20.6
## # ... with 3 more variables: body_mass_g <int>, sex <fct
## #   year <int>
```

Data in packages

```r
class(penguins)
```

```
## [1] "tbl_df"     "tbl"        "data.frame"
```

Reading in files

- Lots of built in functions to read in common file formats, we will see some of these later.
- Similarly, can extract data from raw html code on the web.

Tibbles

- Tibbles are a slightly more modern form of data frames, part of a collection of packages called the `tidyverse` which are designed for data science.
- Will use these tools as much as possible.

```
library(tidyverse)
```

Types of Variables

When we viewed the tibble above, we saw several different types of random variables.

- `fct`, for categorical variables.
- `int` for integer for integer valued variables.
- `dbl`, for continuous valued variables.

There are also many others, such as `chr`, `lgl`, `dttm` and `date`. Having a variable in an informative format can make data analysis easier, can use existing tools.

Manipulating data

- Several common tasks want to do when analysing a data set.

Manipulating data

- Several common tasks want to do when analysing a data set.
- Look at a specific subset of interest.

Manipulating data

- Several common tasks want to do when analysing a data set.
- Look at a specific subset of interest.
- Select a specific variable to look at

Manipulating data

- Several common tasks want to do when analysing a data set.
- Look at a specific subset of interest.
- Select a specific variable to look at
- Extract new information from an existing variable.

Manipulating data

- Several common tasks want to do when analysing a data set.
- Look at a specific subset of interest.
- Select a specific variable to look at
- Extract new information from an existing variable.
- Compare quantities across groups.

Manipulating data

- Several common tasks want to do when analysing a data set.
- Look at a specific subset of interest.
- Select a specific variable to look at
- Extract new information from an existing variable.
- Compare quantities across groups.
- Will see tools to do all of these.

The pipe

- When we want to perform multiple steps like this, the pipe command, %>%, is a useful tool for combining them.
- Can think `x %>% f(y)` as "piping" `x` into `f`, equivalent to `f(x,y)`.
- Will show this more carefully below.

Filtering Data

- Useful for looking at a subset over one or more variables.

```
head(penguins$island)
```

```
## [1] Torgersen Torgersen Torgersen Torgersen Torgersen To
## Levels: Biscoe Dream Torgersen
```

Filtering Data

```
penguins %>% filter(island == "Biscoe") %>% head()
```

```
## # A tibble: 6 x 8
##   species island bill_length_mm bill_depth_mm flipper_le
##   <fct>   <fct>           <dbl>         <dbl>
## 1 Adelie  Biscoe           37.8          18.3
## 2 Adelie  Biscoe           37.7          18.7
## 3 Adelie  Biscoe           35.9          19.2
## 4 Adelie  Biscoe           38.2          18.1
## 5 Adelie  Biscoe           38.8          17.2
## 6 Adelie  Biscoe           35.3          18.9
## # ... with 3 more variables: body_mass_g <int>, sex <fct
## #   year <int>
```

```
# could have also done
filter(penguins, island == "Biscoe")
```

Selecting a variable

```
penguins %>% select(body_mass_g,year) %>% head()
```

```
## # A tibble: 6 x 2
##   body_mass_g  year
##         <int> <int>
## 1        3750  2007
## 2        3800  2007
## 3        3250  2007
## 4          NA  2007
## 5        3450  2007
## 6        3650  2007
```

Selecting a variable

```
penguins %>% select(-species) %>% head()
```

```
## # A tibble: 6 x 7
##   island bill_length_mm bill_depth_mm flipper_length_~
##   <fct>           <dbl>         <dbl>            <int>
## 1 Torge~           39.1          18.7              181
## 2 Torge~           39.5          17.4              186
## 3 Torge~           40.3          18                195
## 4 Torge~           NA            NA                 NA
## 5 Torge~           36.7          19.3              193
## 6 Torge~           39.3          20.6              190
## # ... with 3 more variables: body_mass_g <int>, sex <fct
## #   year <int>
```

Selecting a variable

```
head( select(penguins, body_mass_g, year) )
```

```
## # A tibble: 6 x 2
##   body_mass_g  year
##         <int> <int>
## 1        3750  2007
## 2        3800  2007
## 3        3250  2007
## 4          NA  2007
## 5        3450  2007
## 6        3650  2007
```

Selecting a variable

```
head( select(penguins, - species))
```

```
## # A tibble: 6 x 7
##   island bill_length_mm bill_depth_mm flipper_length_~
##   <fct>           <dbl>         <dbl>            <int>
## 1 Torge~           39.1          18.7              181
## 2 Torge~           39.5          17.4              186
## 3 Torge~           40.3          18                195
## 4 Torge~           NA            NA                 NA
## 5 Torge~           36.7          19.3              193
## 6 Torge~           39.3          20.6              190
## # ... with 3 more variables: body_mass_g <int>, sex <fct
## #   year <int>
```

Mutating a variable

- Can perform some calculations on a variable, add two variables, etc.

```
penguins %>%
  mutate(body_mass_oz = body_mass_g/28.35) %>%
  select(body_mass_g:body_mass_oz) %>%
  head()
```

```
## # A tibble: 6 x 4
##   body_mass_g sex     year body_mass_oz
##         <int> <fct>  <int>        <dbl>
## 1        3750 male    2007         132.
## 2        3800 female  2007         134.
## 3        3250 female  2007         115.
## 4          NA <NA>    2007          NA
## 5        3450 female  2007         122.
## 6        3650 male    2007         129.
```

Compare across subgroups

- Can easily compare across some subgroups based on one or more variable.

```
penguins %>%
  group_by(species) %>%
  count()
```

```
## # A tibble: 3 x 2
## # Groups:   species [3]
##   species       n
##   <fct>     <int>
## 1 Adelie      152
## 2 Chinstrap    68
## 3 Gentoo      124
```

Compare across subgroups

```
penguins %>%
  group_by(species, island) %>%
  count()
```

```
## # A tibble: 5 x 3
## # Groups:   species, island [5]
##   species   island        n
##   <fct>     <fct>     <int>
## 1 Adelie    Biscoe       44
## 2 Adelie    Dream        56
## 3 Adelie    Torgersen    52
## 4 Chinstrap Dream        68
## 5 Gentoo    Biscoe      124
```

Compare across subgroups

- Often use this together with the summarise command.

```
penguins %>%
  group_by(species) %>%
  summarise( num_peng = n(), ave_mass = mean(body_mass_g, r
```

```
## `summarise()` ungrouping output (override with `.groups`

## # A tibble: 3 x 3
##   species   num_peng ave_mass
##   <fct>        <int>    <dbl>
## 1 Adelie         152    3701.
## 2 Chinstrap       68    3733.
## 3 Gentoo         124    5076.
```

Putting these together

- The real power of the pipe is complex commands combining multiple functions.
- Allows us to do this in a clear way.
- For example, if we wanted to look at the distribution of small penguins by island, species and sex.

Putting these together

```
penguins %>% filter(body_mass_g < 3700) %>%
  group_by(species,island,sex) %>% count()
```

```
## # A tibble: 10 x 4
## # Groups:   species, island, sex [10]
##    species   island    sex        n
##    <fct>     <fct>     <fct>  <int>
##  1 Adelie    Biscoe    female    16
##  2 Adelie    Biscoe    male       3
##  3 Adelie    Dream     female    25
##  4 Adelie    Dream     male       4
##  5 Adelie    Dream     <NA>       1
##  6 Adelie    Torgersen female    19
##  7 Adelie    Torgersen male       4
##  8 Adelie    Torgersen <NA>       2
##  9 Chinstrap Dream     female    25
## 10 Chinstrap Dream     male       7
```

Visualising Data

Plotting Data

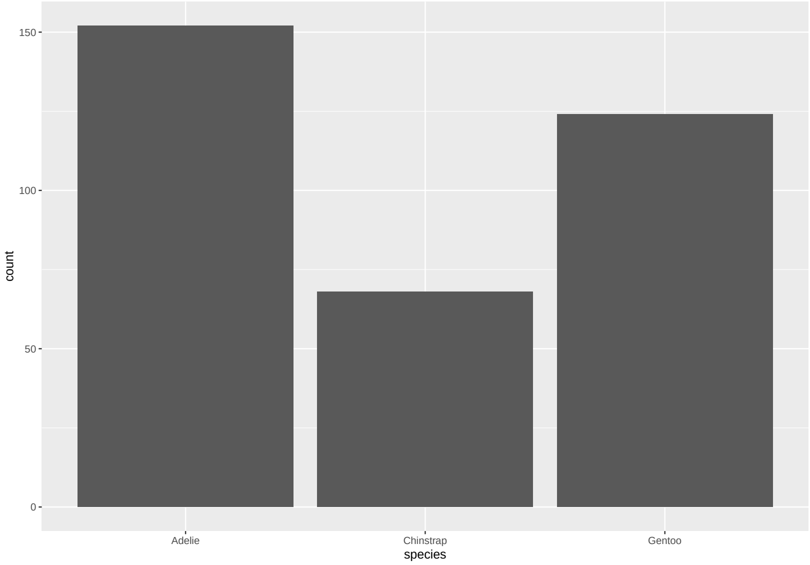
- Visualising data allows us to better understand the overall properties of the data.
- Captures the variation of values seen for a specific variable.
- May indicate interesting relationships between variables.

Categorical Data

- The simplest such case is to summarise a categorical variable, which we can do with a bar chart.

```
ggplot(data = penguins) +
  geom_bar(mapping = aes(x = species))
```

Bar Chart



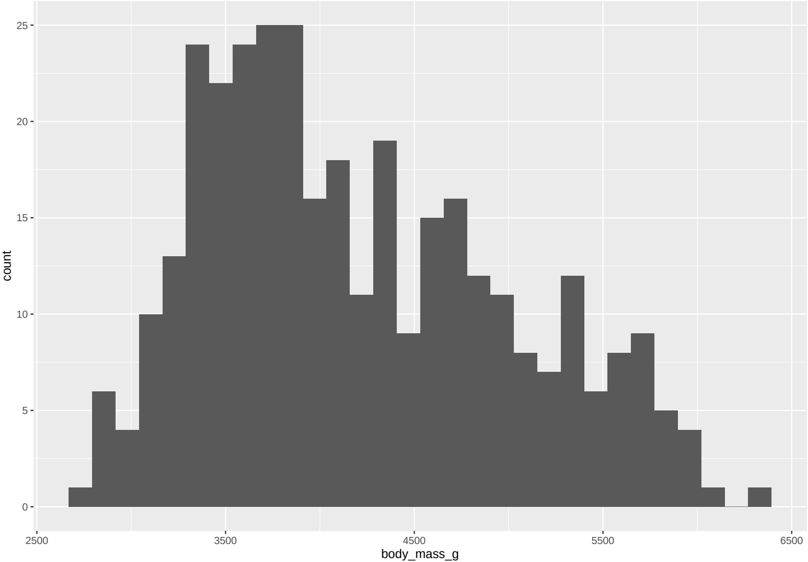
Continuous Data, one variable

- For a single continuous variable generally most informative plot is a histogram.
- Gives an sense of the spread of the variable, range of values it can take.

```
penguins %>% ggplot(aes(body_mass_g)) +
  geom_histogram()
```

Histogram

```
## `stat_bin()` using `bins = 30`. Pick better value with
## `binwidth`.
```

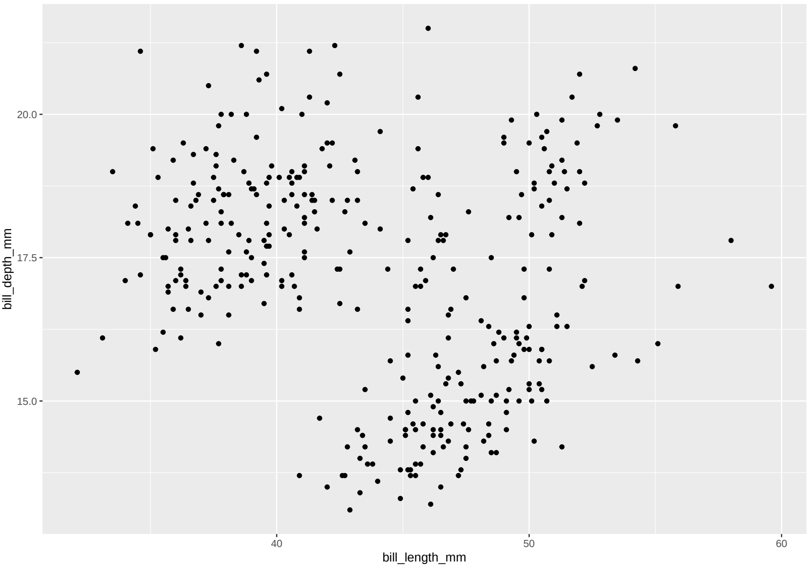


Continuous Data, two variables

- To look at relationship between two continuous variables can construct a scatter plot.

```
penguins %>% ggplot(aes(bill_length_mm,bill_depth_mm)) +
  geom_point()
```

Scatter Plot



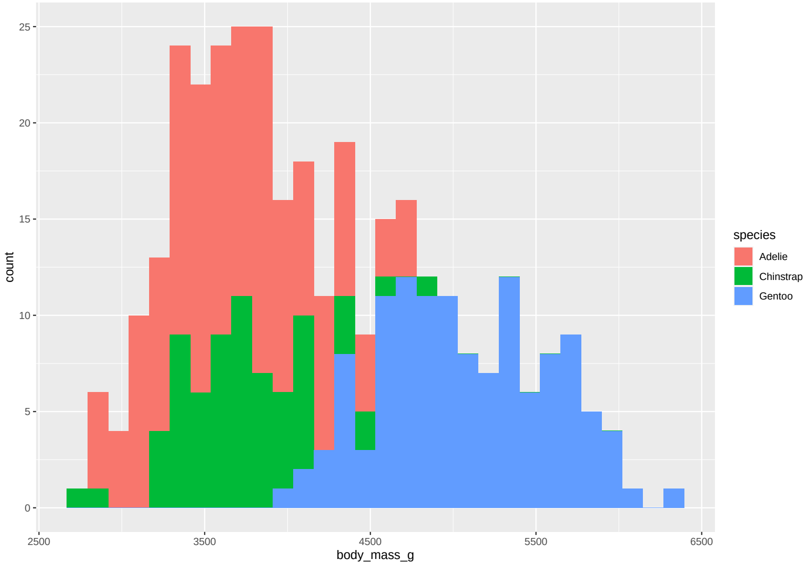
Continuous and Categorical

- If we want to look at the distribution of a continuous variable for different values of a categorical variable multiple options.
- Different plot for each value of the categorical variable.
- Use one plot, different colour for each value of the categorical variable.

```
penguins %>% ggplot(aes(body_mass_g, fill = species)) +
  geom_histogram()
## or
penguins %>% ggplot(aes(body_mass_g)) +
  geom_histogram() +
  facet_wrap(~species)
```

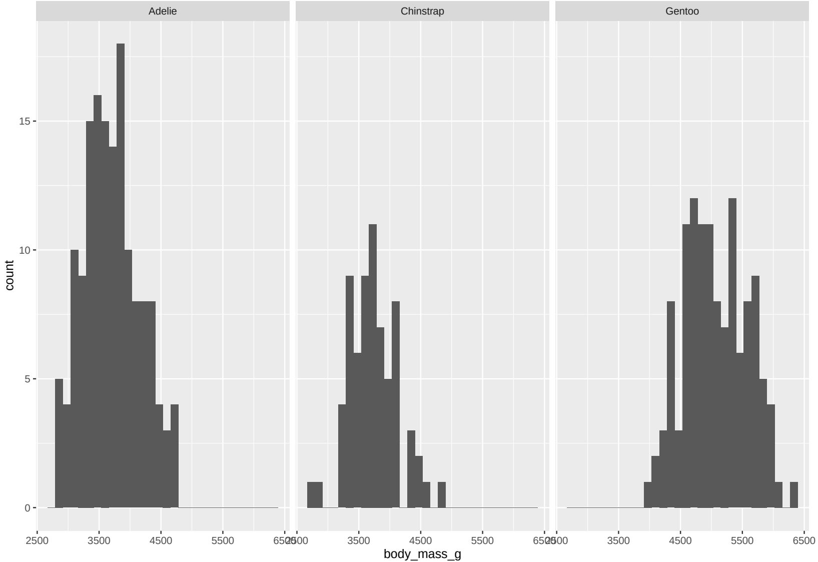
Single Plot with Colour

```
## `stat_bin()` using `bins = 30`. Pick better value with
## `binwidth`.
```



Multiple plots

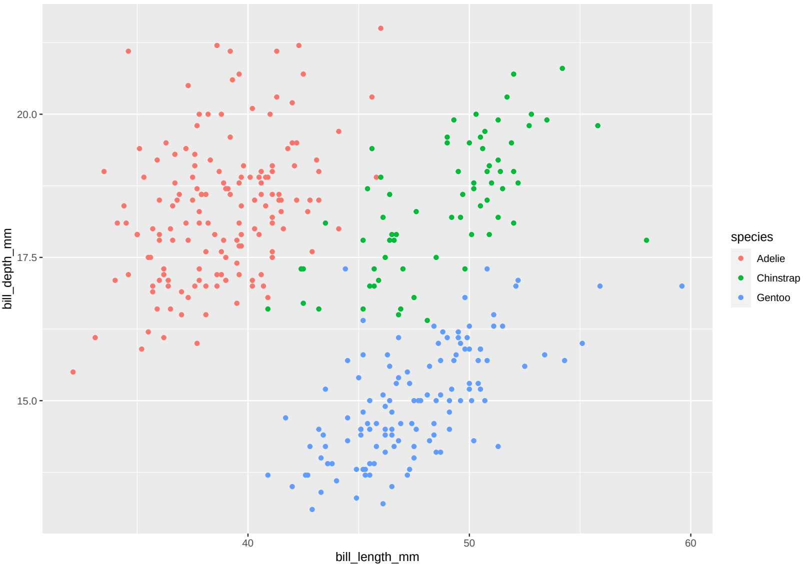
```
## `stat_bin()` using `bins = 30`. Pick better value with
## `binwidth`.
```



Scatter Plot with a categorical variable

```
penguins %>%
  ggplot(aes(bill_length_mm,bill_depth_mm,colour=species))
  geom_point()
## or
penguins %>% ggplot(aes(bill_length_mm,bill_depth_mm)) +
  geom_point() +
  facet_wrap(~species)
```

Different Colour



Multiple Plots