Tidy data wrangling

Athanasia Monika Mowinckel

Sept. 15th 2020

Part 1

Tidy data wrangling

Tidy data wrangling

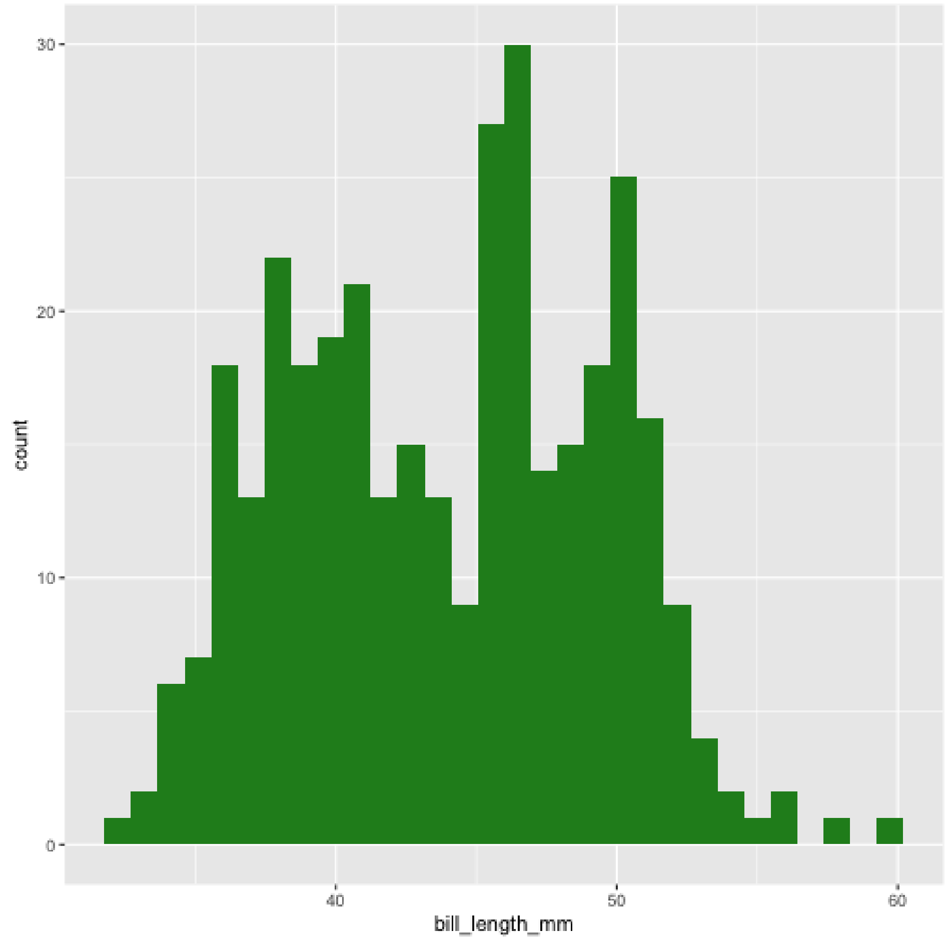
- plotting data with ggplot2 (~25 min)
- sub-setting data with dplyr (~25 min)
- chaining commands with the pipe `%>%` (~10 min)
- adding and altering variables with dplyr (~25 min)

ggplot2

grammar of graphics

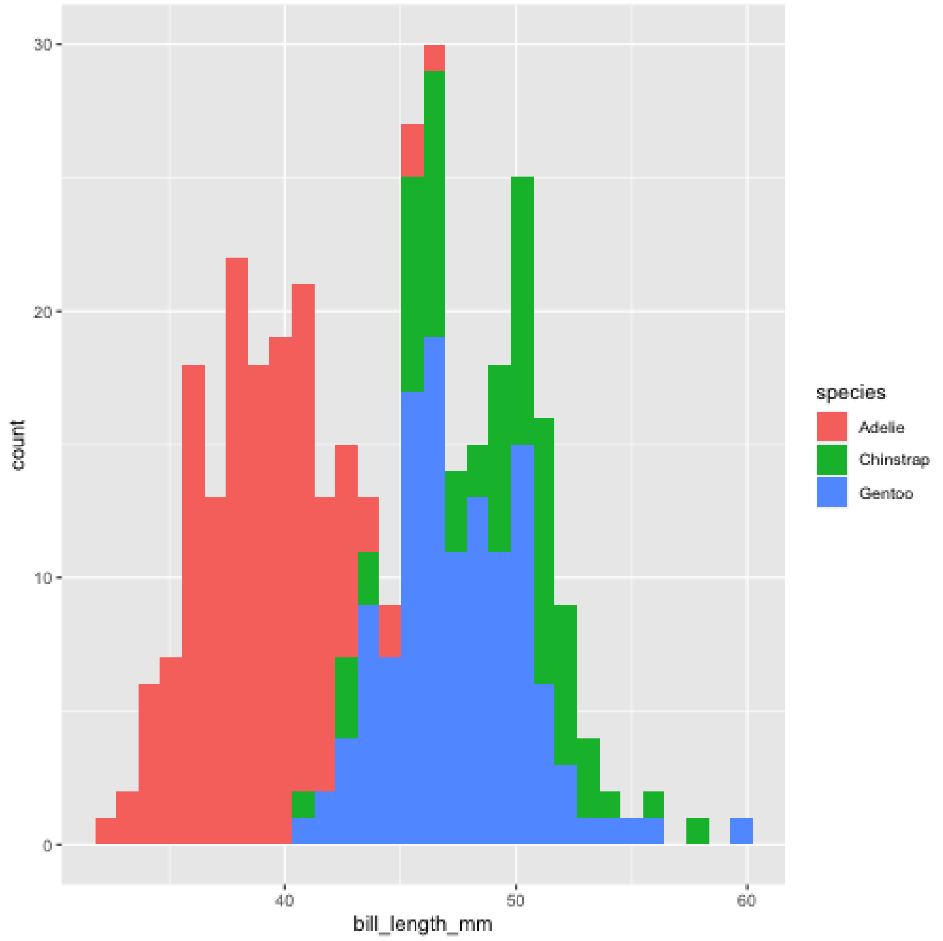
ggplot2 setting

```
ggplot(data = penguins,
       mapping = aes(x = bill_length_mm)) +
  geom_histogram(
    fill = "forestgreen"
  )
```



ggplot2 mapping

```
ggplot(data = penguins,
       mapping = aes(x = bill_length_mm,
                     fill = species)) +
  geom_histogram( )
```



Go to RStudio

live demo

LCBC LIFESPAN CHANGES *in brain and cognition*

Go to plotting exercises

```
learnr::run_tutorial("001-plotting",
"tidyquintro")
```

08:00

dplyr

data subsetting

dplyr

grammar of data manipulation

provides a consistent set of verbs that help you solve the most common data manipulation challenges:

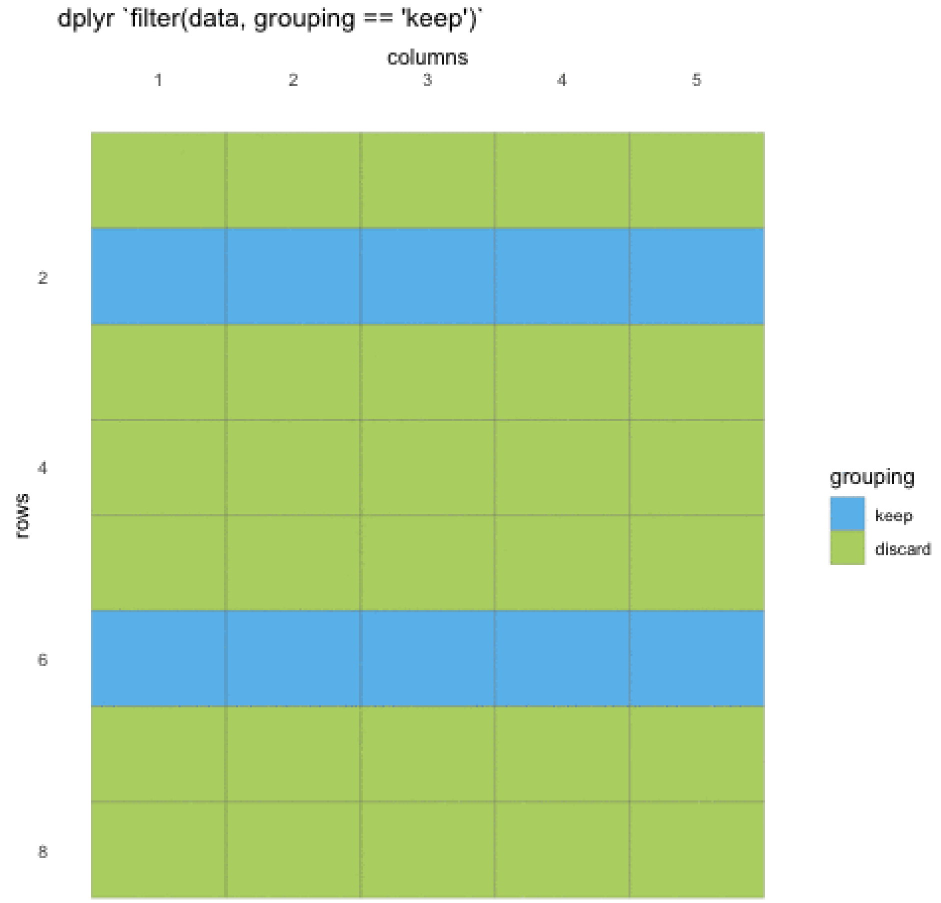
`select()` picks variables based on their names.
`filter()` picks cases based on their values.
`mutate()` - adds or alters variables that are functions of existing variables
`summarise()` reduces multiple values down to a single summary.
`arrange()` changes the ordering of the rows.

dplyr

`filter()` - subsetting rows

Reducing the number of rows in a data sat based on some logic.

- `filter()` evaluates a statement to be logical (`TRUE` or `FALSE`)



dplyr - comparison to base-R

tidy

```
filter(penguins, bill_length_mm > 40)
```

base

```
penguins[penguins$bill_length_mm > 40, ]

# or
subset(penguins, bill_length_mm > 40)
```

https://dplyr.tidyverse.org/articles/base.html

Go to RStudio

live demo

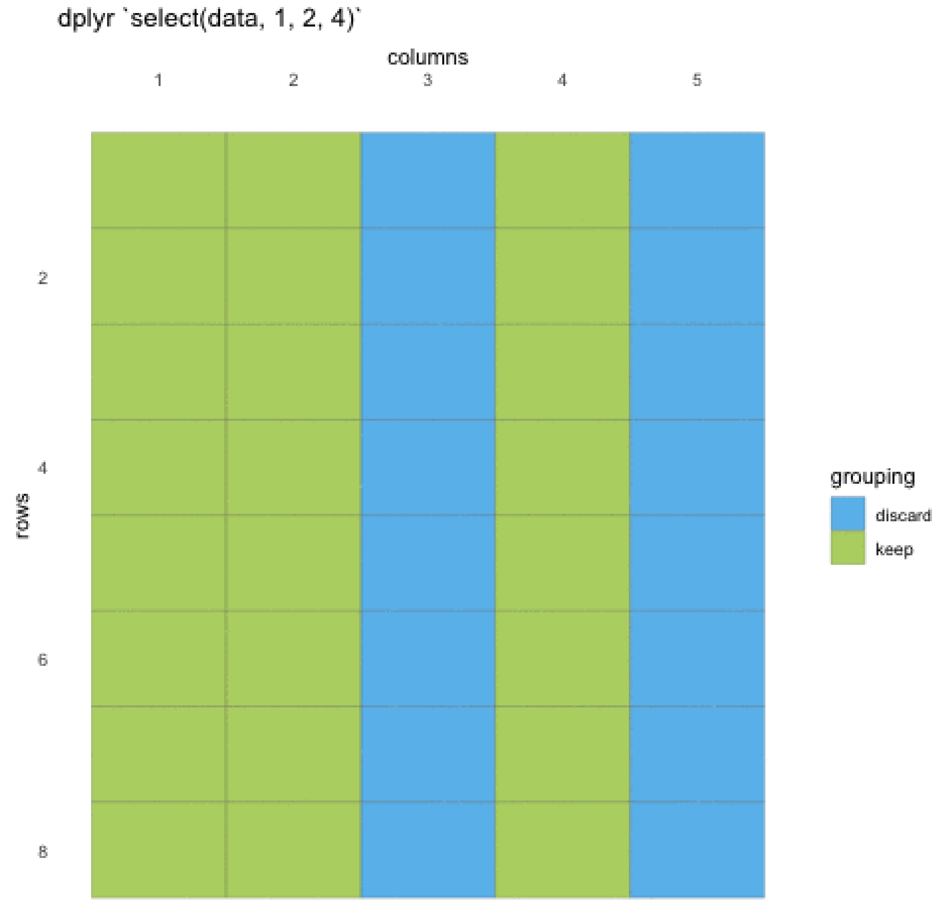
dplyr

`select()` - reduce columns

Reducing the number of columns (or rearranging columns) Can be used with column names, index integer, or tidyselect-functions

tidy-select helpers

- `ends_with("string")` - column names ending with "string"
- `starts_with("string")` - column names starting with "string"
- `contains("string")` - column names containing "string"



dplyr - comparison to base-R

tidy

```r
select(penguins, species, island, ends_with("mm"))
```

base

```r
penguins[c(1, 2, grep("mm$", names(penguins)))]

# or
subset(penguins, select = c("species", "island", "bill_length_mm", "bill_depth_mm", "flipper_
```

https://dplyr.tidyverse.org/articles/base.html

LCBC LIFESPAN CHANGES *in brain and cognition*

Go to RStudio

live demo

Go to subsetting exercises

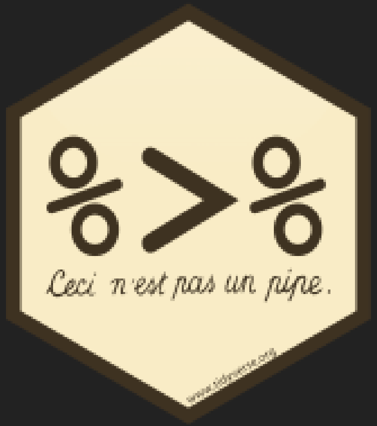
```
learnr::run_tutorial("002-subsetting",
"tidyquintro")
```

08:00

LCBC LIFESPAN CHANGES in brain and cognition

magrittr

the pipe - chaining commands



the pipe - chaining commands

- Common to many programming languages
  - sending the output from one function, straight into another, without saving the intermediary
- Only really work when input is the *first* command to a function
  - This is not the case for most base-R functions, but is *always* the case with tidyverse functions
- The common used pipe in R, `%>%`, originally comes from the magrittr package, but also comes with dplyr

Use

```r
# standard
select(penguins,
       species, island, ends_with("mm"))
```

```r
# piped
penguins %>%
  select(species, island, ends_with("mm"))
```

```
## # A tibble: 344 x 5
##    species island    bill_length_mm bill_depth_mm
##    <fct>   <fct>              <dbl>         <dbl>
##  1 Adelie  Torgersen           39.1          18.7
##  2 Adelie  Torgersen           39.5          17.4
##  3 Adelie  Torgersen           40.3          18
##  4 Adelie  Torgersen           NA            NA
##  5 Adelie  Torgersen           36.7          19.3
##  6 Adelie  Torgersen           39.3          20.6
##  7 Adelie  Torgersen           38.9          17.8
##  8 Adelie  Torgersen           39.2          19.6
##  9 Adelie  Torgersen           34.1          18.1
## 10 Adelie  Torgersen           42            20.2
## # … with 334 more rows
```

Go to RStudio

live demo

Go to chaining exercises

```
learnr::run_tutorial("003-chaining",
"tidyquintro")
```

08:00

dplyr

data wrangling / manipulation

dplyr

grammar of data manipulation

provides a consistent set of verbs that help you solve the most common data manipulation challenges:

`select()` picks variables based on their names.
`filter()` picks cases based on their values.

`mutate()` - adds or alters variables that are functions of existing variables
`summarise()` reduces multiple values down to a single summary.
`arrange()` changes the ordering of the rows.

dplyr - comparison to base-R

tidy

```r
penguins %>%
  mutate(
    new_column = 1,
    bill_ld_ratio = bill_length_mm/bill_depth_mm
  )
```

base

```r
penguins$new_column <- 1
penguins$bill_ld_ratio <- penguins$bill_length_mm/penguins$bill_depth_mm
```

https://dplyr.tidyverse.org/articles/base.html

Go to RStudio

live demo

Go to mutating exercises

```
learnr::run_tutorial("004-mutating",
"tidyquintro")
```

08:00

LCBC LIFESPAN CHANGES *in brain and cognition*

End of part 1

30 minute lunch break

30:00