

LONDON
SCHOOL *of*
HYGIENE
& TROPICAL
MEDICINE



Exploratory Data Analysis

~~2491 - Data Challenge~~

2031 - Introduction to Statistical Computation

Sam Clifford

~~Session 3 2022-01-13~~

2022-11-18

Introduction

Tidyverse

- Suite of R packages to make working with data as easy as possible (Wickham 2020), including
 - `ggplot2`: for plotting data
 - `dplyr`: for manipulating data frames
 - `tidyverse`: for making data tidy
 - `forcats`: for manipulating factor variables
 - `magrittr`: for easy chaining of commands

(Wickham and Grolemund 2017; Wickham et al. 2019)



Summarising data

Summarising data

- Summary statistics one of most common data analysis tasks
- Consider the `Gestation` data from `mosaicData`

Birth weight, date, and gestational period collected as part of the Child Health and Development Studies in 1961 and 1962. Information about the baby's parents — age, education, height, weight, and whether the mother smoked is also recorded (Nolan and Speed 2001).

- We will use some functions from `dplyr` to choose, group and summarise data
- `verb(.data, ...)` applies a `dplyr` `verb` to a data frame



Summarising data

- count how many babies in data set

```
library(mosaicData)
library(tidyverse)
data(Gestation)

count(Gestation) # just like nrow()
## # A tibble: 1 × 1
##       n
##   <int>
## 1 1236
```



Summarising data

- Can also count for a given grouping variable

```
count(Gestation, race) # nrow can't do this
## # A tibble: 6 × 2
##   race      n
##   <chr> <int>
## 1 asian     44
## 2 black    244
## 3 mex      40
## 4 mixed     25
## 5 white   870
## 6 <NA>     13
```



Summarising data

- The `summarise` function allows us to calculate summary statistics of a variable
- Can (and should) give names to summary columns
- Calculate the mean birth weight in the data set

```
summarise(Gestation, wt_mean = mean(wt))  
## # A tibble: 1 × 1  
##   wt_mean  
##       <dbl>  
## 1     120.
```



Summarising data

- We can calculate multiple summaries at once

```
summarise(Gestation,
  Mean = mean(wt),
  SD   = sd(wt),
  Low  = quantile(wt, 0.025),
  High = quantile(wt, 0.975))
## # A tibble: 1 × 4
##   Mean     SD    Low   High
##   <dbl> <dbl> <dbl> <dbl>
## 1 120.   18.2    81    155
```



Summarising grouped data

- We can group the rows in our data and calculate summaries for each group
- `group_by` lets us pass variable names to set the structure
- Row order is maintained

```
Gestation_grouped_by_race <- group_by(Gestation, race)

Gestation_grouped_by_race
## # A tibble: 1,236 × 23
## # Groups:   race [6]
##       id plurality outcome date      gestation sex     wt parity race    age
##   <dbl> <chr>     <chr>   <date>    <dbl> <chr> <dbl> <dbl> <chr> <dbl>
## 1     15 single   fe... live bi... 1964-11-11     284 male   120     1 asian   27
## 2     20 single   fe... live bi... 1965-02-07     282 male   113     2 white   33
## 3     58 single   fe... live bi... 1965-04-25     279 male   128     1 white   28
## 4     61 single   fe... live bi... 1965-02-12      NA male   123     2 white   36
## 5     72 single   fe... live bi... 1964-11-25     282 male   108     1 white   23
## 6    100 single   fe... live bi... 1965-07-31     286 male   136     4 white   25
## 7    102 single   fe... live bi... 1964-12-19     244 male   138     4 black   33
## 8    129 single   fe... live bi... 1965-04-11     245 male   132     2 black   23
## 9    142 single   fe... live bi... 1964-11-08     289 male   120     3 white   25
## 10   148 single   fe... live bi... 1965-04-17     299 male   143     3 white   30
## # ... with 1,226 more rows, and 13 more variables: ed <chr>, ht <dbl>,
## #   wt.1 <dbl>, drace <chr>, dage <dbl>, ded <chr>, dht <dbl>, dwt <dbl>,
## #   marital <chr>, inc <chr>, smoke <chr>, time <chr>, number <chr>
```



Summarising grouped data

- `summarise()` respects the grouping structure

```
summarise(Gestation_grouped_by_race,
  Mean = mean(wt),
  SD   = sd(wt),
  Low  = quantile(wt, 0.025),
  High = quantile(wt, 0.975))
## # A tibble: 6 × 5
##   race    Mean     SD    Low   High
##   <chr>  <dbl>  <dbl> <dbl>  <dbl>
## 1 asian   110.   16.0  78.4  139.
## 2 black   113.   19.1  71     150  
## 3 mex     124.   14.1  99.0  146.
## 4 mixed   120.   20.1  78.8  150.
## 5 white   122.   17.7  85     158  
## 6 <NA>    117.   16.7  86.8  143.
```



Summarising data

- We can summarise multiple variables (`.vars`) with the same function(s) (`.funs`)

```
summarise_at(Gestation,
              .vars = vars(gestation, wt, age),
              .funs = mean, na.rm = T)
## # A tibble: 1 × 3
##   gestation     wt     age
##       <dbl>   <dbl>   <dbl>
## 1      279.    120.   27.3
```



Summarising data

- We can even do multiple summaries across multiple variables by specifying
 - which `.vars` we want to summarise, and
 - a list of which `.funs` we wish to summarise with

```
summarise_at(Gestation,
              .vars = vars(gestation, wt, age),
              .funs = list(mean = mean,
                           sd    = sd), na.rm = T)
## # A tibble: 1 × 6
##   gestation_mean wt_mean age_mean gestation_sd wt_sd age_sd
##             <dbl>   <dbl>    <dbl>      <dbl>   <dbl>    <dbl>
## 1           279.    120.     27.3      16.0    18.2     5.78
```



Summarising grouped data

- All `summarise*()` respects the grouping structure

```
summarise_at(Gestation_grouped_by_race,
              .vars = vars(gestation, wt, age),
              .funs = list(mean = mean), na.rm = T)
## # A tibble: 6 × 4
##   race   gestation_mean wt_mean age_mean
##   <chr>      <dbl>    <dbl>     <dbl>
## 1 asian       274.     110.      29.4
## 2 black       274.     113.      27.6
## 3 mex         278.     124.      26.5
## 4 mixed       279.     120.      26.4
## 5 white       281.     122.      27.1
## 6 <NA>        280      117.      30.7
```



Rows and columns

Operating on columns

- create/modify/delete columns with dplyr's `mutate()`
- e.g. relabelling `race` so words start with a capital,

```
Gestation <- mutate(Gestation, race = str_to_title(race))
```

```
count(Gestation, race)
## # A tibble: 6 × 2
##   race      n
##   <chr> <int>
## 1 Asian     44
## 2 Black    244
## 3 Mex       40
## 4 Mixed     25
## 5 White    870
## 6 <NA>      13
```



Choosing columns

- For one reason or another we may want to `select` only certain columns of our data frame

```
head(Gestation, 1)
## # A tibble: 1 × 23
##   id plurality    outcome date      gestation sex      wt parity race     age
##   <dbl> <chr>       <chr>   <date>    <dbl> <chr> <dbl> <dbl> <chr> <dbl>
## 1 15 single fetus live b... 1964-11-11     284 male     120     1 Asian     27
## # ... with 13 more variables: ed <chr>, ht <dbl>, wt.1 <dbl>, drace <chr>,
## #   dage <dbl>, ded <chr>, dht <dbl>, dwt <dbl>, marital <chr>, inc <chr>,
## #   smoke <chr>, time <chr>, number <chr>

head(select(Gestation, race, wt, number), 1)
## # A tibble: 1 × 3
##   race     wt number
##   <chr> <dbl> <chr>
## 1 Asian    120 never
```



Choosing and renaming columns

- We can also rename columns on the fly as we select them

```
select(Gestation,
       Race           = race,
       `Birthweight (oz)` = wt,
       `Cigs. smoked`   = number)
## # A tibble: 1,236 × 3
##       Race `Birthweight (oz)` `Cigs. smoked`
##     <chr>      <dbl>        <chr>
## 1 Asian         120 never
## 2 White         113 never
## 3 White         128 1-4 per day
## 4 White         123 20-29 per day
## 5 White         108 20-29 per day
## 6 White         136 5-9 per day
## 7 Black          138 never
## 8 Black          132 never
## 9 White          120 never
## 10 White         143 15-19 per day
## # ... with 1,226 more rows
```



Choosing and renaming columns

- Alternatively we can `rename` columns without worrying about failing to select columns we haven't renamed

```
names(Gestation)
## [1] "id"          "pluralty"     "outcome"      "date"        "gestation"    "sex"
## [7] "wt"          "parity"       "race"         "age"         "ed"           "ht"
## [13] "wt.1"        "drace"        "dage"         "ded"         "dht"          "dwt"
## [19] "marital"     "inc"          "smoke"        "time"        "number"

Gestation <- rename(Gestation, `Cigs. smoked` = number)

names(Gestation)
## [1] "id"          "pluralty"     "outcome"      "date"        "gestation"
## [6] "sex"          "wt"          "parity"       "race"        "age"
## [11] "ed"          "ht"          "wt.1"        "drace"      "dage"
## [16] "ded"         "dht"         "dwt"         "marital"    "inc"
## [21] "smoke"       "time"        "Cigs. smoked"
```



Choosing rows

- The dplyr equivalent of `subset` is `filter`
- Takes a logical statement and does non-standard evaluation of variable names
- `filter(data, A & B)` the same as `filter(data, A, B)`

```
Gestation2 <- select(Gestation,
  Race = race,
  `Birthweight (oz)` = wt,
  `Cigs. smoked`)

filter(Gestation2, Race == "White", `Cigs. smoked` == "never")
## # A tibble: 352 × 3
##   Race `Birthweight (oz)` `Cigs. smoked`
##   <chr>      <dbl> <chr>
## 1 White       113   never
## 2 White       120   never
## 3 White       144   never
## 4 White       125   never
## 5 White       122   never
## 6 White       113   never
## 7 White       134   never
## 8 White       128   never
## 9 White       129   never
## 10 White      110   never
## # ... with 342 more rows
```



Choosing rows

- `slice*`() functions allow you to select rows based on their properties, e.g. which babies have lowest birth weight overall and in each race group?

```
slice_min(Gestation2, `Birthweight (oz)`)  
## # A tibble: 1 × 3  
##   Race   `Birthweight (oz)` `Cigs. smoked`  
##   <chr>          <dbl> <chr>  
## 1 Black           55 never  
  
slice_min(group_by(Gestation2, Race), `Birthweight (oz)`)  
## # A tibble: 6 × 3  
## # Groups:   Race [6]  
##   Race   `Birthweight (oz)` `Cigs. smoked`  
##   <chr>          <dbl> <chr>  
## 1 Asian            71 5-9 per day  
## 2 Black            55 never  
## 3 Mex              97 never  
## 4 Mixed            77 20-29 per day  
## 5 White            63 never  
## 6 <NA>             82 20-29 per day
```



Reshaping data frames

Long and wide tidy data

- Tidy **data frames** consist of a number of observations (rows) of variables (columns), they can be either **wide** or **long** (Wickham 2014)
- We can pivot between wide and long format with `tidyR's pivot_*` functions
- Pivoting to a longer data frame helps put data in *key-value* pairs, useful for `ggplot2`
- The value is the **value** of the **named** variable for a given **id**



Long and wide tidy data

To make this pivot, we specify

- which `cols` are to be converted from being k columns of length n to one column of length $n \times k$
- the `names_to` column name, which contains the names of the pivoted columns
- the `values_to` name of the column containing the `value` of each variable for each `id`



Long and wide tidy data

```
Gestation_igwa <- select(Gestation, id, gestation, wt, age)

Gestation_long <- pivot_longer(
  data              = Gestation_igwa,
  cols              = c(gestation, wt, age),
  names_to         = 'name',
  values_to        = 'value')

head(Gestation_long, 6)
## # A tibble: 6 × 3
##       id name      value
##   <dbl> <chr>    <dbl>
## 1     15 gestation 284
## 2     15 wt        120
## 3     15 age        27
## 4     20 gestation 282
## 5     20 wt        113
## 6     20 age        33
```

- NB: we need to use ' quotes for `names_to` and `values_to` arguments because they are strings defining new columns



Long and wide tidy data

Or specify which columns *not* to pivot, e.g. `-id` selects all variables except `id`

```
Gestation_long <- pivot_longer(Gestation_igwa, -id)

head(Gestation_long, 6)
## # A tibble: 6 × 3
##       id   name    value
##   <dbl> <chr>    <dbl>
## 1     15 gestation 284
## 2     15 wt        120
## 3     15 age        27
## 4     20 gestation 282
## 5     20 wt        113
## 6     20 age        33
```



Long and wide tidy data

- To convert to a wider format, we use `pivot_wider`
- For example, we specify:
 - the *data* source
 - where we get the new column *names from*
 - where we get the new column *values from*

```
Gestation_wide <- pivot_wider(data      = Gestation_long,
                                names_from = name,
                                values_from = value)

head(Gestation_wide, 3) # recovered original data frame
## # A tibble: 3 × 4
##       id gestation     wt     age
##   <dbl>    <dbl> <dbl>   <dbl>
## 1     15     284    120     27
## 2     20     282    113     33
## 3     58     279    128     28
```



Pipe (Optional)

Pipe

- dplyr imports the `%>%` pipe from magrittr
- `f(g(x))` is equivalent to `x %>% g %>% f`
- Makes it easier to chain operations together without storing temporary objects
- Output on left of `%>%` becomes first argument of function on right
 - by convention, all tidyverse functions take a data frame as their first argument

```
x %>% f_1 %>% f_2 %>% f_3

# rather than
f_3(f_2(f_1(x)))

# or even worse...
x_1 <- f_1(x)
x_2 <- f_2(x_1)
x_3 <- f_3(x_2)
```



Pipe

- An example

```
Gestation %>% group_by(race, smoke) %>%  
  summarise(wt = mean(wt)) %>% pivot_wider(names_from = race,  
                                                values_from = wt)  
## # A tibble: 5 × 7  
##   smoke          Asian  Black   Mex  Mixed  White `NA`  
##   <chr>        <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 never         114.   117.   123.   125.   125.   119.  
## 2 now           98.8  108.   127.   108.   116.   114.  
## 3 once did, not now  111    117.    NA    134    126.    NA  
## 4 until current pregnancy 117    113.   129.   111    127.    NA  
## 5 <NA>          126    119    115    NA    131.    NA
```

- NB the `smoke` variable is character and sorted alphabetically.
- We don't *expect* you to use pipes, but they're useful



Summary

Summary

- Summarising data
 - `group_by` to set a group structure
 - `summarise` to calculate summary stats across group structure
 - `count` to see how many rows in each group
- Reshaping data frames
 - `pivot_longer` from variables side by side to key-value
 - `pivot_wider` from key-value to named column variables



Summary

- Dealing with rows and columns
 - `mutate` to create/modify/delete columns
 - `select` to choose columns
 - `filter` to choose rows based on logical condition
 - `slice*` to choose rows based on position or property
- Pipe
 - `%>%` to chain operations
- Wickham (2014) on what tidy data is
- Wickham et al. (2019) for more explanation of tidyverse

