Introduction to R from Zero to Hero
Part 2
“Data Manipulation and Exploration with R”

Isabella Gollini

isabella.gollini@ucd.ie
igollini.github.io

UCD Netsoc – 8th November 2022
All material available at: github.com/igollini/netsoc22

What will we do today?

1. Introduction to R and RStudio (Part 1)
2. RStudio Projects (Part 1)
3. Data import and handling (Part 1)
4. Data manipulation and summaries with `dplyr` (Part 2)
5. Graphics with `ggplot2` (Part 2)
6. Dynamic documents with R Markdown (Part 1 and 2)

Data Pre-processing

The imported data are a subset of records from a US Child Health and Development Study, corresponding to live births of a single male foetus.

The data requires a lot of pre-processing:

- converting numeric variables to factors
- converting coded values to missing values
- filtering rows and selecting columns

The *dplyr* package can be used to help with many of these steps

dplyr

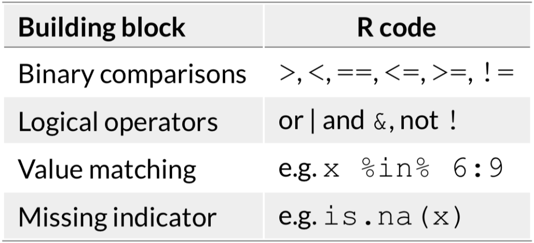
The *dplyr* package provides the following key functions to operate on data frames

- `filter()`
- `arrange()`
- `select()` (and `rename()`)
- `distinct()`
- `mutate()` (and `transmute()`)
- `summarise()`

filter()

filter() selects rows of data by criteria

```r
library(dplyr)
infant <- filter(infant, smoke != 9 & age > 18)
```



Building block	R code
Binary comparisons	>, <, ==, <=, >=, !=
Logical operators	or | and &, not !
Value matching	e.g. x %in% 6:9
Missing indicator	e.g. is.na(x)

select()

select() selects variables from the data frame.

Select named columns:

```
select(infant, gestation, sex)
```

Select a sequence of columns, along with an individual column

```
select(infant, pluralty:gestation, parity)
```

Select all columns *except* specified columns

```
select(infant, -(id:outcome), -sex)
```

mutate()

mutate() computes new columns based on existing columns. Re-using an existing name replaces the old variable.

```
mutate(infant,
       wt = ifelse(wt == 999, NA, wt),
       wt = wt * 0.4536)
```

To only keep the computed columns, use transmute() in place of mutate().

Chaining

We can use | > (or % > % magrittr pipe) to pipe the data from one step to the next

```
infant <- infant |>
    filter(smoke != 9 & age > 18) |>
    select(-(id:outcome), -sex) |>
    mutate(wt = ifelse(wt == 999, NA, wt),
           wt = wt * 0.4536)
```

Any function with data as the first argument can be added to the data pipeline.

summarise()

summarise() function is for computing single number summaries of variables, so typically comes at the end of a pipeline or in a separate step

```
stats <- summarise(infant,
                   `Mean mother's weight (kg)` = mean(wt, na.rm = TRUE),
                   `Sample size` = sum(!is.na(wt)),
                   `Total sample size` = n())
stats
```

```
# # A tibble: 1 x 3
#   `Mean mother's weight (kg)` `Sample size` `Total sample size`
#                         <dbl>         <int>               <int>
# 1                        58.4          1167                1203
```

In an R markdown document we can convert the result to a markdown table using kable from the **knitr** package

```
kable(stats, align = "ccc")
```

Grouped Operations

Grouping can be set on a data frame using group_by. This is most useful summarise().

```
infant <- infant |> filter(race != 10) |>
    mutate(`Ethnic group` = recode(race, `6` = "Latino", `7` = "Black",
                                   `8` = "Asian", `9` = "Mixed",
                                   .default = "White"))
res <- infant |> group_by(`Ethnic group`) |>
    summarise(`Mean weight (kg)` = mean(wt, na.rm = TRUE))
kable(res, align = "lc")
```

Ethnic group	Mean weight (kg)
Asian	49.84
Black	62.63
Latino	54.12
Mixed	58.46
White	57.86

Plots

In RStudio, graphs are displayed in the Plots window. The plot is sized to fit the window and will be rescaled if the size of the window is changed.

Back and forward arrows allow you to navigate through graphs that have been plotted.

Graphs can be saved in various formats using the Export drop down menu, which also has an option to copy to the clipboard.

First we consider “no-frills” plots, for quick exploratory plots.

```
infant <- infant |>
    mutate(gestation = ifelse(gestation == 999,
                              NA, gestation))
```

Boxplots

```
boxplot(infant$bwt)
with(infant, boxplot(bwt ~ `Ethnic group`))
```

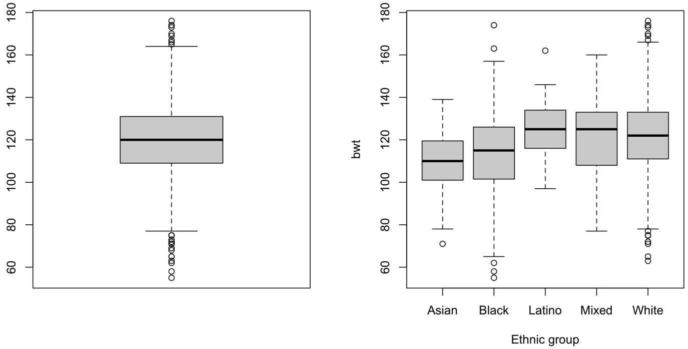


Figure 1: Left: boxplot of of birth weight. Right: boxplots of birth weight for each ethnic group.

Histogram/Density

```
hist(infant$bwt)
plot(density(infant$bwt))
```

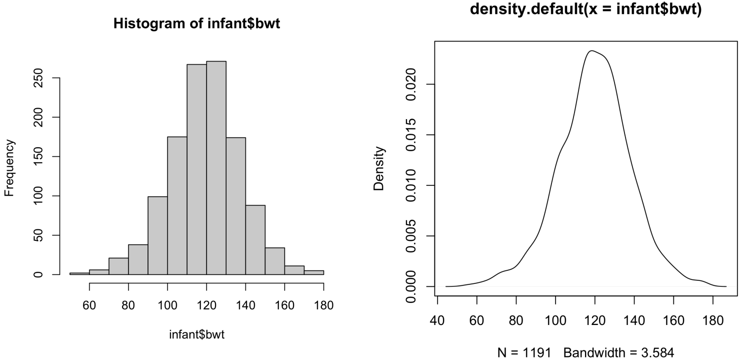


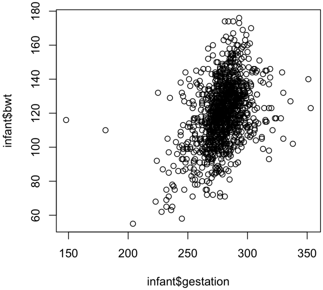
Figure 2: Left: histogram of of birth weight. Right: density curve of birth weight.

Scatterplots

```
plot(bwt ~ gestation, data = infant)
```

or

```
plot(infant$gestation, infant$bwt)
```



ggplot2

ggplot2 is a package that produces plots based on the *grammar of graphics* (Leland Wilkinson), the idea that you can build every graph from the same components

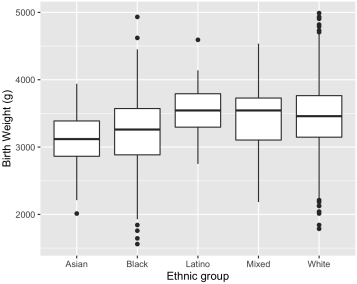
- a data set
- a co-ordinate system
- and *geoms* visual marks that represent data points

To display values, you map variables in the data to visual properties of the geom (*aesthetics*), like size, colour, x-axis and y-axis.

It provides a unified system for graphics, simplifies tasks such as adding legends, and is fully customisable.

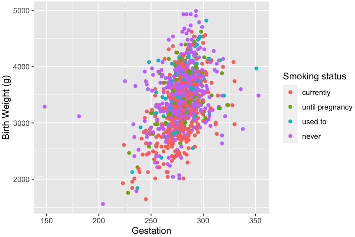
ggplot2 boxplots

```
library(ggplot2)
ggplot(infant, aes(y = bwt * 28.35, x = `Ethnic group`)) +
    geom_boxplot() +
    ylab("Birth Weight (g)")
```



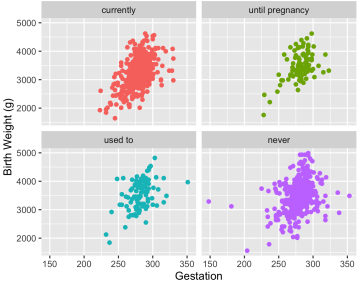
ggplot2 with Colour

```
infant <- mutate(infant, smoke = recode_factor(smoke,
                                        `1` = "currently", `2`= "until pregnancy",
                                        `3` = "used to", `0` = "never"))
p <- ggplot(infant, aes(y = bwt * 28.35, x = gestation, color = smoke)) +
    geom_point() + labs(x = "Gestation", y = "Birth Weight (g)",
                        color = "Smoking status")
p
```



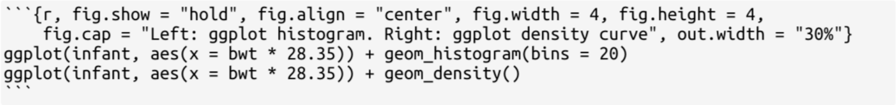
ggplot2 Facetting

```
p + facet_wrap(~ smoke) + guides(color = "none")
```



Plots in R Markdown

The chunk options enable you to arrange plots (caption used as alt text here)



~~~
```{r, fig.show = "hold", fig.align = "center", fig.width = 4, fig.height = 4,
    fig.cap = "Left: ggplot histogram. Right: ggplot density curve", out.width = "30%"}
ggplot(infant, aes(x = bwt * 28.35)) + geom_histogram(bins = 20)
ggplot(infant, aes(x = bwt * 28.35)) + geom_density()
```
~~~

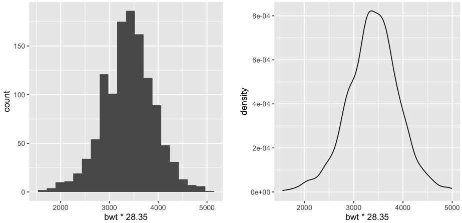


Figure 3: Left: ggplot histogram. Right: ggplot density curve

Your Turn! 1/2

`cut()` (in the **base** package) can be used to create a factor from a continuous variable, e.g.:

```
cut(c(0.12, 1.37, 0.4, 2.3), breaks = c(0, 1, 2, 3))

# [1] (0,1] (1,2] (0,1] (2,3]
# Levels: (0,1] (1,2] (2,3]
```

where $(0, 1]$; is the set of numbers > 0 and ≤ 1, etc.

1. The infant birth weight `bwt` is given in ounces. In the `table-bwt` chunk use `mutate()` to create a new factor called `Birth weight (g)`, by converting `bwt` to grams through multiplication by 28.35 and then converting the result to a factor with the following categories: $(1500, 2000]$, $(2000, 2500]$, $(2500, 3000]$, $(3000, 3500]$, and $(3500, 5000]$.
2. An infant is categorised as low weight if its birth weight is ≤ 2500 grams, regardless of gestation. Use `group_by()` to group the data by the new factor weight factor, then pipe to `summarise()` to count the number of infants in each weight category.

Your Turn! 2/2

1. Do some nice graph!
2. Include them into your R Markdown file, select appropriate size for the figure and caption.
3. Briefly comment your findings in your R Markdown file.

Learning more/getting support

- Posit/RStudio cheatsheets (`rmarkdown`, `dplyr`, `ggplot2`) `https://posit.co/resources/cheatsheets/`

- R for Data Science (1st edition) (data handling, basic programming and modelling, R markdown) `https://r4ds.had.co.nz/`

- R for Data Science (Work in Progress 2nd edition) (data handling, basic programming and modelling, R markdown) `https://r4ds.hadley.nz/`

- RStudio Community (friendly forum focused on "tidyverse" packages, including `dplyr`, `ggplot2`) `https://community.rstudio.com/`

- If you are planning on using R and Python in the same project you may want to use `quarto` instead of R Markdown `https://quarto.org/`

Learning more/getting support

- UCD modules STAT40620/STAT40730 Data Programming with R (in-class and online) [autumn] and STAT40830 Adv Data Prog with R (online) [summer]

- Task views `http://cran.r-project.org/web/views/` provide an overview of R's support for different topics, e.g. Bayesian inference, survival analysis, ...

- `http://www.rseek.org/` – search the web, online manuals, task views, mailing lists and functions.

- Package vignettes, many books, e.g. on `https://bookdown.org/`.

- Create your own package: `https://r-pkgs.org/` Wickham, Hadley, and Bryan, Jenny "R packages: organize, test, document, and share your code" O'Reilly Media, Inc.

Journals

- The R Journal https://journal.r-project.org/
- Journal of Statistical Software https://www.jstatsoft.org/
- The Journal of Open Source Software https://joss.theoj.org/
- Several other journals have special sessions or special issues dedicated to software development.