THE UNIVERSITY *of* EDINBURGH | usher institute

HDS Tutorial 1

Week 2

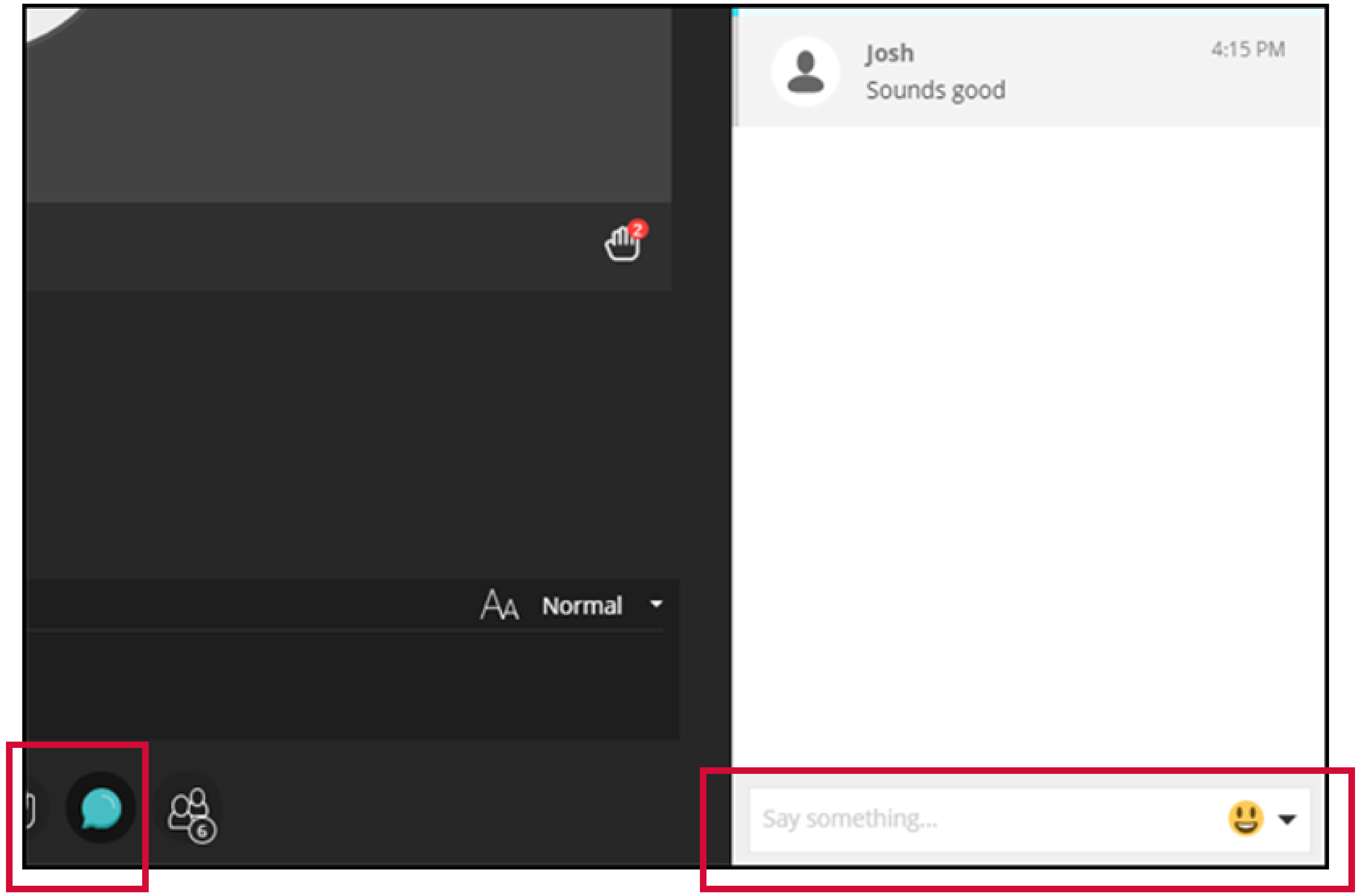
Audio check

Can you hear the presenter talking?

Please type **yes** or **no** in the "Text chat area"

If you can't hear:

- Check your Audio/Visual settings in the Collaborate Panel
- Try signing out and signing back into the session
- Type into the chat box and a moderator will try to assist you

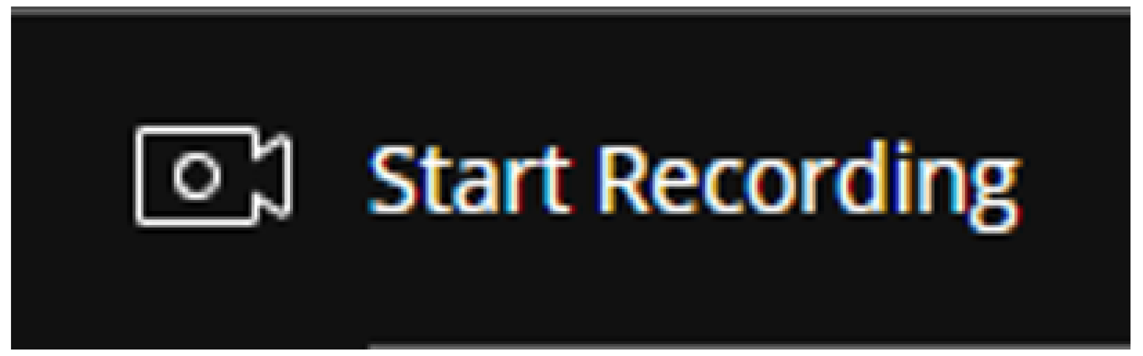

Josh 4:15 PM
Sounds good

Recording

This session will now be recorded. Any further information that you provide during a session is optional and in doing so you give us consent to process this information.

These sessions will be stored by the University of Edinburgh for one year and published for 30 days after the event. Schools or Services may use the recordings for up to a year on relevant websites.

By taking part in a session, you give us your consent to process any information you provide during it.



ddi.hsc.talent@ed.ac.uk

Supported by

HDS Tutorial 1

Week 2

Agenda

- What is R
- Why use R
- Example data flow presentation using health data
- Tips for starting out with R
- How to search for help online
- Q&A

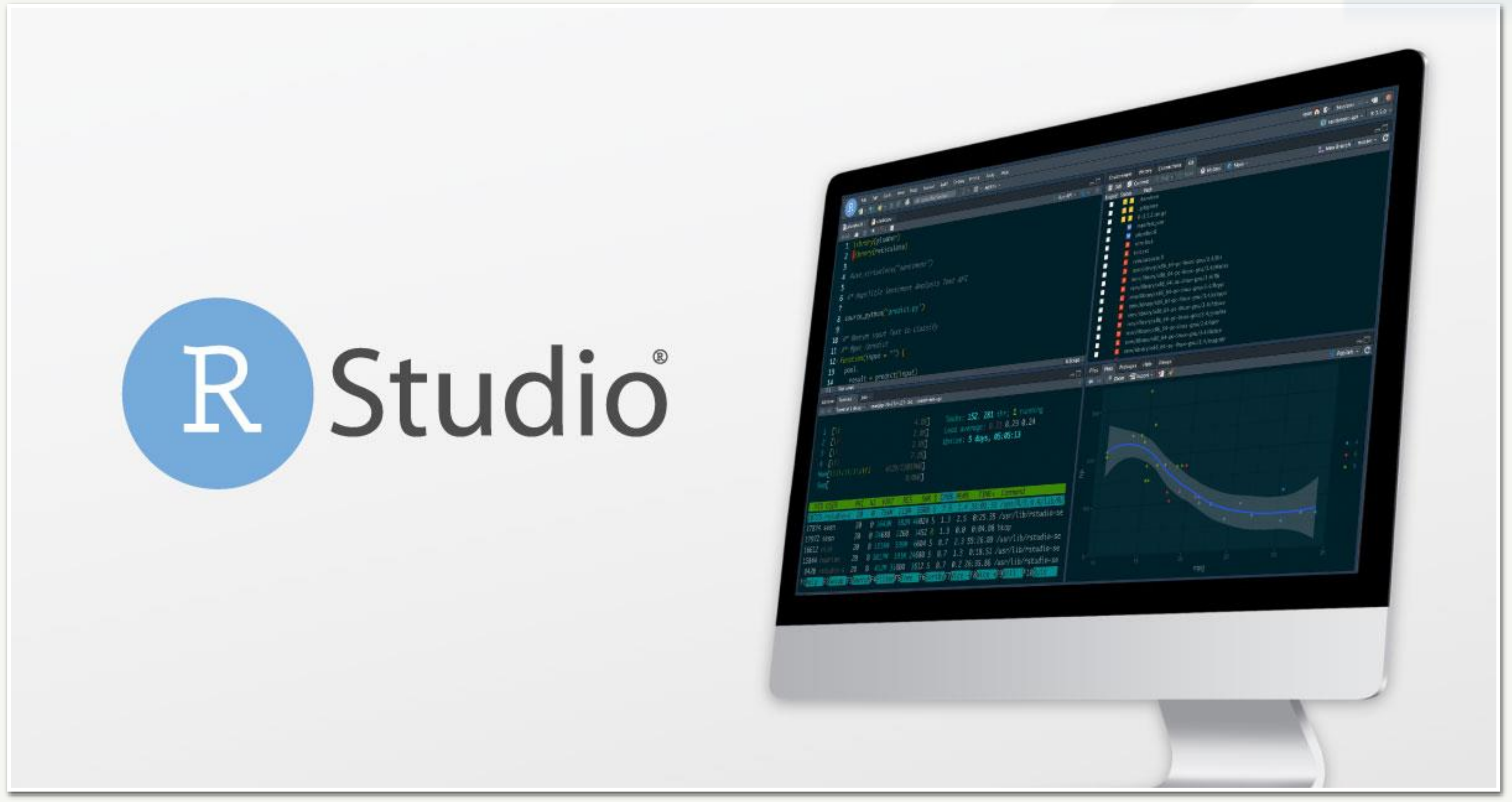
Has everyone downloaded R/R Studio?

Or are you using Noteable?

Organisations behind R

- The R Project
- Comprehensive R Archive Network (CRAN)
- R Studio

The Comprehensive R Archive Network



What are some of the reasons you want to learn R?
What benefits have you heard over other programming languages?

Advantages

- **“Open source” software**
  - Free (!!)
- **New methods implemented faster**
- **More flexible/customizable**
- **Anyone can contribute**
  - Do not have to work at R to contribute to R

Disadvantages

- **No centralized support**
- **Many find it harder to learn -** but that is why we are here!
- **Less consistency across procedures**
  - other software you rely on the programmers
- **…anyone can contribute (some packages are far better than others)**

Further Reasons for Why R…

- **High quality and robust data visualization**
  - e.g., ggplot2 — covered in depth in Week 4
- **Go-to language for statistics and data science; used in almost every industry**
- **Vast R community support**
  - Stack Overflow !
  - Twitter Rstats
  - etc.

- **The most comprehensive statistical analysis package new technology & ideas often appear first in R**
- **tidyverse**
- **RMarkdown (covered in Week 6)**
- **you can do so many things beyond data analysis/processing… maps, calendars, etc.!**
- **It can be very satisfying and fun once you get the hang of it!**

Formalized set of packages and tools that have a consistently structured programming interface

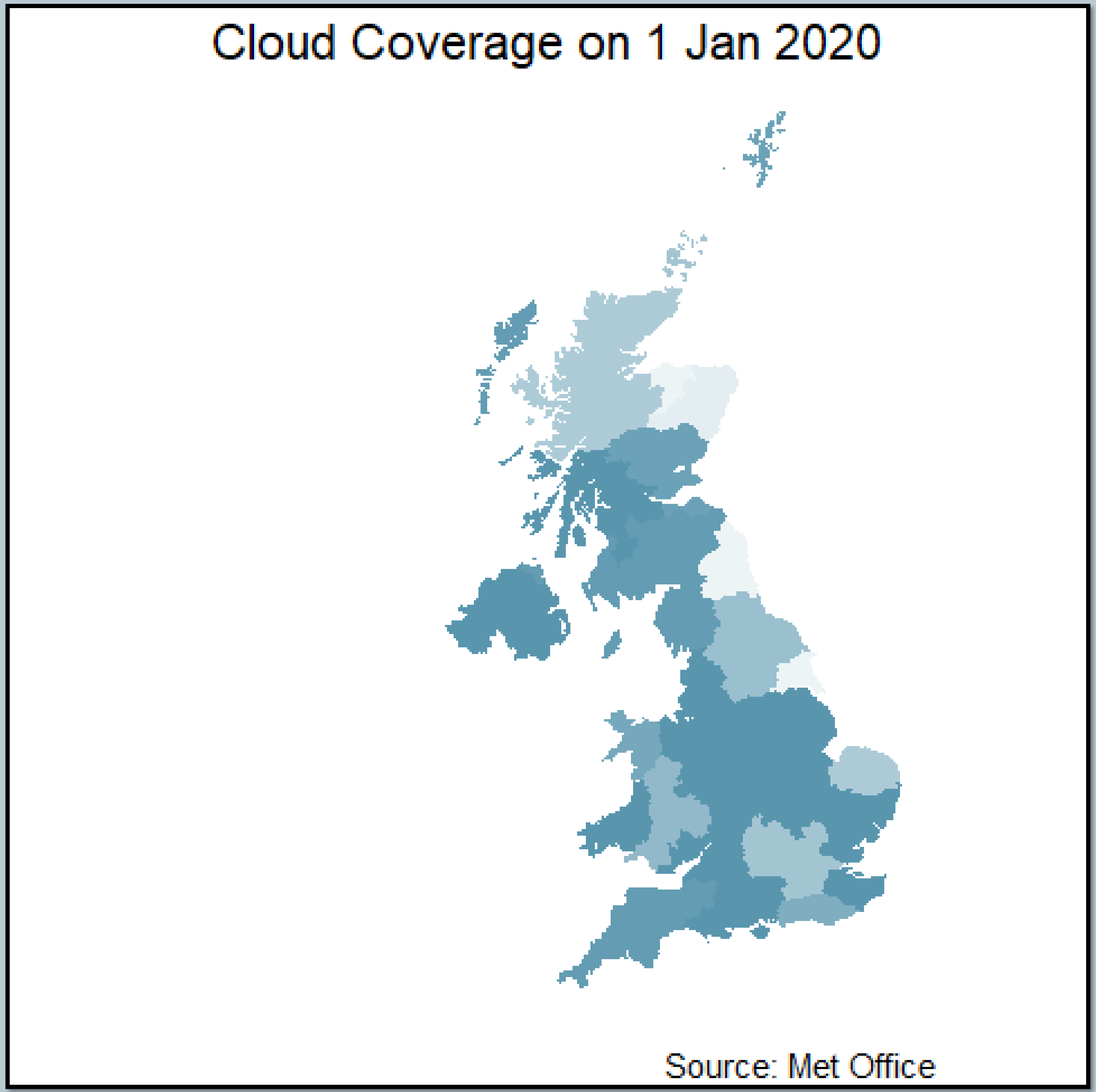
- as opposed to base R, which is more complex/varied and less user friendly

`library(tidyverse)` loads the core tidyverse packages

`ggplot2`	Make amazing graphs
`tibble`	Nice data frames
`tidyr`	Tidy your data
`readr`	Get data into R
`purrr`	Cool functional programming stuff
`dplyr`	Action verbs for manipulating data
`stringr`	Regular Expression tools for strings
`forcats`	Deal with factors

Into the Tidyverse!

streamlined data wrangling & visualization



An animation made by Kevin!

An example of:

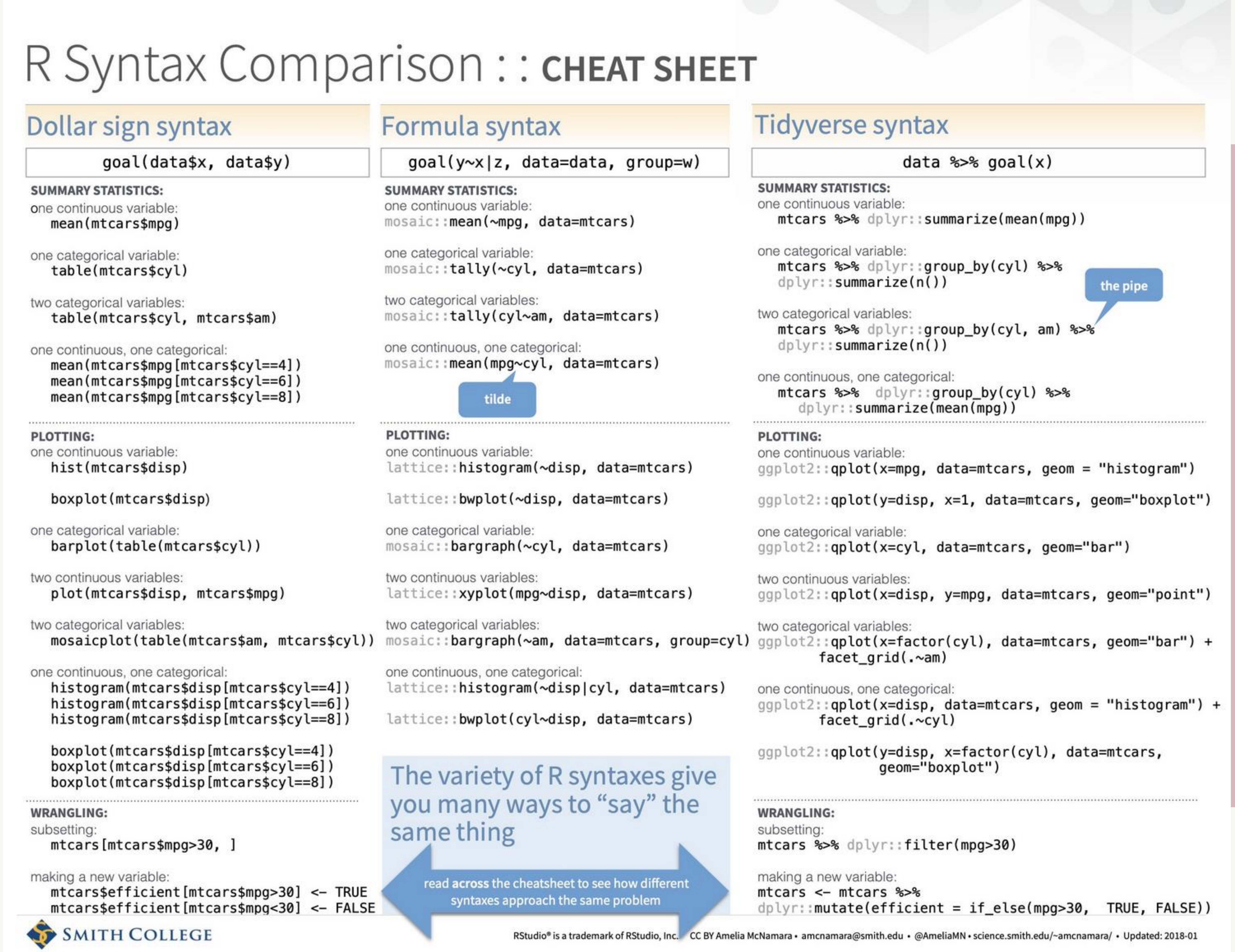
- the power of data visualization in R
- what HDS is building you towards

Animation (gganimate package) with maps (sf package) and Met Office data

R Syntax

There are 3 main different types of syntax you might come across:

1. **base R (**$**)**
2. **tidy verse (**%>%**)**
3. **formula (**~**)**



R Syntax Comparison : : CHEAT SHEET

Dollar sign syntax

```
goal(data$x, data$y)
```

SUMMARY STATISTICS:

one continuous variable:
```
mean(mtcars$mpg)
```
one categorical variable:
```
table(mtcars$cyl)
```
two categorical variables:
```
table(mtcars$cyl, mtcars$am)
```
one continuous, one categorical:
```
mean(mtcars$mpg[mtcars$cyl==4])
mean(mtcars$mpg[mtcars$cyl==6])
mean(mtcars$mpg[mtcars$cyl==8])
```

PLOTTING:

one continuous variable:
```
hist(mtcars$disp)

boxplot(mtcars$disp)
```
one categorical variable:
```
barplot(table(mtcars$cyl))
```
two continuous variables:
```
plot(mtcars$disp, mtcars$mpg)
```
two categorical variables:
```
mosaicplot(table(mtcars$am, mtcars$cyl))
```
one continuous, one categorical:
```
histogram(mtcars$disp[mtcars$cyl==4])
histogram(mtcars$disp[mtcars$cyl==6])
histogram(mtcars$disp[mtcars$cyl==8])

boxplot(mtcars$disp[mtcars$cyl==4])
boxplot(mtcars$disp[mtcars$cyl==6])
boxplot(mtcars$disp[mtcars$cyl==8])
```

WRANGLING:

subsetting:
```
mtcars[mtcars$mpg>30, ]
```
making a new variable:
```
mtcars$efficient[mtcars$mpg>30] <- TRUE
mtcars$efficient[mtcars$mpg<30] <- FALSE
```

Formula syntax

```
goal(y~x|z, data=data, group=w)
```

SUMMARY STATISTICS:

one continuous variable:
```
mosaic::mean(~mpg, data=mtcars)
```
one categorical variable:
```
mosaic::tally(~cyl, data=mtcars)
```
two categorical variables:
```
mosaic::tally(cyl~am, data=mtcars)
```
one continuous, one categorical:
```
mosaic::mean(mpg~cyl, data=mtcars)
```

PLOTTING:

one continuous variable:
```
lattice::histogram(~disp, data=mtcars)

lattice::bwplot(~disp, data=mtcars)
```
one categorical variable:
```
mosaic::bargraph(~cyl, data=mtcars)
```
two continuous variables:
```
lattice::xyplot(mpg~disp, data=mtcars)
```
two categorical variables:
```
mosaic::bargraph(~am, data=mtcars, group=cyl)
```
one continuous, one categorical:
```
lattice::histogram(~disp|cyl, data=mtcars)

lattice::bwplot(cyl~disp, data=mtcars)
```

The variety of R syntaxes give you many ways to "say" the same thing

read **across** the cheatsheet to see how different syntaxes approach the same problem

Tidyverse syntax

```
data %>% goal(x)
```

SUMMARY STATISTICS:

one continuous variable:
```
mtcars %>% dplyr::summarize(mean(mpg))
```
one categorical variable:
```
mtcars %>% dplyr::group_by(cyl) %>%
dplyr::summarize(n())
```

two categorical variables:
```
mtcars %>% dplyr::group_by(cyl, am) %>%
dplyr::summarize(n())
```
one continuous, one categorical:
```
mtcars %>%  dplyr::group_by(cyl) %>%
  dplyr::summarize(mean(mpg))
```

PLOTTING:

one continuous variable:
```
ggplot2::qplot(x=mpg, data=mtcars, geom = "histogram")

ggplot2::qplot(y=disp, x=1, data=mtcars, geom="boxplot")
```
one categorical variable:
```
ggplot2::qplot(x=cyl, data=mtcars, geom="bar")
```
two continuous variables:
```
ggplot2::qplot(x=disp, y=mpg, data=mtcars, geom="point")
```
two categorical variables:
```
ggplot2::qplot(x=factor(cyl), data=mtcars, geom="bar") +
        facet_grid(.~am)
```
one continuous, one categorical:
```
ggplot2::qplot(x=disp, data=mtcars, geom = "histogram") +
        facet_grid(.~cyl)

ggplot2::qplot(y=disp, x=factor(cyl), data=mtcars,
               geom="boxplot")
```

WRANGLING:

subsetting:
```
mtcars %>% dplyr::filter(mpg>30)
```
making a new variable:
```
mtcars <- mtcars %>%
dplyr::mutate(efficient = if_else(mpg>30,  TRUE, FALSE))
```

SMITH COLLEGE

RStudio® is a trademark of RStudio, Inc. • CC BY Amelia McNamara • amcnamara@smith.edu • @AmeliaMN • science.smith.edu/~amcnamara/ • Updated: 2018-01

- last updated 2018, needs some updating
- credit to Amelia McNamara
- Full cheat sheet here

Bar chart demonstration in R using heath care data

Tips to start out with R

- **Add comments to your code if using script**
  - Use a # at the beginning of the line or at the end of the line
- **You can also add sections to your script code**
  - At the end of a section add 4 dashes - or 4 hashes #
- **Avoid dots and spaces in variable names**
  - instead used capital letters or underscore
- **Check Cheatsheets, especially when using a new package**
  - Top menu *Help > Cheatsheets* or https://rstudio.com/resources/cheatsheets/
- **If you are unsure of what something does, type into the console a question mark followed by the function or package**
  - >?function
  - for common functions, check the introverse package documentation
- **General introduction to R as a programming language: A Succinct Intro to R**
- **Always load packages at the start of a script**
- **Make sure everything is spelled correctly! and capitalization is consistent when loading in data**
- **Practice makes perfect. Build familiarity with the system and don't be afraid to make mistakes**
- **When in doubt, Google it!!**

Have you looked for R help online?
If so, was it effective?
What did you type in?

How to effectively search for help online:

- when in doubt, copy and paste an error into google
- include the package name or “in R” in your search
- built-in R help function >?function
- StackOverflow
- RStudio Community
- Be willing and ready to adapt code to your context
- Holly has a lovely document with more details

Questions?

Some helpful definitions…

Working Directory

- After installing R & RStudio, you need to set the working directory
  - This is the location on your computer where any data files to imported can be found, and where any R scripts (the files that save your code) will be saved
- In R studio, you can set the working directory with the menus (*Session >> Set Working Directory >> Choose Directory*) or with a line of code that gives the path of the folder on your computer:
  - >setwd("Drive:/Folder1/Folder2")
- If you have made a new R Project from an existing Directory as shown in the live demo, working directory will be set this way
- can always check working directory with the function:
  - >getwd()

Some vocabulary

- **Function =** how you get stuff done in R (chapters in metaphor)
- **Argument =** specifications of functions (specific pages in metaphor)
- **Packages =** are a collection of R functions, complied code and sample data. By default a set of packages are installed during installation. They are stored under a directory called "library" in the R environment (books in metaphor)
- **Documentation =** the explanations of functions and arguments for different packages written by the authors (glossary in metaphor)
- **"run" or "running code" =** enter command into the R console to make it happen
- **Script** = a text file containing (almost) the same commands that you would enter on the command line of R
- **Data frame** = a *special type of list* where every element of the list has same length (i.e. data frame is a "rectangular" list); *de facto* data structure for most tabular data and what we use for statistics.
  - **Tibble** = tidyverse style dataframe
- **Indexing** = selecting a subset of the elements in order to use them in further analysis or possibly change them. Style depends on syntax (see slide 13 + cheat sheets)

Variable Types in R

- **character**: `"a"`, `“swc"`
- **numeric**: `2`, `15.5`
  - “continuous” variable in other software
  - **integer** is similar, use numeric in practice (less limitations)
  - **double** is similar, but is a numeric value with decimal places
- **logical**: `TRUE`, `FALSE`
- **complex:** `1+4i` (complex numbers with real and imaginary parts)
- **factor**: used to describe items that can have a finite number of values (gender, social class, etc.).
  - A factor has a `levels` attribute and class `"factor"`
  - Optionally, it may also contain a `contrasts` attribute which controls the parametrization used when the factor is used in a modeling functions.
  - “categorical” variable in other software. Tell R that a variable is **nominal** by making it a factor. An ordered factor is used to represent an **ordinal variable**.

Data Structures in R

- **Vectors** = most common and basic structure in R; a collection of elements that are most commonly of mode `character`, `logical`, `integer` or `numeric`.
  - **Atomic vector** = vector where elements much be the same data type; default vector type
  - **List =** a special type of vector. Each element can be a different type.
- **Matrix =** an extension of the numeric or character vectors. They are not a separate type of object but simply an atomic vector with dimensions; the number of rows and columns. As with atomic vectors, the elements of a matrix must be of the same data type.
- **Array** = similar to matrices but can have more than two dimensions
- **Data frame =** a *special type of list* where every element of the list has same length (i.e. data frame is a "rectangular" list); *de facto* data structure for most tabular data and what we use for statistics