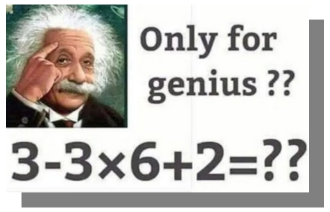
R^4H_2O : : SESSION ONE CHEAT SHEET

Arithmetic

R is meme proof (applies BODMAS)

```
3 - 3 * 6 + 2
```



Variables

Use meaningful variable names

```
diameter <- 150
nonRevWater <- 10
non_rev_water <- 20
```

Vectors

Create vectors:

```
nonRevWater <- c(1, 3, -1, 0)
day <- 1:4
```

Vector subsets:

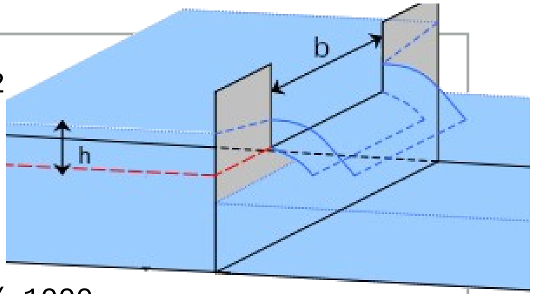
```
nonRevWater[2]
nonRevWater[2:3]
```

Math Functions

```
sqrt(nonRevWater)
sum(nonRevWater)
prod(nonRevWater)
factorial(nonRevWater)
abs(nonRevWater)
exp(nonRevWater)
log(nonRevWater, base = 10)
```

Extreme outcomes can be NaN (Not a Number) or Inf (approaching infinity).

Case Study 0



```
Cd <- 0.62
g <- 9.81
b <- 0.5

h <- 100 / 1000
q <- (2 / 3) * Cd * sqrt(2 * g) *
     b * h^(3 / 2)
```

Packages

Packages are libraries of functions and data files to extend R. The CRAN website lists most packages. Initiate libraries every script with the packages you need, e.g.:

```
library(tidyverse)
```

Tidyverse is a collection of packages. Click on the hexagonal package logos for more info.

Reading CSV Files

The **readr package** reads delimited files, such as CSV (Comma-Separated Values).

```
library(readr)
gormsey <-
   read_csv("data/water_quality.csv")
```

Reading Excel Files

The **readxl package** imports Excel files.

```
library(readxl)
read_excel("data/water_quality.xlsx",
           sheet = 2, skip = 3)
```

Data Frames

Rectangular data (called a Tibble in Tidyverse) to store structured data.

```
glimpse(gormsey)
```

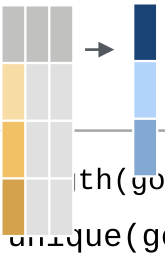
Columns: variables

```
gormsey$Result
gormsey[, 6]
gormsey[, "Result"]
```

Rows: observations

```
gormsey[1:10, ]
Gormsey[1:10, 4]
```

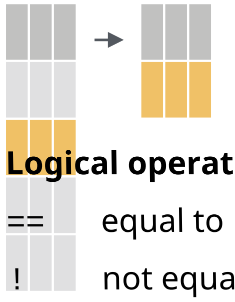
Counting Data



Count number of rows for each variable

```
length(gormsey$Measure)
unique(gormsey$Measure)
count(gormsey, Measure)
count(gormsey, Measure, Town)
```

Filtering Data



Filter extracts rows that meet logical criteria

Logical operators

==	equal to	!	'or'
!	not equal to	&	'and'

Use stringr package for wildcards

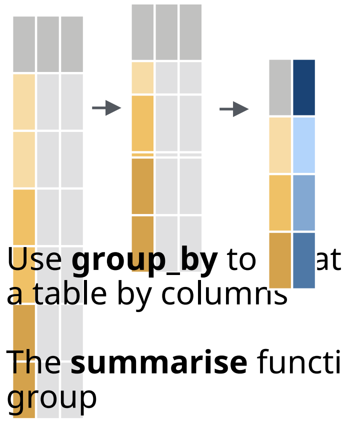
```
turb_mert <- filter(gormsey,
         Measure == "Turbidity" &
         Town != "Merton")

Turbidity <- filter(gormsey,
str_detect(Measure, "^t")
```

Statistics

```
turb_results <- turbidity$Result
mean(turb_results)
median(turb_results)
min(turb_results)
max(turb_results)
range(turb_results)
quantile(turb_results,
         probs = 0.95,
         method = 7)
var(turb_results)
sd(turb_results)
```

Grouping



Use **group_by** to create a grouped copy of a table by columns

The **summarise** function acts on each group

```
turb_gr <- group_by(turbidity,
                    Town)
summarise(turb_gr,
          avg = mean(Result),
          max = max(Result))
```

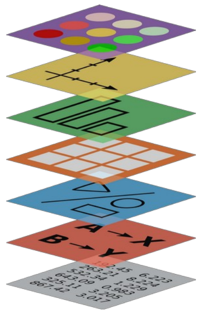
Finding Help

Use **help** function or **?** to read internal help.

Read the detailed cheat sheets on rstudio.com/resources/cheatsheets

R^4H_2O : : SESSION ONE CHEAT SHEET

ggplot2



Grammar of Graphics

- Theme
- Coordinates
- Statistics
- Facets (graph grids)
- Geometries
- Aesthetics
- Data

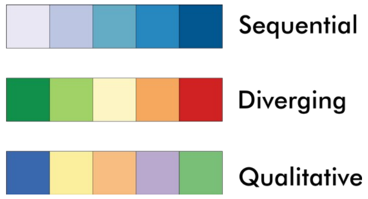
Layer 1: Data

```
ggplot(gormsey)
```

Layer 2: Aesthetics

```
ggplot(gormsey, aes(Measure))
```

Colour aesthetics (**`fill`** = for surfaces and **`color`** = for lines)



```
scale_fill_brewer()
scale_color_brewer()
scale_*_manual()
```

Layer 3: Geometries

```
ggplot(gormsey, aes(Measure)) +
    geom_bar()
```

```
ggplot(turbidity, aes(Date, Result)) +
    geom_line()
```

```
ggplot(turbidity, aes(Date, Result)) +
    geom_boxplot()
```

ggplot2 website has extensive documentation on geometries

Layer 4: Facets

line

```
facet_wrap(~Suburb)
```

grid

```
facet_grid(Measure~Suburb)
```

ggplot2 continued

Layer 5: Statistics and annotations

```
geom_smooth(method = "lm")
```

Add linear model to data (default is Loess)

```
geom_hline(aes(yintercept = 5))
```

Add horizontal line (indicating limits)

Layer 6: Coordinates

```
coords_flip()
```

Rotate axes

```
scale_*_continuous()
scale_*_log10()
```

Define *x* and *y* scales

Layer 7: Theme

Pre-configured themes (use TAB key to explore)

```
theme_*()
```

Use **`theme()`** for fine-tuning the design.

Cleaning Data

Variable Types

```
customers <- read_csv("casestudy2/customers")
type_convert(customers)
```

Manually convert to a number:

```
as.numeric()
```

Selecting Variables

```
select(customers, p01:p10)
select(customers, sarts_with("p"))
rename(new_name = old_name)
```

Changing of Adding Variables

```
mutate(variable1 = formula1,
       variable2 = formula2)
```

Missing Data

Indicated with **NA**. Test for missing data with **`is.na()`**

Use **`na.rm = TRUE`** option to ignore missing data in calculations. Check help for each function for details.

```
is.na(customers$term)
mean(customers$p01, na.rm = TRUE)
```

Tidy Data

```
pivot_longer(data, 2:3, names_to =
        "year", values_to = "cases")
```

country	1999	2000
A	0.7K	2K
B	37K	80K
C	212K	213K

↔

country	year	cases
A	1999	0.7K
B	1999	37K
C	1999	212K
A	2000	2K
B	2000	80K
C	2000	213K

```
pivot_wider(data, names_from = year,
          values_from = cases)
```

Joining Data

a

x1	x2
A	1
B	2
C	3

+

b

x1	x3
A	T
B	F
D	T

=

x1	x2	x3
A	1	T
B	2	F
C	3	NA

```
left_join(a, b, by = "x1")
```

Join matching rows from b to a

x1	x3	x2
A	T	1
B	F	2
D	T	NA

```
right_join(a, b, by = "x1")
```

Join matching rows from a to b

x1	x2	x3
A	1	T
B	2	F

```
inner_join(a, b, by = "x1")
```

Retain only rows in both sets

x1	x2	x3
A	1	T
B	2	F
C	3	NA
D	NA	T

```
full_join(a, b, by = "x1")
```

Retain all rows