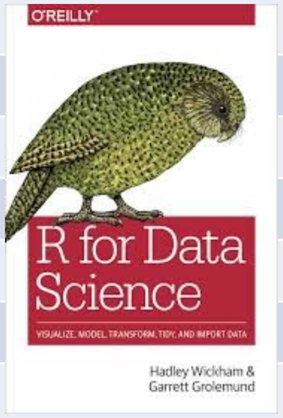
CT1100: Computer Systems

Topic 2: Programming in R

Prof. Jim Duggan,
School of Engineering & Informatics
National University of Ireland Galway.

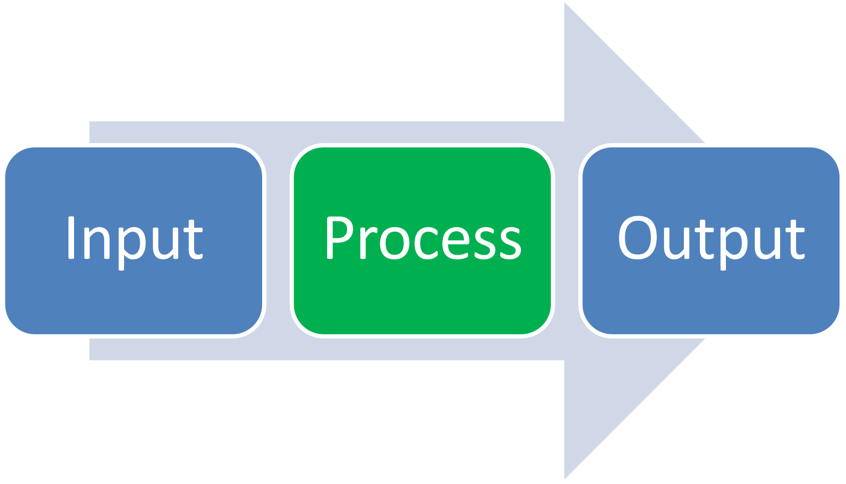
Topics to be Covered (R)

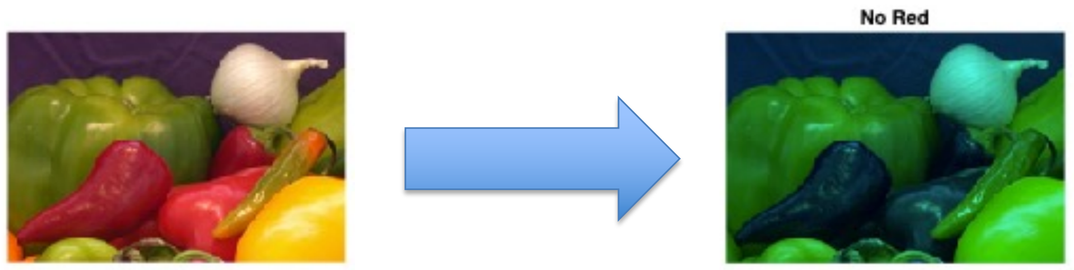
Topic	Description
1	Introduction to R and R Studio Cloud
2	A program in R
3	The tibble – a way of storing information
4	Data Visualisation I
5	Data Transformation I
6	Running a Script in R
7	Data Visualisation II
8	Data Transformation II
9	Exploring Data
10	Communicating Results


O'REILLY
R for Data Science
VISUALIZE, MODEL, TRANSFORM, TIDY, AND IMPORT DATA
Hadley Wickham & Garrett Grolemund

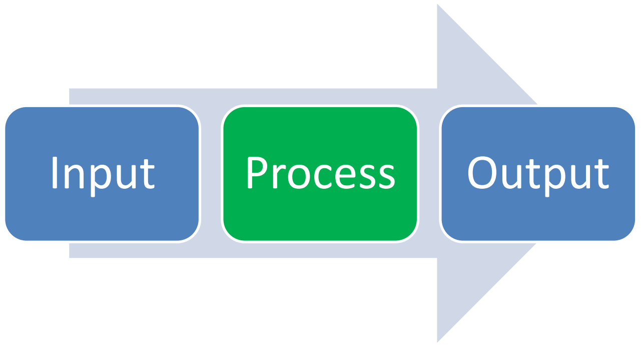
https://r4ds.had.co.nz

An Important Diagram!



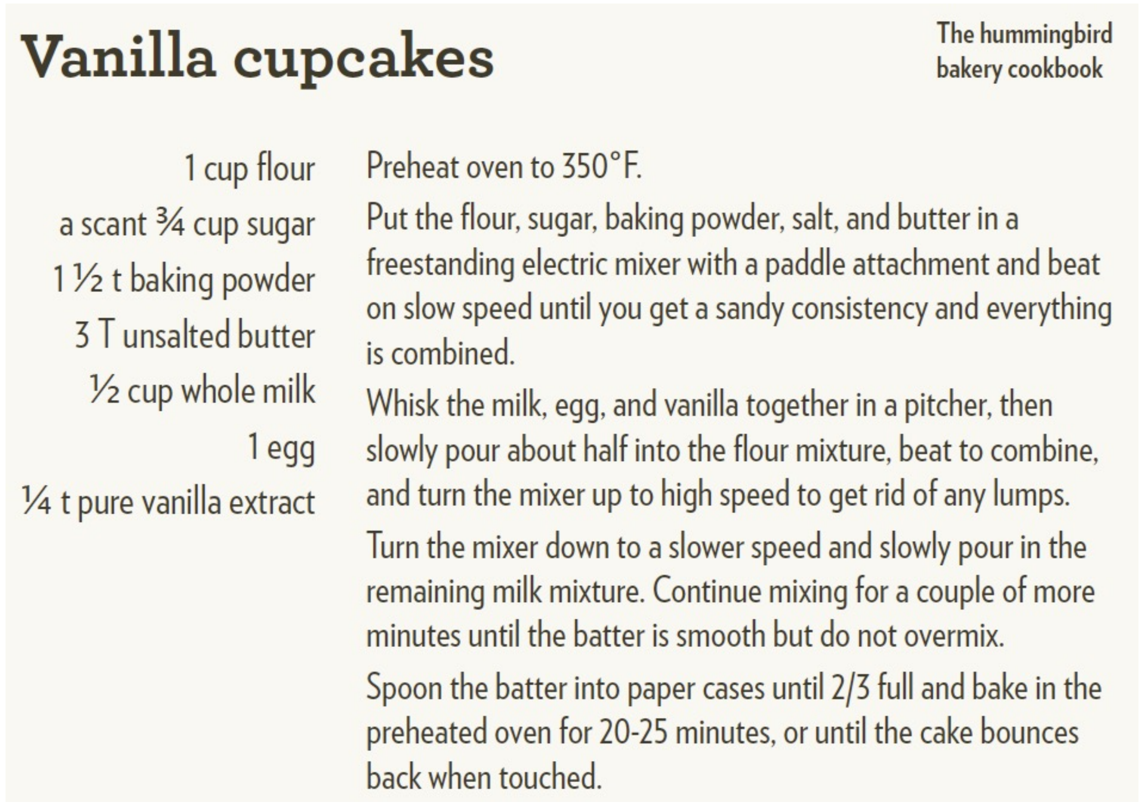


The Logic of a Program



- An algorithm is a plan for solving a problem, a program is an implementation of an algorithm
- Start by focusing on the (1) inputs and (2) outputs
- What is the key transformation(s) you want to achieve?
- What are the sequence of steps?

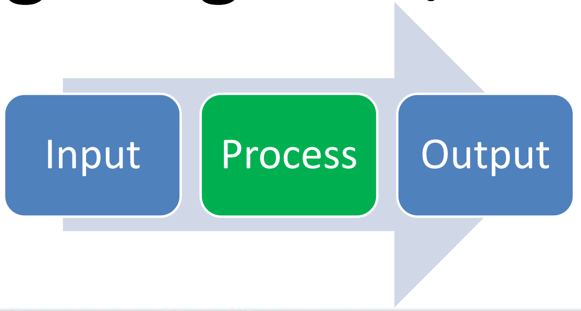
Cooking as an algorithm…

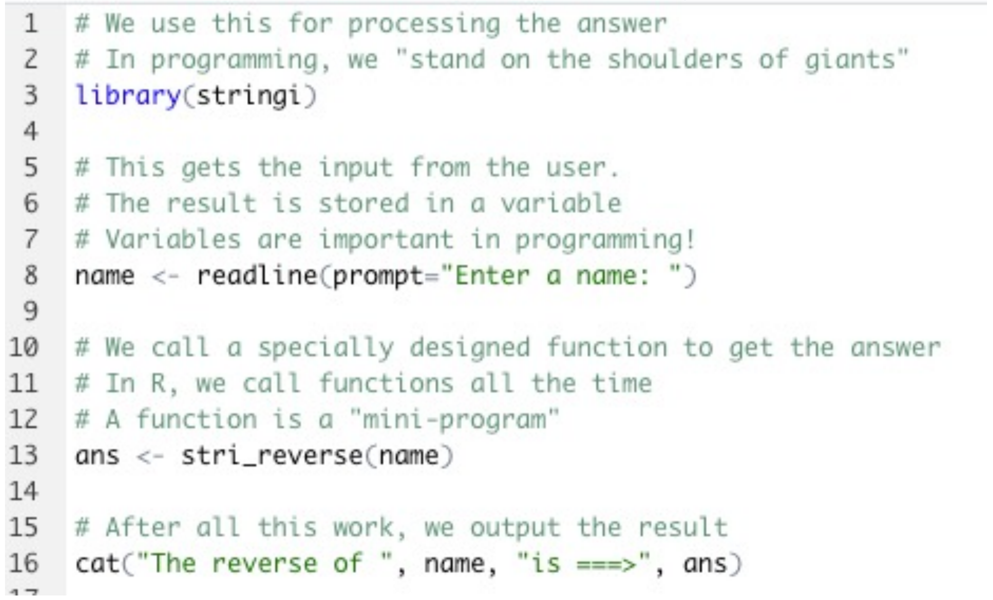

Vanilla cupcakes

The hummingbird bakery cookbook

1 cup flour	Preheat oven to 350°F.
a scant ¾ cup sugar	Put the flour, sugar, baking powder, salt, and butter in a freestanding electric mixer with a paddle attachment and beat on slow speed until you get a sandy consistency and everything is combined.
1 ½ t baking powder	Whisk the milk, egg, and vanilla together in a pitcher, then slowly pour about half into the flour mixture, beat to combine, and turn the mixer up to high speed to get rid of any lumps.
3 T unsalted butter	Turn the mixer down to a slower speed and slowly pour in the remaining milk mixture. Continue mixing for a couple of more minutes until the batter is smooth but do not overmix.
½ cup whole milk	Spoon the batter into paper cases until 2/3 full and bake in the preheated oven for 20-25 minutes, or until the cake bounces back when touched.
1 egg	
¼ t pure vanilla extract	

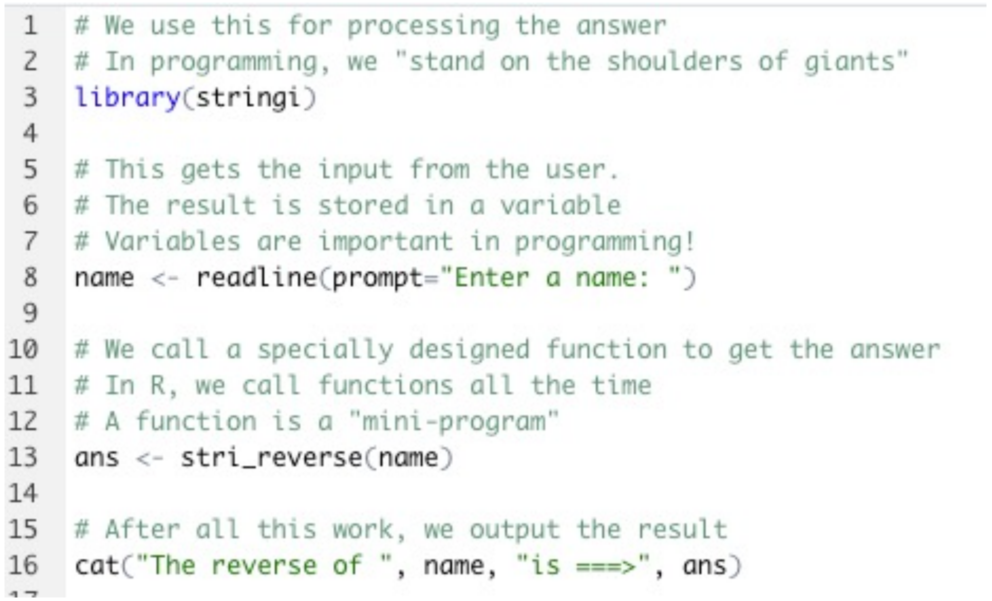
Writing Programs/Scripts



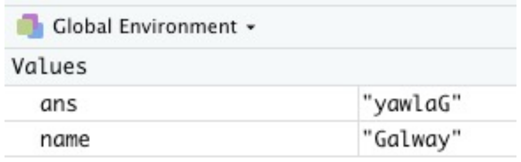


```r
# We use this for processing the answer
# In programming, we "stand on the shoulders of giants"
library(stringi)

# This gets the input from the user.
# The result is stored in a variable
# Variables are important in programming!
name <- readline(prompt="Enter a name: ")

# We call a specially designed function to get the answer
# In R, we call functions all the time
# A function is a "mini-program"
ans <- stri_reverse(name)

# After all this work, we output the result
cat("The reverse of ", name, "is ===>", ans)
```

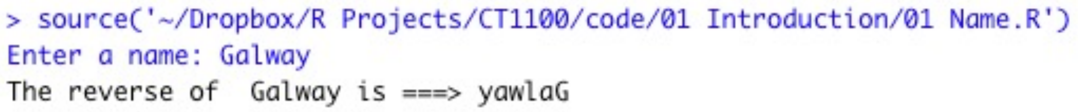
Running the Program in R



```r
# We use this for processing the answer
# In programming, we "stand on the shoulders of giants"
library(stringi)

# This gets the input from the user.
# The result is stored in a variable
# Variables are important in programming!
name <- readline(prompt="Enter a name: ")

# We call a specially designed function to get the answer
# In R, we call functions all the time
# A function is a "mini-program"
ans <- stri_reverse(name)

# After all this work, we output the result
cat("The reverse of ", name, "is ===>", ans)
```



ans	"yawlaG"
name	"Galway"

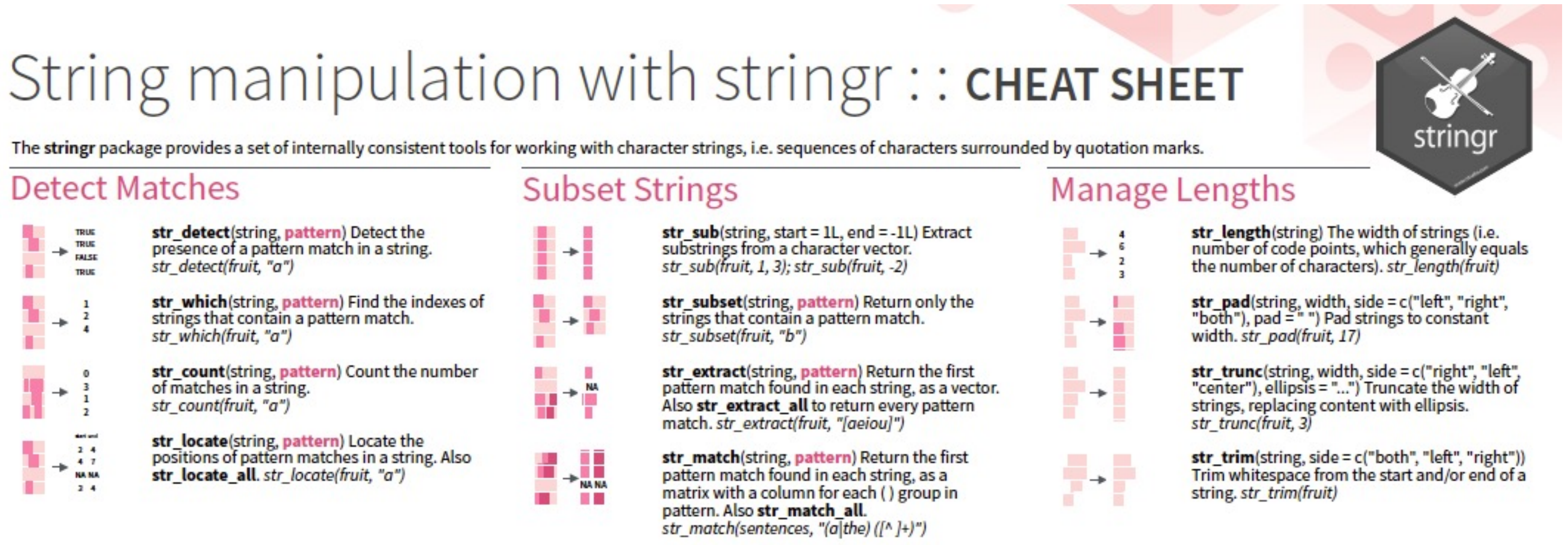


```
> source('~/Dropbox/R Projects/CT1100/code/01 Introduction/01 Name.R')
Enter a name: Galway
The reverse of  Galway is ===> yawlaG
```

Challenge 2.1

- This is our first program
- Look at the output…
- Is there any way you might change or improve this?
- Think in terms of transforming the output in some further way
- Do you think R might have a function that could help?

Install stringr and use and function



String manipulation with stringr : : **CHEAT SHEET**

The **stringr** package provides a set of internally consistent tools for working with character strings, i.e. sequences of characters surrounded by quotation marks.

Detect Matches

str_detect(string, pattern) Detect the presence of a pattern match in a string. *str_detect(fruit, "a")*

str_which(string, pattern) Find the indexes of strings that contain a pattern match. *str_which(fruit, "a")*

str_count(string, pattern) Count the number of matches in a string. *str_count(fruit, "a")*

str_locate(string, pattern) Locate the positions of pattern matches in a string. Also **str_locate_all**. *str_locate(fruit, "a")*

Subset Strings

str_sub(string, start = 1L, end = -1L) Extract substrings from a character vector. *str_sub(fruit, 1, 3); str_sub(fruit, -2)*

str_subset(string, pattern) Return only the strings that contain a pattern match. *str_subset(fruit, "b")*

str_extract(string, pattern) Return the first pattern match found in each string, as a vector. Also **str_extract_all** to return every pattern match. *str_extract(fruit, "[aeiou]")*

str_match(string, pattern) Return the first pattern match found in each string, as a matrix with a column for each ( ) group in pattern. Also **str_match_all**. *str_match(sentences, "(a|the) ([^]+)")*

Manage Lengths

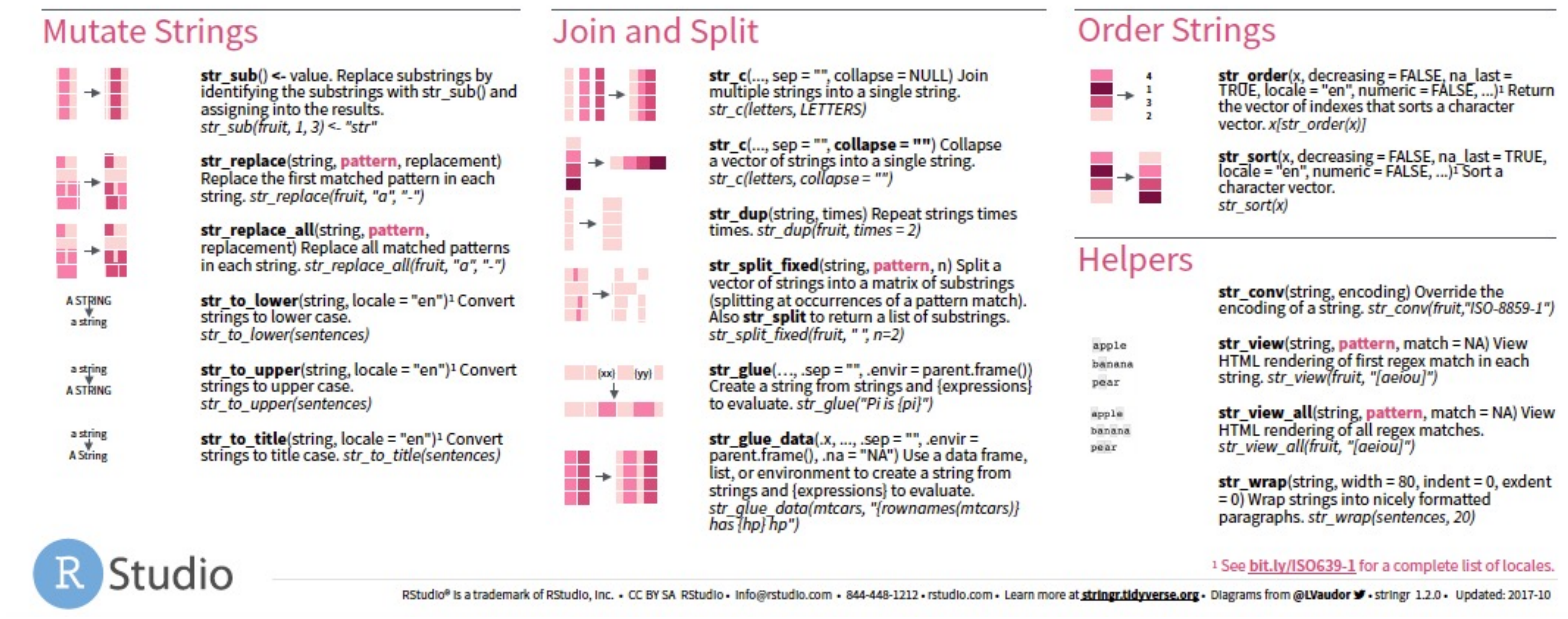
str_length(string) The width of strings (i.e. number of code points, which generally equals the number of characters). *str_length(fruit)*

str_pad(string, width, side = c("left", "right", "both"), pad = " ") Pad strings to constant width. *str_pad(fruit, 17)*

str_trunc(string, width, side = c("right", "left", "center"), ellipsis = "...") Truncate the width of strings, replacing content with ellipsis. *str_trunc(fruit, 3)*

str_trim(string, side = c("both", "left", "right")) Trim whitespace from the start and/or end of a string. *str_trim(fruit)*

Install stringr and use



Mutate Strings

str_sub() <- value. Replace substrings by identifying the substrings with str_sub() and assigning into the results. *str_sub(fruit, 1, 3) <- "str"*

str_replace(string, pattern, replacement) Replace the first matched pattern in each string. *str_replace(fruit, "a", "-")*

str_replace_all(string, pattern, replacement) Replace all matched patterns in each string. *str_replace_all(fruit, "a", "-")*

str_to_lower(string, locale = "en")¹ Convert strings to lower case. *str_to_lower(sentences)*

str_to_upper(string, locale = "en")¹ Convert strings to upper case. *str_to_upper(sentences)*

str_to_title(string, locale = "en")¹ Convert strings to title case. *str_to_title(sentences)*

Join and Split

str_c(..., sep = "", collapse = NULL) Join multiple strings into a single string. *str_c(letters, LETTERS)*

str_c(..., sep = "", **collapse = ""**) Collapse a vector of strings into a single string. *str_c(letters, collapse = "")*

str_dup(string, times) Repeat strings times times. *str_dup(fruit, times = 2)*

str_split_fixed(string, pattern, n) Split a vector of strings into a matrix of substrings (splitting at occurrences of a pattern match). Also **str_split** to return a list of substrings. *str_split_fixed(fruit, " ", n=2)*

str_glue(..., .sep = "", .envir = parent.frame()) Create a string from strings and {expressions} to evaluate. *str_glue("Pi is {pi}")*

str_glue_data(.x, ..., .sep = "", .envir = parent.frame(), .na = "NA") Use a data frame, list, or environment to create a string from strings and {expressions} to evaluate. *str_glue_data(mtcars, "{rownames(mtcars)} has {hp} hp")*

Order Strings

str_order(x, decreasing = FALSE, na_last = TRUE, locale = "en", numeric = FALSE, ...)¹ Return the vector of indexes that sorts a character vector. *x[str_order(x)]*

str_sort(x, decreasing = FALSE, na_last = TRUE, locale = "en", numeric = FALSE, ...)¹ Sort a character vector. *str_sort(x)*

Helpers

str_conv(string, encoding) Override the encoding of a string. *str_conv(fruit,"ISO-8859-1")*

str_view(string, pattern, match = NA) View HTML rendering of first regex match in each string. *str_view(fruit, "[aeiou]")*

str_view_all(string, pattern, match = NA) View HTML rendering of all regex matches. *str_view_all(fruit, "[aeiou]")*

str_wrap(string, width = 80, indent = 0, exdent = 0) Wrap strings into nicely formatted paragraphs. *str_wrap(sentences, 20)*

¹ See bit.ly/ISO639-1 for a complete list of locales.

RStudio

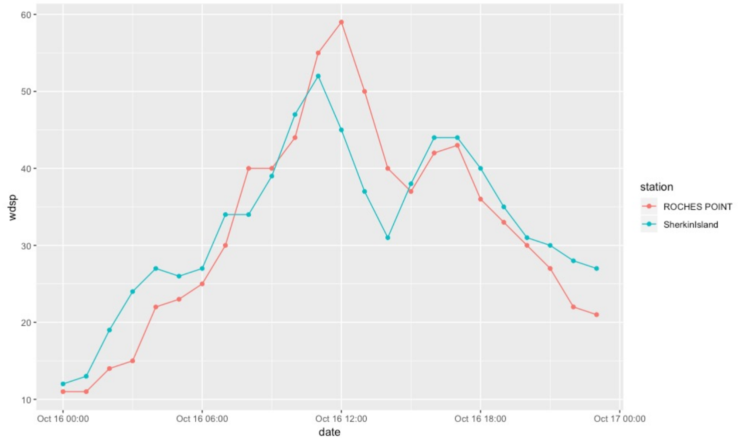
RStudio® is a trademark of RStudio, Inc. • CC BY SA RStudio • info@rstudio.com • 844-448-1212 • rstudio.com • Learn more at stringr.tidyverse.org • Diagrams from @LVaudor • stringr 1.2.0 • Updated: 2017-10

Exploring Data – the tibble

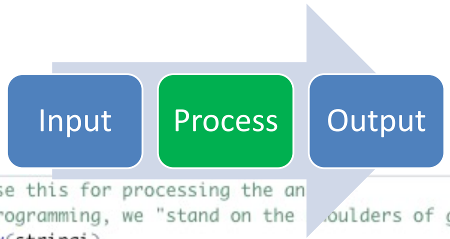
Package	Purpose
ggplot2	Produce graphics for data
dplyr	Analysis of data held in tibbles/data frames
aimsir17	2017 Weather data for Ireland
stringi	For manipulating strings

Challenge 2.2

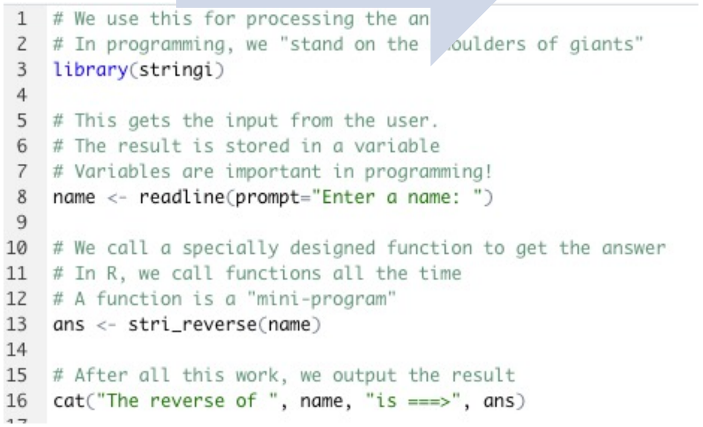
- Install the package aimsir17
- Check out the variable:
  - observations
  - stations
  - eirgrid17
- Explore Storm Ophelia
  - https://github.com/JimDuggan/aimsir17/tree/master/data-raw/Examples/02%20Storm%20Ophelia
  - 16th October 2017



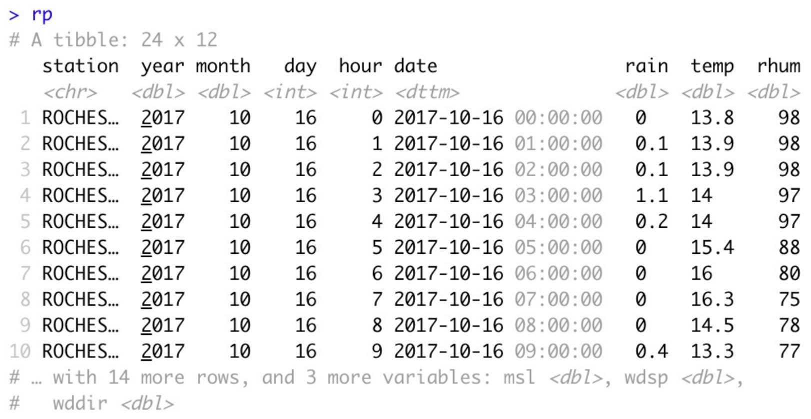
Summary



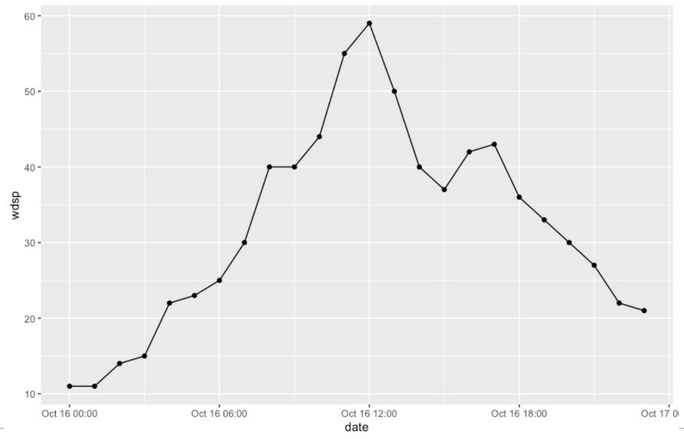
Topic	Description
1	Introduction to R and R Studio Cloud
2	A program in R
3	The tibble – a way of storing information
4	Data Visualisation I
5	Data Transformation I
6	Running a Script in R
7	Data Visualisation II
8	Data Transformation II
9	Exploring Data
10	Communicating Results



```r
# We use this for processing the an
# In programming, we "stand on the  oulders of giants"
library(stringi)

# This gets the input from the user.
# The result is stored in a variable
# Variables are important in programming!
name <- readline(prompt="Enter a name: ")

# We call a specially designed function to get the answer
# In R, we call functions all the time
# A function is a "mini-program"
ans <- stri_reverse(name)

# After all this work, we output the result
cat("The reverse of ", name, "is ===>", ans)
```



```
> rp
# A tibble: 24 x 12
   station  year month   day  hour date                 rain  temp  rhum
   <chr>   <dbl> <dbl> <int> <int> <dttm>              <dbl> <dbl> <dbl>
 1 ROCHES…  2017    10    16     0 2017-10-16 00:00:00   0    13.8    98
 2 ROCHES…  2017    10    16     1 2017-10-16 01:00:00   0.1  13.9    98
 3 ROCHES…  2017    10    16     2 2017-10-16 02:00:00   0.1  13.9    98
 4 ROCHES…  2017    10    16     3 2017-10-16 03:00:00   1.1  14      97
 5 ROCHES…  2017    10    16     4 2017-10-16 04:00:00   0.2  14      97
 6 ROCHES…  2017    10    16     5 2017-10-16 05:00:00   0    15.4    88
 7 ROCHES…  2017    10    16     6 2017-10-16 06:00:00   0    16      80
 8 ROCHES…  2017    10    16     7 2017-10-16 07:00:00   0    16.3    75
 9 ROCHES…  2017    10    16     8 2017-10-16 08:00:00   0    14.5    78
10 ROCHES…  2017    10    16     9 2017-10-16 09:00:00   0.4  13.3    77
# … with 14 more rows, and 3 more variables: msl <dbl>, wdsp <dbl>,
#   wddir <dbl>
```


wdsp
Oct 16 00:00
Oct 16 06:00
Oct 16 12:00
Oct 16 18:00
Oct 17 0
date