

Recap II

The R Bootcamp
Twitter: [@therbootcamp](https://twitter.com/therbootcamp)
September 2017

Essentials of the R language

"To understand computations in R, two slogans are helpful:

(1) Everything that exists is an object
and

(2) everything that happens is a
function call."



John Chambers

Author of S and developer of R

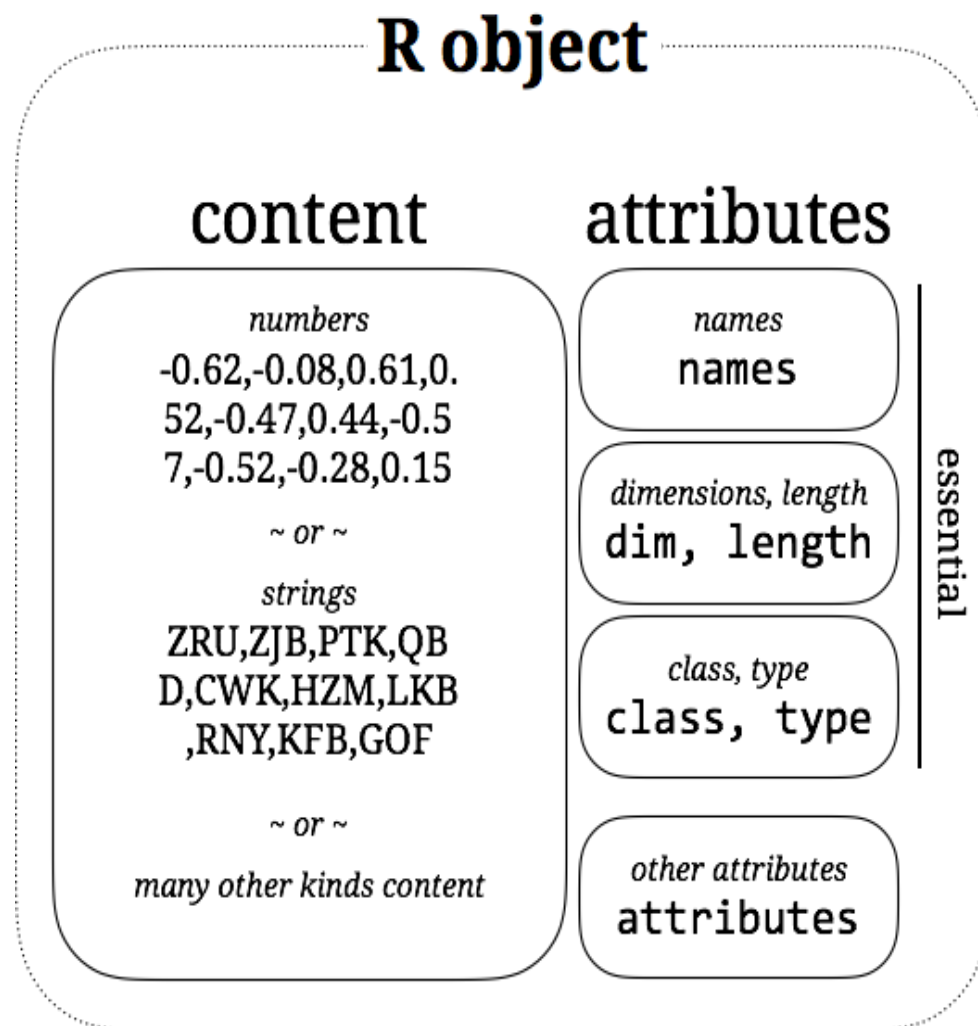
statweb.stanford.edu

Objects

"Everything in R is an object"

John Chambers

- R's objects are have **content** and **attributes**.
- The **content** can be **anything** from numbers or strings to functions or complex data structures.
- Attributes often encompass **names**, **dimensions**, and the **class** or **type** of the object, but other attributes are possible.
- Practically all data objects are equipped with those **three essential attributes**.



Accessing & changing complex objects

Data frames can be accessed **exactly like lists**. In addition, data frames allow for a matrix-like access using **single bracket** [. Note however that selecting rows using single bracket returns a data frame, whereas for selecting columns returns a vector.

```
# retrieve elements from list
my_df <- data.frame('v_1'=c('A','B','C'),
                    'v_2'=c(1,2,3))
my_df[1] ; my_df[[1]] ; my_df[['v_1']]
```

```
##      v_1
## 1     A
## 2     B
## 3     C

## [1] A B C
## Levels: A B C

## [1] A B C
## Levels: A B C
```

```
# retrieve elements from list
my_df <- data.frame('v_1'=c('A','B','C'),
                    'v_2'=c(1,2,3))
my_df[1,] ; my_df[,1] ; my_df[1,2]
```

```
##      v_1 v_2
## 1     A    1

## [1] A B C
## Levels: A B C

## [1] 1
```

Calls, assignments, and expressions

In R every action is a function call. Specifically, R programs advance by **passing on arguments to functions**, **calling the function**, and **receiving and storing its output**. And this goes deep, many operations are functions in disguise.

```
# defining a function - arithm. mean
my_fun <- function(x, b){ x * b }

# define some data
my_data <- c(1, 5, 7, 3)

# pass on arguments and call function
my_fun(my_data, 5)
```

```
## [1] 5 25 35 15
```

```
# store output by assignment
my_out <- my_fun(my_data, 5)
```

```
# a basic expression
2 + 2
```

```
## [1] 4
```

```
# is also a function
'+'(2,2)
```

```
## [1] 4
```

Help

An facilitator for using R are **help files** and **vignettes**. Help files are required documentations for every R function and package published on **CRAN**. Don't worry if help files may appear cryptical, however, over time you will realise how helpful they are. **Vignettes** are long tutorials sometimes provided by the authors of a package.

```
# To access help files
help("name_of_function")
?name_of_function

# find help files
??name_of_function

# To list and access vignettes
vignette(package="name_of_package")
vignette(package="name_of_vignette")
```

404

File not found

The site configured at this address does not contain the requested file.

If this is your site, make sure that the filename case matches the URL. For root URLs (like `http://example.com/`) you must provide an `index.html` file.

[Read the full documentation](#) for more information about using **GitHub Pages**.

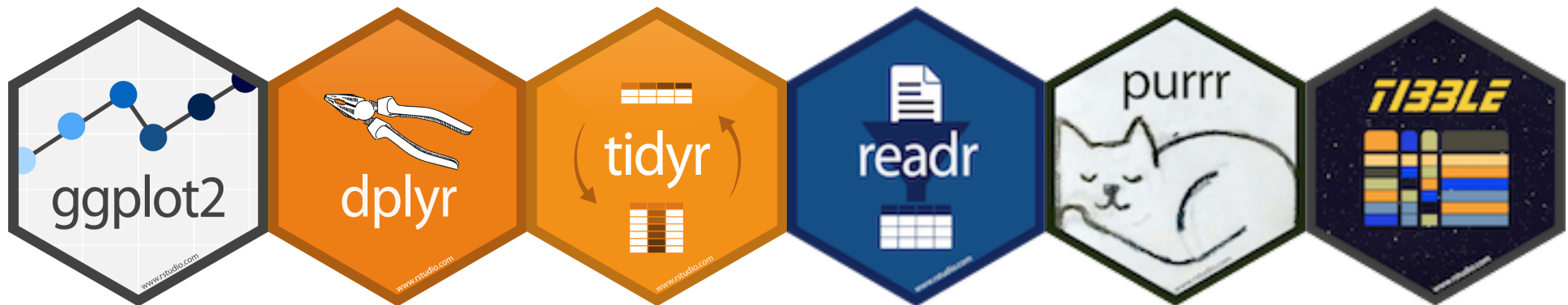
[GitHub Status](#) — [@githubstatus](#)



The almighty tidyverse

Among its many packages, R contains a collection of high-performance, easy-to-use packages (libraries) designed specifically for handling data know as the **tidyverse**. The tidyverse includes:

1. ggplot2 -- creating graphics.
2. dplyr -- data manipulation.
3. tidyr -- tidying data.
4. readr -- read wild data.
5. purrr -- functional programming.
6. tibble -- modern data frame.



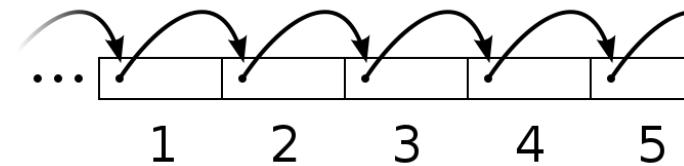
dplyr

dplyr is a combination of 3 things:

1. **objects** like dataframes
2. **verbs** that **do** things to objects.
3. **pipes** `%>%` that string together objects and verbs



Sequential



dplyr is meant to be sequential and work like language

Take data X, then do Y, then do Z...

Here's the basic structure of dplyr in action

```
data %>%           # Start with data, and THEN  
  VERB1 %>%        # Do VERB1, (and THEN)  
  VERB2 %>% ...    # Do VERB2, (and THEN)
```


Common dplyr verbs

verb	action	example
<code>filter()</code>	Select rows based on some criteria	<code>filter(age > 40 & sex == "m")</code>
<code>arrange()</code>	Sort rows	<code>arrange(date, group)</code>
<code>select()</code>	Select columns (and ignore all others)	<code>select(age, sex)</code>
<code>rename()</code>	Rename columns	<code>rename(DATE_MONTHS_X24, date)</code>
<code>mutate()</code>	Add new columns	<code>mutate(height.m = height.cm / 100)</code>
<code>case_when()</code>	Recode values of a column	<code>sex.n = case_when(sex == 0 ~ "m", sex == 1 ~ "f")</code>
<code>group_by(), summarise()</code>	Group data and then calculate summary statistics	<code>group_by(treatment) %>% summarise(...)</code>

Inferential Statistics

Formula

- Many tests allow you to include a formula argument

```
| formula = y ~ a + b + ...
```

Means...

```
| Model a dependent variable y as a  
| function of a and b and ...
```

- Formulas go together with dataframes data containing all variables in the formula, and optional subset arguments to specify which cases in data to include.

General structure of a hypothesis test and formula

```
my.test(formula = y ~ a + b, # Formula  
        data = my.data,      # Dataframe  
        ...                   # Additional  
        )
```

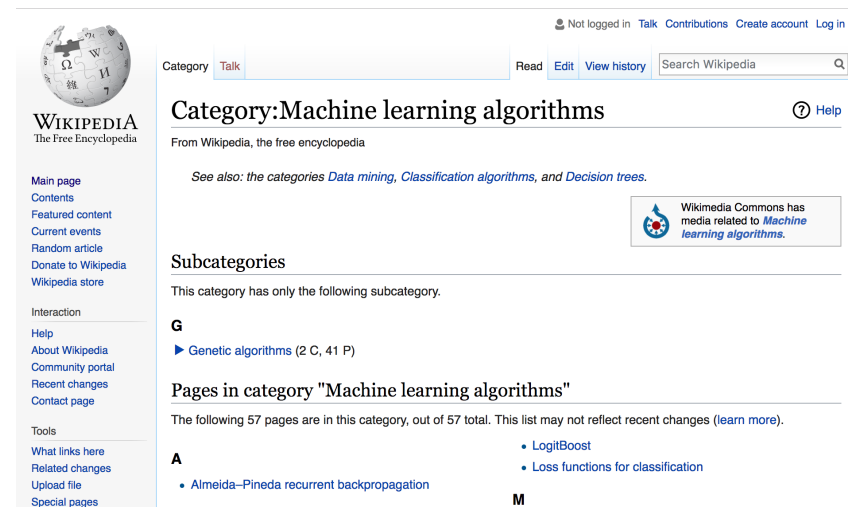
- y is the dependent variable (e.g.; age), a and b are independent variables
- data is a dataframe containing the variables in formula; (y, a, b)
- ... additional arguments specific to test

What machine learning algorithms are there?

- There are hundreds of **machine learning algorithms** from many different fields.
 - E.g.; Computer vision, natural language processing, reinforcement learning, graphical models, etc.
- In this section, we will focus on 4:

Algorithm	Complexity?
Regression	Low / Medium
Decision Trees	Low
Random Forests	High
Support Vector Machines	High

- Wikipedia lists 57 *Categories* of machine learning algorithms, each with dozens of examples.



The screenshot shows the Wikipedia category page for "Machine learning algorithms". The page includes a sidebar with navigation links, a search bar, and a list of subcategories and pages. The main content area displays the category title, a brief description, and a list of subcategories. The "Pages in category" section lists 57 pages, with a few examples shown: LogitBoost, Loss functions for classification, and Almeida-Pineda recurrent backpropagation.

Category: Machine learning algorithms

From Wikipedia, the free encyclopedia

See also: the categories *Data mining*, *Classification algorithms*, and *Decision trees*.

Subcategories

This category has only the following subcategory.

G

- ▶ Genetic algorithms (2 C, 41 P)

Pages in category "Machine learning algorithms"

The following 57 pages are in this category, out of 57 total. This list may not reflect recent changes ([learn more](#)).

A

- Almeida-Pineda recurrent backpropagation

M

- LogitBoost
- Loss functions for classification