HEALTH
UNIVERSITY OF UTAH

R Syntax & Data Structures

Andrew Redd, PhD.

R Bootcamp 2020

This will be short, I promise.

Basic R Syntax

The absolute most important part of syntax …

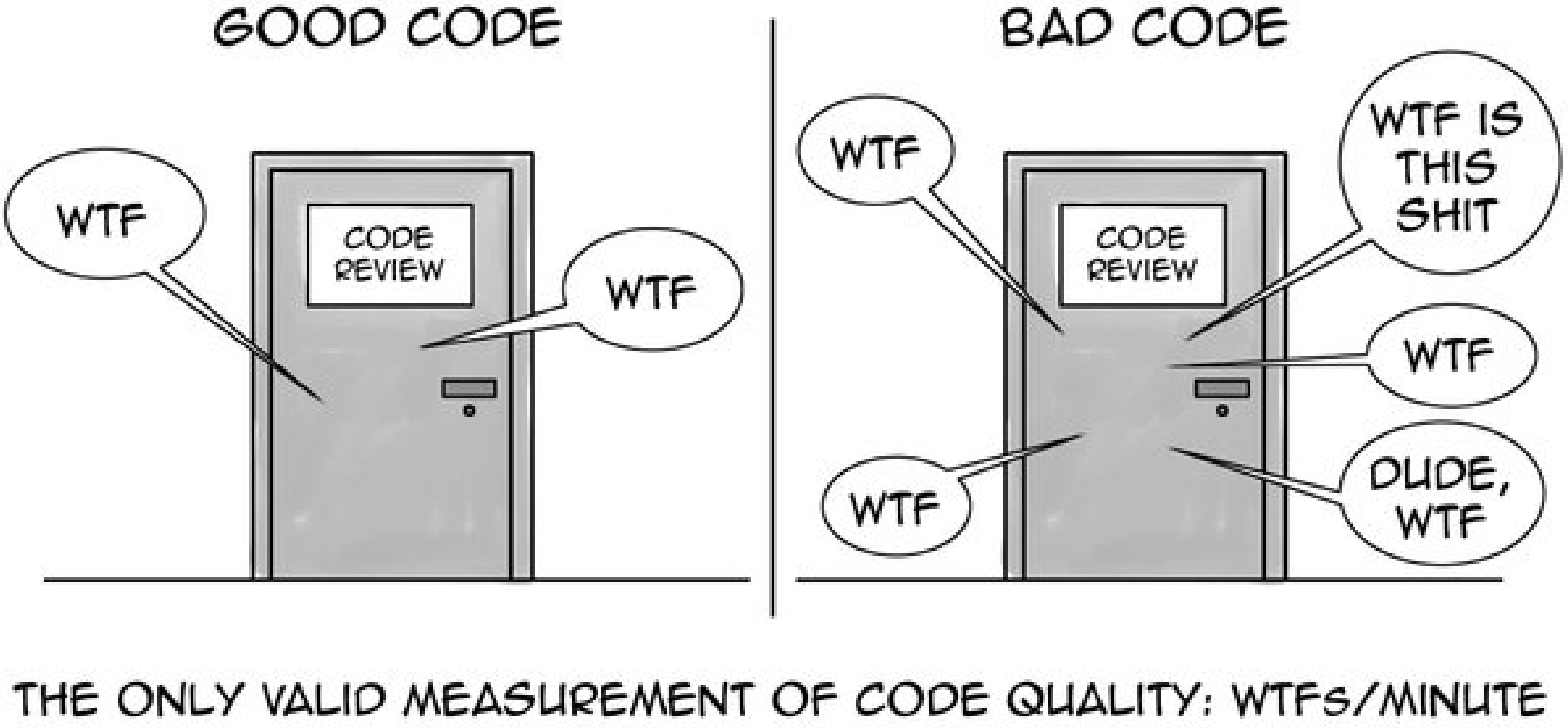
Comments

Comments are started by the # character

```
# Comments use the '#' character
```

Some comments have special meaning.

- `#'` documentation comment
- `#!` sometimes configuration comments
- `#<` hey i’m talking about this comment.


GOOD CODE
WTF
CODE REVIEW
WTF
BAD CODE
WTF
WTF IS THIS SHIT
CODE REVIEW
WTF
WTF
DUDE, WTF
THE ONLY VALID MEASUREMENT OF CODE QUALITY: WTFs/MINUTE

Assignments

- `a <- 1` assigns the value `1` to the variable `a`.
- `a = 1` does the same, *generally*.
    - The generally is why this is also generally discouraged.

Variables

- Variables “can” include:
  - letters (case-sensitive)
  - numbers, as long as it does not start with a number
  - period, if it starts with a period it is ‘hidden’
  - underscore, but may not start with underscore
- But if you want to be fancy use back ticks ` ` ` and you can do whatever you want.

```
`A variable name that is actually informative` <-
    "A string that is not"
`A variable name that is actually informative`
```

```
## [1] "A string that is not"
```

Functions

- create a function with

```r
hello_world <- function(who = 'world', how = 'hello'){
    print(paste(how, who))
}
```

- Now call the function with

```r
hello_world()
```

```
## [1] "hello world"
```

Data Types: Vectors

- In R everything is a vector
- Sequential vectors can be created with `:`

```
a <- 1:5
```

- Create longer vectors by combining smaller vectors

```
b <- c(a, 6, 7, 8)
```

Types, i.e. Mode

- integer
- numeric
- logical
- list (i.e. anything)
- *language*

Vector Operations

```
# Most operations are obvious
length(b)
```

```
## [1] 8
```

```
sort(b)
```

```
## [1] 1 2 3 4 5 6 7 8
```

```
class(b)
```

```
## [1] "numeric"
```

Indexing (2-types)

- [- subsetting
    - preserves class
    - multiple indices

```
# A list holds anything
a <- list(first=1L, second=2, FALSE)
```

```
# by position
class(a[1])
```

```
## [1] "list"
```

```
# by name
class(a['second'])
```

```
## [1] "list"
```

- [[- extraction
    - no guarantee of class
    - single index

```
# by position
class(a[[1]])
```

```
## [1] "integer"
```

```
# by name
class(a[['second']])
```

```
## [1] "numeric"
```

$ Indexing

lists may also be indexed by the `$` sign.

But be careful not all vectors can be.

```
a$first
```

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

```
a$second
```

```
## [1] 2
```

This comes in useful for data.

Data Types

Data for our purpose is:

- A (named) list of variables,
  - where all variables have the same length;
- Rectangular with and rows as observations.
  - columns as variables `ncol(iris)`=5
  - and rows as observations `nrow(iris)`=150

SPECIAL: The Pipe `%>%`

The pipe is a newer but very handy tool in R.

Use it to tie multiple short statements together into a single complex statement that is easy to understand and use. It comes from the `magrittr` package but it is more common to use it through the tidyverse package, which will be covered in detail later.

```
library(tidyverse)
iris %>% #< take iris data
    filter(Species=='setosa') %>% #< perform a filter
    nrow() #< count the rows.
```

```
## [1] 50
```