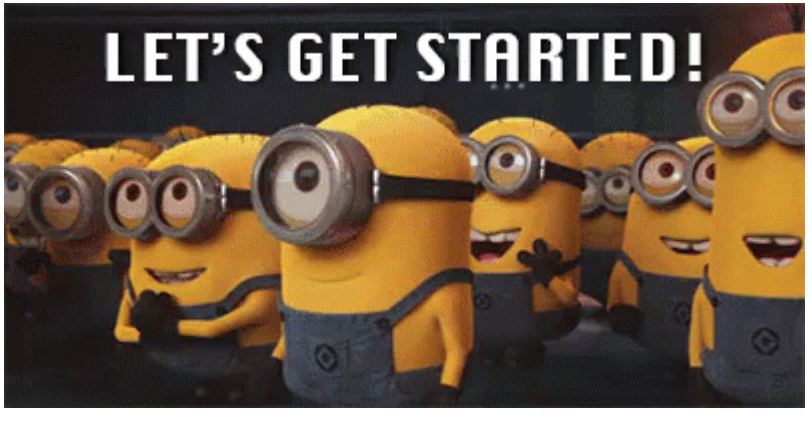
Advanced R - Hadley Wickham - Ch 4

R Ladies Netherlands Booookclub

Martine Jansen, May 12, 2020

Welcome!

- This is joint effort between RLadies Nijmegen, Rotterdam, 's-Hertogenbosch (Den Bosch), Amsterdam and Utrecht
- We meet every 2 weeks to go through a chapter
- Use the HackMD to present yourself, ask questions and see your breakout room
- We split in breakout rooms after the presentation, and we return to the main jitsi link after xx min
- There are still possibilities to present a chapter :) Sign up at https://rladiesnl.github.io/book_club/
- https://advanced-r-solutions.rbind.io/ has some anwers and we could PR the ones missing
- The R4DS book club repo has a Q&A section.https://github.com/r4ds/bookclub-Advanced_R



Contents

Selecting multiple elements [ ]

From vectors, with indices, positive OR negative

```
x[<expression resulting in indices>]
```

Positive indices

```
x<- c("g", "a", "c", "f", "b")
# Get item 2 and item 5
x[c(2,5)]
```

```
[1] "a" "b"
```

Negative indices

```
x<- c("g", "a", "c", "f", "b")
# Get everything EXCEPT item 2 and item 5
x[-c(2,5)]
```

```
[1] "g" "c" "f"
```

You can be creative with indices:

Repeat them

```
x<- c("g", "a", "c", "f", "b")
# repeat the first 2 items, 5 times
x[rep(1:2, times = 5)]
```

```
[1] "g" "a" "g" "a" "g" "a" "g" "a" "g" "a"
```

Use it to order a vector

```
x<- c("g", "a", "c", "f", "b")
# order the vector
x[order(x)]
```

```
[1] "a" "b" "c" "f" "g"
```

You do not HAVE to use indices

```
x<- c("g", "a", "c", "f", "b")
# return the whole vector
x[]
```

```
[1] "g" "a" "c" "f" "b"
```

From vectors, with logical vectors

`x[<expression resulting in T/F vector>]`

Specify a T/F vector

```
x<- c("g", "a", "c", "f", "b", "b")
# Get all the items "a" and "b"
x[c(F,T,F,F,T,T)]
```

```
[1] "a" "b" "b"
```

What if T/F vector has different length?

```
x<- c("g", "a", "c", "f", "b", "g", "a")
x[c(F,T)]
```

```
[1] "a" "f" "g"
```

The F/T vector will be recycled:
F T F T F T F
But be very careful with recyling of vectors longer than length 1

Specify an expression RESULTING in a T/F vector

```
x<- c("g", "a", "c", "f", "b", "b")
# Get all the items "a" and "b"
x[x %in% c("a", "b")]
```

```
[1] "a" "b" "b"
```

Another example of that

```
x<- c("Anna", "Bea", "Corry-Ann", "Annabel", "Ann", "Quinta")
# Get all the items that have "Ann" in them
x[grepl("Ann",x)]
```

```
[1] "Anna"      "Corry-Ann" "Annabel"   "Ann"
```

And another example of that

```
x<- c("Anna", "Bea", "Corry-Ann", "Annabel", "Ann", "Quinta")
x[grepl("Beatrix",x)]
```

```
character(0)
```

In the last example, there were no matches for "Beatrix".
This resulted in a T/F vector with every value equal to FALSE.
And hence nothing (vector of length 0) in the output.

From lists

```
lucky_list <- list(lucky_numbers = 1:9,
                   lucky_words = c("cat", "cure"))
lucky_list
```

```
$lucky_numbers
[1] 1 2 3 4 5 6 7 8 9

$lucky_words
[1] "cat"  "cure"
```

subset by indices

```
lucky_list[2]   # keeps being a list
```

```
$lucky_words
[1] "cat"  "cure"
```

Or subset by using the name(s)

```
lucky_list["lucky_words"]
```

```
$lucky_words
[1] "cat"  "cure"
```

From matrices and arrays, with indices

Subsetting matrix A : `A[<row indices>, <column indices>]`

```
A <- matrix(1:9, nrow = 3)
colnames(A) <- c(LETTERS[4:6])  # :) vector subsetting
A
```

```
     D E F
[1,] 1 4 7
[2,] 2 5 8
[3,] 3 6 9
```

```
A[1:2, c("E", "F")]
```

```
     E F
[1,] 4 7
[2,] 5 8
```

```
A[1:2,]
```

```
     D E F
[1,] 1 4 7
[2,] 2 5 8
```

Consistant with vector: no indices, return all

Subsetting with matrix

Again matrix A:

```
     D E F
[1,] 1 4 7
[2,] 2 5 8
[3,] 3 6 9
```

From matrix A we want:

- first row third value
- third row, second value

Specify this in matrix B:

```
     [,1] [,2]
[1,]    1    3
[2,]    3    2
```

```
A[B]
```

```
[1] 7 6
```

Be ware: this result is a **vector**, not a matrix.
[ ] simplifies to the lowest possible dimensionality.
Lists stay lists

From dataframes and tibbles

data.frame df

```
  x y z
1 1 1 a
2 2 2 b
3 3 3 c
```

Subsetting like a list

```
df["z"]
```

```
  z
1 a
2 b
3 c
```

Subsetting like a matrix

```
df[,"z"]
```

```
[1] "a" "b" "c"
```

Again, loss of dimensions!

Preserving the dimensions: tibbles :)

data.frame

```
dframe <-
  data.frame(x = 1:3,
             y = 1:3,
             z = c("a", "b", "c"))
dframe
```

```
  x y z
1 1 1 a
2 2 2 b
3 3 3 c
```

```
dframe[,"z"]
```

```
[1] "a" "b" "c"
```

tibble

```
dtibble <-
  tibble(x = 1:3,
         y = 1:3,
         z = c("a", "b", "c"))
dtibble
```

```
# A tibble: 3 x 3
      x     y z
  <int> <int> <chr>
1     1     1 a
2     2     2 b
3     3     3 c
```

```
dtibble[,"z"]
```

```
# A tibble: 3 x 1
  z
  <chr>
1 a
2 b
3 c
```

Preserving the dimensions: drop = FALSE

data.frame

```
dframe <-
  data.frame(x = 1:3,
             y = 1:3,
             z = c("a", "b", "c"))
dframe
```

```
  x y z
1 1 1 a
2 2 2 b
3 3 3 c
```

```
dframe[,"z"]
```

```
[1] "a" "b" "c"
```

```
dframe[,"z", drop = FALSE]
```

```
  z
1 a
2 b
3 c
```

matrix

```
A
```

```
     D E F
[1,] 1 4 7
[2,] 2 5 8
[3,] 3 6 9
```

```
A[2,]
```

```
D E F
2 5 8
```

```
A[2, ,drop = FALSE]
```

```
     D E F
[1,] 2 5 8
```

Selecting single elements [[ ]]

```
x[[<single positive value or a string>]]
```

```
lucky_list <- list(lucky_numbers = 1:9,
                   lucky_words = c("cat", "cure"))
```

subset by index with SINGLE [ ]

```
lucky_list[2]
```

```
$lucky_words
[1] "cat"  "cure"
```

keeps being a list

subset by index with DOUBLE [[ ]]

```
lucky_list[[2]]
```

```
[1] "cat"  "cure"
```

no longer a list, outer layer removed, like an unwrapped present

Subsubsetting

```
lucky_list
```

```
$lucky_numbers
[1] 1 2 3 4 5 6 7 8 9

$lucky_words
[1] "cat"  "cure"
```

How to extract the word "cure"?
It is the second item from the item in the second part of the list.
First extract the item in the second part of the list:

```
lucky_list[[2]]
```

```
[1] "cat"  "cure"
```

Then extract the second item:

```
lucky_list[[2]][2]
```

```
[1] "cure"
```

Selecting a single variable from dataframe

```
dframe <-
  data.frame(x1 = 1:3,
             y2 = 4:6,
             z3 = c("a", "b", "c"))
```

```
dframe["z3"]
```

```
  z3
1  a
2  b
3  c
```

```
dframe[,"z3"]
```

```
[1] "a" "b" "c"
```

```
dframe[["z3"]]
```

```
[1] "a" "b" "c"
```

```
dframe[[3]]
```

```
[1] "a" "b" "c"
```

```
dframe$z3
```

```
[1] "a" "b" "c"
```

```
dframe$z
```

```
[1] "a" "b" "c"
```

(partial matching)

Some advice from the author

- Also use [[ ]] for vectors when extracting a single element, so [[ ]] grows to be synonym to extracting a single element
- Be careful with partial matching. Either use:
  - tibbles instead of data.frames :)
  - options(warnPartialMatchDollar = TRUE)

```
dframe$z
```

```
[1] "a" "b" "c"
```

```
options(warnPartialMatchDollar = TRUE)
dframe$z
```

```
Warning in dframe$z: partial match of 'z' to 'z3'
```

```
[1] "a" "b" "c"
```

Missing and OOB indices

```
X <- c(1, 2, 3)
L <- list(1,2,3)
```

```
X[[4]]
```

```
Error in X[[4]]: subscript out of bounds
```

```
L[[4]]
```

```
Error in L[[4]]: subscript out of bounds
```

```
NULL[[4]]
```

```
NULL
```

```
X[["a"]]
```

```
Error in X[["a"]]: subscript out of bounds
```

```
L[["a"]]
```

```
NULL
```

```
NULL[["a"]]
```

```
NULL
```

Inconsistent behaviour.
Solution: `purr:pluck()` and `purr::chuck()`

purr::pluck( ) and purrr::chuck( )

Element is missing?

Pluck() --> NULL ALWAYS (or the default)
Chuck() --> error ALWAYS

```
X[["a"]]
```

```
Error in X[["a"]]: subscript out of bounds
```

```
L[["a"]]
```

```
NULL
```

```
purrr::pluck(X, "a")
```

```
NULL
```

```
purrr::pluck(L,"a")
```

```
NULL
```

```
purrr::chuck(L,"a")
```

```
Error: Index 1 is attempting to pluck from an un
```

Subsetting and assignment: subassignment

Vector

```
x <- c(1:5)
```

```
[1] 1 2 3 4 5
```

```
x[3:5] <- c(30, 40, 50)
```

```
[1]  1  2 30 40 50
```

```
x[7] <- 7
```

```
[1]  1  2 30 40 50 NA  7
```

```
x[2] <- NA
```

```
[1]  1 NA 30 40 50 NA  7
```

```
x[1] <- NULL
```

```
Error in x[1] <- NULL: replacement has length zero
```

List

```
L <- list(a = 1:2, b = letters[13:17])
```

```
$a
[1] 1 2

$b
[1] "m" "n" "o" "p" "q"
```

```
L[["c"]] <- c(10:12)
```

```
$a
[1] 1 2

$b
[1] "m" "n" "o" "p" "q"

$c
[1] 10 11 12
```

```
L[[1]] <- NA
```

```
$a
[1] NA

$b
[1] "m" "n" "o" "p" "q"

$c
```

```
L[[1]] <- NULL # removes
```

```
$b
[1] "m" "n" "o" "p" "q"

$c
[1] 10 11 12
```

Applications

Lookup tables

```
x <- c("a", "p", "a", "p", "p")
lookup <- c(a = "absent", p = "present")
```

```
lookup
```

```
       a         p
"absent" "present"
```

```
# "subset" lookup with a lot of indices
lookup[x]
```

```
       a         p         a         p         p
"absent" "present"  "absent" "present" "present"
```

```
# and now without the names
unname(lookup[x])
```

```
[1] "absent"  "present" "absent"  "present" "present"
```

Matching and merging

```
info
```

```
  grade      desc  fail
1     A Excellent FALSE
2     B      Good FALSE
3     C      Poor  TRUE
```

```
# For each of these grades, get the info from dataframe info
grades <- c("A", "B", "B", "C", "A")
```

```
# First, find where in the dataframe these grades are:
id <- match(grades, info$grade)
id
```

```
[1] 1 2 2 3 1
```

```
# Second, subset dataframe `info` with these indices:
# (get only the found rows, but all the columns)
info[id, ]
```

```
    grade      desc  fail
1       A Excellent FALSE
2       B      Good FALSE
2.1     B      Good FALSE
3       C      Poor  TRUE
1.1     A Excellent FALSE
```

Random sampling and bootstraps

```
x <- 1:5
x
```

```
[1] 1 2 3 4 5
```

Reorder randomly

```
x[sample(length(x))]
```

```
[1] 4 3 2 5 1
```

Sample

```
x[sample(length(x), 2)]
```

```
[1] 2 3
```

Sample with replacement

```
x[sample(length(x), replace = TRUE)]
```

```
[1] 1 3 5 5 2
```

Ordering (with indices)

```
x <- c("m", "a", NA, "z")
x
```

```
[1] "m" "a" NA  "z"
```

```
# the order the elements are in
order(x)
```

```
[1] 2 1 4 3
```

```
# order x in the right order
x[order(x)]
```

```
[1] "a" "m" "z" NA
```

```
# the order the elements are in, reverse order
order(x, decreasing = TRUE)
```

```
[1] 4 1 2 3
```

```
# order x in the right order
x[order(x, decreasing = TRUE)]
```

```
[1] "z" "m" "a" NA
```

Expanding aggregated counts

```
df <- data.frame(item = c("coffee", "tea"),
                 amount = c(2,3))
```

```
df
```

```
    item amount
1 coffee      2
2    tea      3
```

```
df[rep(1:nrow(df), times = df$amount),]
```

```
      item amount
1   coffee      2
1.1 coffee      2
2      tea      3
2.1    tea      3
2.2    tea      3
```

Alternative expanding dplyr::uncount( ) (not in book)

```
df
```

```
    item amount
1 coffee      2
2    tea      3
```

```
library(tidyverse)
df %>% uncount(amount, .remove = FALSE)
```

```
      item amount
1   coffee      2
1.1 coffee      2
2      tea      3
2.1    tea      3
2.2    tea      3
```

```
library(tidyverse) # or, like it should, to avoid confusion
df %>% uncount(amount)
```

```
      item
1   coffee
1.1 coffee
2      tea
2.1    tea
2.2    tea
```

Removing columns from df (character subsetting)

```
df
```

```
    item amount
1 coffee      2
2    tea      3
```

```
# set unwanted ones to NULL
df$amount <- NULL
df
```

```
    item
1 coffee
2    tea
```

```
# only keep want you want
df <- df[c("item")]
```

```
    item
1 coffee
2    tea
```

```
# exclude from colnames those you do not want
setdiff(names(df), "amount")
df <- df[setdiff(names(df), "amount")]
df
```

```
[1] "item"
```

```
    item
1 coffee
2    tea
```

Tidyverse alternative selecting columns (not in book)

```
# select the columns you want
df <- df %>%
  select(item)
df
```

```
    item
1 coffee
2    tea
```

```
# unselect the columns you do not want
df <- df %>%
  select(-amount)
df
```

```
    item
1 coffee
2    tea
```

Selecting rows based on a condition(logical subsetting)

```
df <- data.frame(item = c("coffee", "tea", "water"),
                 amount = c(2,3, 2))
df
```

```
    item amount
1 coffee      2
2    tea      3
3  water      2
```

```
# select all rows with amount == 2
df[df$amount == 2,]
```

```
    item amount
1 coffee      2
3  water      2
```

```
# select all rows with amount == 2 and item == "water"
df[df$amount == 2 & df$item == "water",]
```

```
   item amount
3 water      2
```

```
# same with tidyverse - not in book
df %>% filter(amount == 2, item == "water")
```

```
   item amount
1 water      2
```

Boolean algebra versus sets

```
x <- c(3, 5, 2, 6, 2, 7)
```

```
# gives T/F
x < 4
```

```
[1]  TRUE FALSE  TRUE FALSE  TRUE FALSE
```

```
# gives indices
which(x < 4)
```

```
[1] 1 3 5
```

```
# subsetting with T/F, same result as with indices
x[x<4]
```

```
[1] 3 2 2
```

```
# subsettubg with indices, same result as with T/F vector
x[which(x < 4)]
```

```
[1] 3 2 2
```

Major code block on page 93 - 1/5

which: gives the TRUE indices of a logical object

```
which(c(TRUE, FALSE, FALSE, TRUE))
```

```
[1] 1 4
```

```
# code between ( ) automatically prints result
(x1 <- 1:10 %% 2 == 0)  # F/T depending on being a 2 fold
```

```
 [1] FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE
```

```
(x2 <- which(x1))
```

```
[1]  2  4  6  8 10
```

```
(y1 <- 1:10 %% 5 ==0)   # F/T depending on being a 5 fold
```

```
 [1] FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE
```

```
(y2 <- which(y1))
```

```
[1]  5 10
```

Major code block on page 93 - 2/5

intersect of X and Y

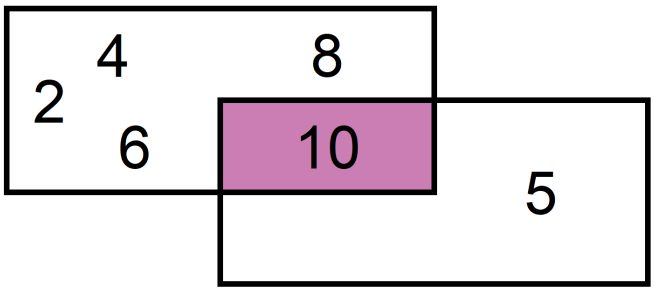
$X\&Y, X \cap Y, x \wedge y$

```
x2
```

```
[1]  2  4  6  8 10
```

```
y2
```

```
[1]  5 10
```




```
x1 & y1
```

```
 [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
```

```
intersect(x2,y2)
```

```
[1] 10
```

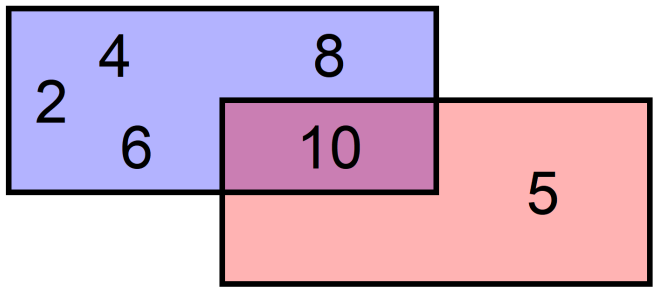
Major code block on page 93 - 3/5

union of X and Y

$$X|Y, X \cup Y, X \vee Y$$

```
x2
```

```
[1]  2  4  6  8 10
```




```
y2
```

```
[1]  5 10
```

```
x1 | y1
```

```
 [1] FALSE  TRUE FALSE  TRUE  TRUE  TRUE FALSE  TRUE FALSE  TRUE
```

```
union(x2,y2)
```

```
[1]  2  4  6  8 10  5
```

Major code block on page 93 - 4/5

setdiff of X and Y

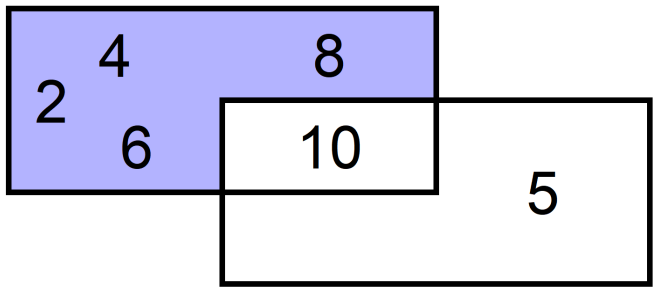
$X\&!Y, X \wedge \neg Y$

```
x2
```

```
[1]  2  4  6  8 10
```

```
y2
```

```
[1]  5 10
```




```
x1 & !y1
```

```
 [1] FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE FALSE
```

```
setdiff(x2,y2)
```

```
[1] 2 4 6 8
```

Major code block on page 93 - 5/5

Exclusive OR of X and Y

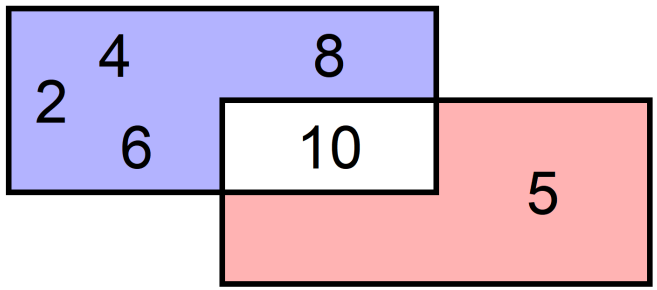
xor(X,Y), $(X \wedge \neg Y) \cup (\neg X \wedge Y)$

```
x2
```

```
[1]  2  4  6  8 10
```

```
y2
```

```
[1]  5 10
```




```
xor(x1, y1)
```

```
 [1] FALSE  TRUE FALSE  TRUE  TRUE  TRUE FALSE  TRUE FALSE FALSE
```

```
setdiff(union(x2, y2), intersect(x2, y2))
```

```
[1] 2 4 6 8 5
```

The puzzle!

Answer the 18 questions.
Transfer the letters found to the corresponding numbers below:

8	17	4	18

11	10	1	2

7	3	6

5	12	14	13	16	15	9

Question 1

```
x <- c(3, 1, 4, 1, 5, 9, 2)
```

What is the result of `x[c(3,5)]`?

R `c(1,4,1,9,2)`
P `c(3,1,1,9,2)`
V `c(3,5)`
N `c(4,5)`

Question 2

```
x <- c(3, 1, 4, 1, 5, 9, 2)
```

Which expression will result in `c(3, 1, 1, 5, 9)`?

E `x[x%%2 == 1]`
H `x[c(T,T,F,T,T,T,F)]`
G both expressions above will do that
D none of the expressions above will do that

Question 3

We want to create matrix $A = \begin{pmatrix} 3 & 1 & 4 & 1 \\ 5 & 9 & 2 & 6 \\ 5 & 3 & 8 & 9 \end{pmatrix}$

with this r code: `A <- matrix(c(3,1,4,1,5,9,2,6,5,3,8,9), nrow = 3)`.

Is this correct?

N No!
Y Yes!

Question 4

What are the dimensions of `A[,1]`?

A 1 3
R 3 1
V There are no dimensions

Question 5

What is the length of `A[,1]`?

N 1
P 3
R There is no length

Question 6

What are the dimensions of `A[1:2,-c(1,2)]`

D 2 2
E 4 1 2 6
F There are no dimensions

Question 7

`x <- c("a", "b", "c")`.
What is the result of `x[4]`?

O character(0)
A NA
E 1

Question 8

What is the result of `x[[-1]]`?

L Error in x[[-1]] : invalid negative subscript in get1index
M Error in x[[-1]] : subscript out of bounds
N "a"
O "b" "c"

Question 9

```
L <- list(lucky_numbers = 1:9,
       lucky_words = c("cat", "cure"),
       happy_times = data.frame(
         location = c("Online", "Friesland", "Pub"),
         activity = c("R-Ladies Boookclub", "sailing", "talking to friends")))
L
```

```
$lucky_numbers
[1] 1 2 3 4 5 6 7 8 9

$lucky_words
[1] "cat"  "cure"

$happy_times
   location           activity
1    Online R-Ladies Boookclub
2 Friesland            sailing
3       Pub talking to friends
```

Which code would extract "R-Ladies Boookclub"?

R `L[[3]][1,2]`
S `L[1,2][[3]]`
R `L[[3]][[2]][[1]]`
S `L[[1]][[2]][[3]]`

Question 10 (exercise 4.3.5)

`mod` is a linear regression model, which explains `mpg` from `wt`:

```
mod <- lm(mpg ~ wt, data = mtcars)
```

What kind of R object is `mod` ?

M data.frame
N vector
O list

Question 11

What code will extract from `mod` the residual degrees of freedom as an integer ?

I `mod$df.residual`
J `mod[["df.residual"]]`
K `mod["df.residual"]`
L first two answers are correct
M first three answers are correct
N none of the answers is correct

Question 12

```
mod <- lm(mpg ~ wt, data = mtcars)
summ_mod <- summary(mod)
```

What code with "$" will extract the R^2 from the model summary?

Take of this code the 16th letter from the start. That is the letter for slot 12.

Question 13

How would you remove the residuals from the summary? (Not that this is a useful skill to have :))

S `summ_mod[["residuals"]] <- NULL`
T `summ_mod[["residuals"]] <- NA`
U `summ_mod[[residuals]] <- NULL`
V `summ_mod[[residuals]] <- NA`
S `summ_mod$residuals <- NULL`
T `summ_mod$residuals <- NA`

Question 14 (exercise 4.5.9)

```
df <- data.frame(item = c(1,2), color = c("red", "blue"), amount = c(3,6))
df
```

```
  item color amount
1    1   red      3
2    2  blue      6
```

What code would order df so that the columns are alphabetically?

O `df[order(names(df))]`
P `df[order(names(df)),]`
O `df[,order(names(df))]`

Question 15 (exercise 4.5.9)

What code samples 2 random columns from df?

A `df[sample(names(df),2),]`
E `df[sample(names(df),2)]`
I `df[sample(names(df,2))]`

Question 16

```
df <- data.frame(item = c(1,2, 3),
                 color = c("red", "green", "green"),
                 amount = c(3,6,2))
df
```

```
  item color amount
1    1   red      3
2    2 green      6
3    3 green      2
```

What code would select the rows with green items with a minimal amount of 4?

M `df[df$color = "green" & df$amount > 4]`
N `df[df$color = "green" & df$amount > 4,]`
O `df[df$color != "red" & df$color != "blue" & df$amount > 4]`
P `df[df$color != "red" & df$color != "blue" & df$amount > 4,]`

Question 17

```
setdiff(setdiff(LETTERS[1:13],LETTERS[1:8]), LETTERS[10:13])
```

Question 18

```
x <- 1:10
y <- x %% 5 == 0
z <- x %% 2 != 0
intersect( intersect(LETTERS[x], LETTERS[y]), LETTERS[z])
```


I feel amazing! It's all done.