# Lists

# Agenda and reminders

Exam 1 next Monday (October 6)

- Data wrangling fundamentals (`select`, `filter`, `mutate`, `summarize`, `group_by`, etc.)
- Reshaping data (pivoting), joins (left join, inner join)
- Wrangling across columns (`across`, `starts_with`, `where`, etc.)
- Iteration (`for` loops, `while` loops, `map`)
- Functions and simulations

# Agenda and reminders

- Exam 1 next Monday (October 6)
  - You will read and write short pieces of code
  - I expect you to know what kind of things are possible in R (key ideas like joining, reshaping data, summarizing, grouping, iterating, etc.)
  - I expect you to be familiar with key functions in R
  - Minor syntax errors will not be penalized
  - Example review questions on course website
  - Also look back at class activities and examples

# Agenda and reminders

- Exam 1 next Monday (October 6)
  - Review day this Friday (October 3)
- Today: lists
- Wednesday: more on functions (function defaults, function scoping)
- After exam 1:
  - Functions and unit tests
  - Starting text wrangling

# Previously: `purrr::map`

```
grade_files <- list.files("intro_stats_grades", full.names=T)
grade_tables <- map(grade_files, read_csv)
```

`map`: apply a function to each element of a list or vector

Output: a list

```
typeof(grade_tables)
```

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

```
length(grade_tables)
```

```
[1] 10
```

double bracket for indexing lists

```
glimpse(grade_tables[[1]])
```

```
Rows: 35
Columns: 14
$ student_id <dbl> 55817, 32099, 40295, 54195, 15297, 81786, 49747,
78226, 102…
$ hw_1       <dbl> 10, 10, 10, 10, 10, 7, 10, 10, 9, 9, 8, 10, 10, 7,
8, 8, 10…
```

```
$ hw_2        <dbl> 10, 9, 10, 9, 8, 8, 9, 9, 9, 8, 10, 10, 10, 6, 9,
10, 8, 10…
$ hw_3        <dbl> 9, 10, 9, 9, 9, 6, 8, 9, 10, 10, 8, 9, 9, 9, 10, 9,
10, 8, …
$ hw_4        <dbl> 9, 9, 9, 6, 10, 6, 8, 10, 7, 9, 9, 10, 10, 9, 9, 8,
9, 10, …
```

# Vectors revisited

Vectors can contain numbers, booleans, characters, etc:

```
1 x <- c(0, 1, 2)
2 x
```

```
[1] 0 1 2
```

```
1 typeof(x)
```

```
[1] "double"
```

(numeric vector)

```
1 x <- c("a", "b", "c")
2 x
```

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

```
1 typeof(x)
```

```
[1] "character"
```

(character vector)

The `typeof` function tells what *type* of object we have

# Vectors of multiple types?

```
1 x <- c(0, 1, "a")
2 x
3 x[1] + 1
```

What do you think will happen when we run this code?

# Vectors of multiple types?

```
1 x <- c(0, 1, "a")
2 x
```

```
[1] "0" "1" "a"
```

← forces everything to be a character

```
1 x[1] + 1
```

```
Error in x[1] + 1: non-numeric argument to binary operator
```

Basic vectors (called *atomic* vectors) only contain one type.

I can't do arithmetic with characters

# Lists

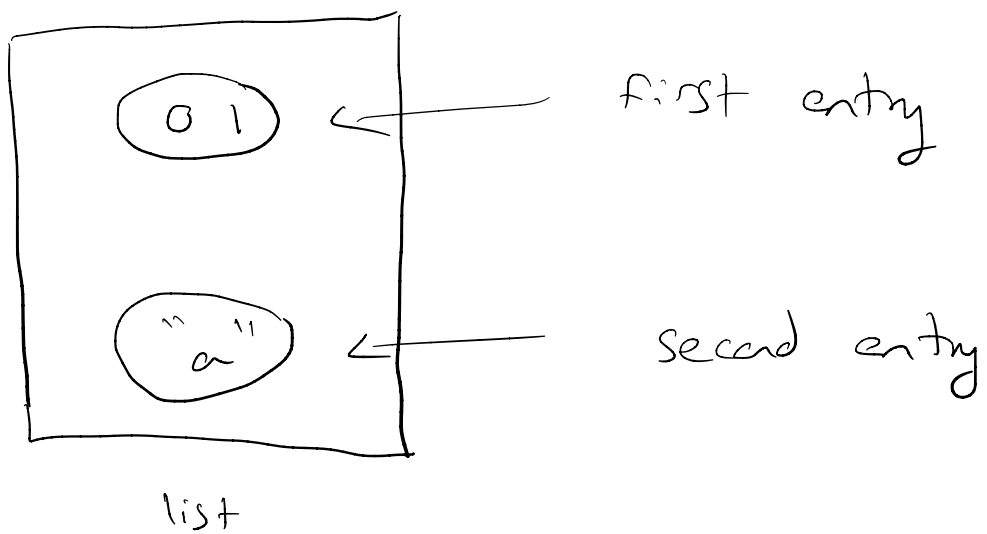
```
1 x <- list(c(0, 1), "a")
2 x
```

```
[[1]]
[1] 0 1

[[2]]
[1] "a"
```

first entry (vector (0, 1))

second entry (character "a")



# Lists

```
1 x <- list(c(0, 1), "a")
2 x
```

```
[[1]]
[1] 0 1

[[2]]
[1] "a"
```

x[[1]]

first entry in the list
(in this case, x[[1]]
is a vector)

```
1 x[[1]]
```

```
[1] 0 1
```

```
1 x[[1]][1]
```

```
[1] 0
```

x[[1]] [1]

vector ↑ first entry in that vector

# Lists

```r
x <- list(c(0, 1), "a")
x
```

```
[[1]]
[1] 0 1

[[2]]
[1] "a"
```

```r
x[[1]]
```

```
[1] 0 1
```

```r
x[[1]][1]
```

```
[1] 0
```

```r
typeof(x[[1]])
```

```
[1] "double"
```

```r
x[[2]]
```

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

```r
typeof(x[[2]])
```

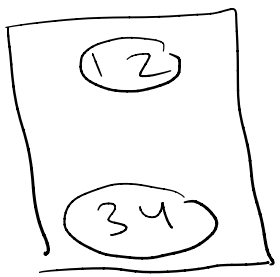
"character"

# Visualizing list structure

```
1 x1 <- list(c(1, 2), c(3, 4))
2 x1
```

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

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



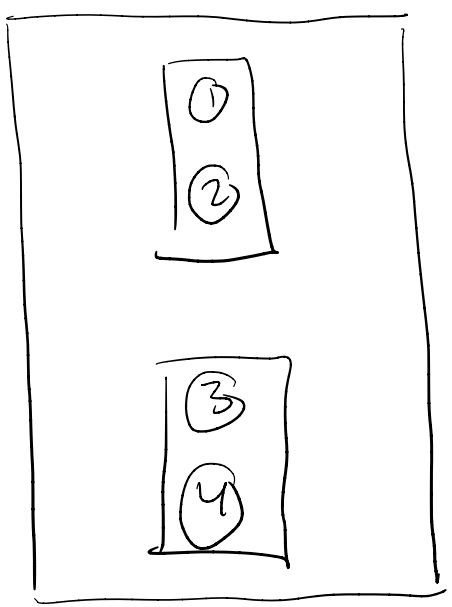

```
1 x2 <- list(list(1, 2), list(3, 4))
2 x2
```

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

[[1]][[2]]
[1] 2



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



```
[1] 3

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

# Indexing lists

```
1 x <- list(c(1, 2), c(3, 4))
2
3 x[1]
```

<- a list containing the first entry of x (list(c(1,2)))

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

```
1 typeof(x[1])
```

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

```
1 x[[1]]
```

<- the first entry of x

```
[1] 1 2
```

```
1 typeof(x[[1]])
```

```
[1] "double"
```

- `x[1]` returns a *list* which contains the first component of x
- `x[[1]]` returns the object stored in the first component

# Indexing lists

```
1 x <- list(list(1, 2), list(3, 4))
2 x[1]
```

x[[1]]

**Question:** What will `x[1]` return?

equiv. of list(list(1,2))

# Indexing lists

```r
x <- list(list(1, 2), list(3, 4))
x[1]
```

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

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

# Indexing lists

```
1 x <- list(list(1, 2), list(3, 4))
2 x[[1]]
```

**Question:** What will `x[[1]]` return?

# Indexing lists

```r
x <- list(list(1, 2), list(3, 4))
x[[1]]
```

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

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

**Question:** How do I get just the 3?

# Indexing lists

```
1 x <- list(list(1, 2), list(3, 4))
2 x[[2]][[1]]
```

```
[1] 3
```

# Class activity

https://sta279-f25.github.io/class_activities/ca_14.html