Data Structures in R

Dr. Rachael Tatman, Kaggle Data Scientist

R-Ladies Seattle
1/23/2019

@rctatman

What are data structures?

Ways of storing and organizing data.

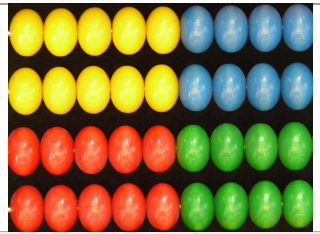
@rctatman

@rctatman

	Same data type	Different data types
1 dimension		
2 dimensions		
N dimensions		

*I straight up copied the margins of this chart from Hadley's "Advanced R" book, which is very good!

@rctatman

	Same data type	Different data types
1 dimension		
2 dimensions		
N dimensions		

@rctatman

	Same data type	Different data types
1 dimension	Atomic vector c("a","b","c")	
2 dimensions		
N dimensions		

	Same data type	Different data types
1 dimension	Atomic vector c("a","b","c")	
2 dimensions	Matrix	
N dimensions		

	Same data type	Different data types
1 dimension	Atomic vector c("a","b","c")	
2 dimensions	Matrix	
N dimensions	Array	

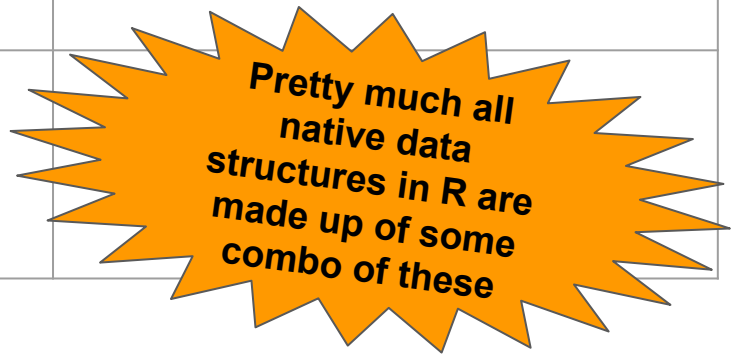
	Same data type	Different data types
1 dimension		
2 dimensions		
N dimensions		

	Same data type	Different data types
1 dimension		List
2 dimensions		
N dimensions		

	Same data type	Different data types
1 dimension		List
2 dimensions		Data frame
N dimensions		

	Same data type	Different data types
1 dimension	Atomic vector	List
2 dimensions	Matrix	Data frame
N dimensions	Array	

	Same data type	Different data types
1 dimension	Atomic vector	List
2 dimensions	Matrix	Data frame
N dimensions	Array	



Some important points

- Basically everything in R is some flavor of vector
  - Any single piece of data, like a single number or character, is a vector of length one
  - Each column in a dataframe is a vector (so can only have a single data type!)
- This means every piece of data in a data structure has a numeric "address" that lets you go right to it
  - vector[1]
  - list[[1]]
  - Dataframe[1,1]
- Having addresses make it very fast to find and perform operations on data, but slower to add and delete data from existing structures
  - Remember to pre-allocate your data structures!

R is a bit of an odd langage...

- Doesn't have native **linked lists** (including **stacks** or **queues**)
- Doesn't have any native **hashed data structures**
- Doesn't use **pointers**
- Doesn't have any **scalars**
- Doesn't use **binary trees** as native data structures

R is a bit of an odd langage...

- Doesn't have native **linked lists** (including **stacks** or **queues**)
- Doesn't have any native **hashed data structures**
- Doesn't use **pointers**
- Doesn't have any **scalars**
- Doesn't use **binary trees** as native data structures

Unfortunately, a lot of coding interview questions are set up *explicitly* to have you show that you know how to use these things.

R is a bit of an odd langage...

- Doesn't have native **linked lists** (inc… …es)
- Doesn't have any native **hashed …**
- Doesn't use **pointers**
- Doesn't have any **scalars**
- Doesn't use **binary trees** as na…

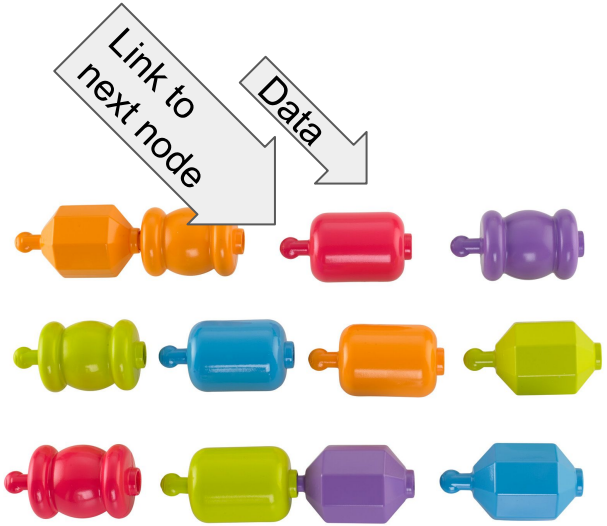
Unfortunately, a lot of coding interview questions are set up *explicitly* to have you show that you know how to use these things.

R is a bit of an odd langage...

- Doesn't have native **linked lists** (including **stacks** or **queues**)
- Doesn't have any native **hashed data structures**
- ~~Doesn't use **pointers**~~ (neither does Python or SQL though, so... ¯_(ツ)_/¯)
- ~~Doesn't have any **scalars**~~ (just, like, a single number or string on its own)
- ~~Doesn't use **binary trees** as native data structures~~ (basically just like decision trees, which most data folks know about already)

Unfortunately, a lot of coding interview questions are set up *explicitly* to have you show that you know how to use these things.

Linked List



- Made of up a series of nodes which contain:
  - A piece of data
  - A link to the next node in the list
  - (Sometimes a link to the previous node)
- Benefits?
  - It's very fast to add or remove data; you just need to update the links in the nodes next to the ones you've changed
  - Nodes don't need to be next to each other in memory (items in a vector do)
- Drawbacks?
  - It's very slow to get the n^{th} element in the list; you need to start from the beginning and go through the first n items in the list

Hashed data structures

(E.g. hash table, hash map, Python dictionaries)

- Each piece of data (value) is associated with a specific address (hash or key)
- Benefits:
  - It takes the same amount of time to get a piece of data, no matter how many pieces are stored (also true of R's vectors!)
  - It can be very fast to add or remove data
- Drawbacks:
  - There's no innate order to hash tables
  - You need to make sure the addresses are unique or can handle multiple things at the same address
  - If you don't save a list of addresses, you may not know how many items are in your table
  - Vectorization is right out

To review…

R's Data Structures

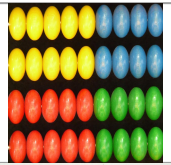
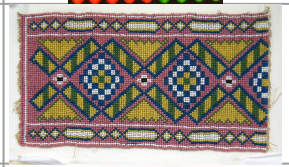
	Same data type	Different data types
1D	Atomic vector	List
2D	Matrix	Data frame
N D	Array	

Data Structures NOT in R

- Linked lists
- Hashtables

To review…

R's Data Structures

	Same data type	Different data types
1D		
2D		
N D		

Data Structures NOT in R

- Linked lists

- Hashtables

Thanks!
Questions?

@rctatman