DCD
Data Computing Diagram
We’ll be working with “Data”

How do statisticians / analysts think of data?

How do computers treat data?

How do data sets get stored?

How do programs “understand” data?
Be the boss of your data
How do programming languages handle data?
Data Sets

Computers

Software & Languages

Analyst / Scientist

Code, Scripts, Programs

OS

R

Python

C++

Code, Scripts, Programs
Data for Software & Languages?

Data Types

- Basic kinds

Data Structures

- Containers
Data Types (for programming languages)

Also refer to as data primitives or primitive types

They serve as the building blocks (i.e. they are like the atoms)
Common Data Types (for programming languages)

- Integers (i.e. whole numbers)
- Real numbers (i.e. decimal numbers)
- Boolean (i.e. logical)
- Character (i.e. strings)
Common Data Types (for programming languages)

In many programming languages, everytime you create an object or a variable, you must declare its type:

```c
char first_name
```
```c
int age
```

*(you don’t have to do this in R)*
Data Types in R
Data types in R

- **Logical** (boolean)
- **Integer** (whole numbers)
- **Double** (real, decimal numbers)
- **Character** (or strings)
  - *Complex (rarely used)*
  - *Raw (rarely used)*
Data Types (primitives)

TRUE     # logical
1L       # integer
2.5      # double (real)
"hello"  # character
1 + 3i    # complex
Vectors in R
To a large extent, **R** is a *vector*-based language
R vectors

A vector is the most basic data structure in R.

Vectors are contiguous cells containing data.
R vectors

Can be of any length (including zero)
Different kinds of vectors

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>numeric</td>
</tr>
<tr>
<td>TRUE</td>
<td>FALSE</td>
<td>TRUE</td>
<td>FALSE</td>
<td>logical</td>
<td></td>
</tr>
<tr>
<td>&quot;I&quot;</td>
<td>&quot;you&quot;</td>
<td>&quot;we&quot;</td>
<td>&quot;they&quot;</td>
<td>character</td>
<td></td>
</tr>
</tbody>
</table>
Common *(and not so common*) data types in R

A **logical** vector stores TRUE and FALSE values

An **integer** vector stores integers

A **double** vector stores regular (real) numbers

A **character** vector stores character strings

*A **complex** vector stores complex numbers*

*A **raw** vector stores raw bytes*
“Scalars” = one element vectors

z <- TRUE    # logical
x <- 1L      # integer
y <- 2.5     # real
w <- "hello" # character
u <- 1 + 3i  # complex
R parlance: Types and Modes

The function `typeof()` returns the type of data: this is how the values are stored internally in R.

In S terminology, instead of talking about types we talk about modes.

The function `mode()` returns the “mode” of an R object.
### Data types and modes

<table>
<thead>
<tr>
<th>value</th>
<th>example</th>
<th>mode</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>integer</td>
<td>1L, 2L</td>
<td>numeric</td>
<td>integer</td>
</tr>
<tr>
<td>real</td>
<td>1, -0.5</td>
<td>numeric</td>
<td>double</td>
</tr>
<tr>
<td>complex</td>
<td>3 + 5i</td>
<td>complex</td>
<td>complex</td>
</tr>
<tr>
<td>logical</td>
<td>TRUE, FALSE</td>
<td>logical</td>
<td>logical</td>
</tr>
<tr>
<td>character</td>
<td>“hello”</td>
<td>character</td>
<td>character</td>
</tr>
</tbody>
</table>

*useRs typically talk about the mode*

A bit confusing at the beginning
Special Values
There are some special data values in R

**NULL** = null object

**NA** = Not Available (missing value)

**Inf** = positive infinite

**-Inf** = negative infinite

**NaN** = Not a Number (different from NA)
Creating Vectors
Creating vectors

R provides a very large number of functions for creating all kinds of vectors.
Creating vectors with the combine function \texttt{c()}

\texttt{x <- c(1, 2, 3, 4, 5)}

\texttt{y <- c("one", "two", "three")}

\texttt{z <- c(TRUE, FALSE, TRUE)}
Sequences
Sequences

A common task involves creating sequences. The primary function is `seq()` but there’s also `seq_along()`, `seq_len()` and `seq.int()`
Numeric Sequences with colon operator:

1:5
1.5:5.5
5:1
-5:5
Numeric Sequences

```
seq(from = 1, to = 10)
```

```
seq(from = 1, to = 10, by = 2)
```

```
seq(from = 10, to = 1, by = -2)
```
Numeric Sequences

`seq(from=1, to=100, length.out=10)`

`seq_along(c(2,4,6,8))`

`seq.int(from = 2, to = 10, by = 2)`