# Implicit casting and data loss

Coercion, or the triggering of implicit data casting when two vectors or values are combined into a single vector using c() or other similar operations, is important because if not careful it can cause a loss of data or corrupt the original values of the data:

```
as.numeric( c( “900”, “1,000”, “36”) )  > 900 NA 36

c( x=c(0,1,2,3), L=c(T,F,T) ) >> c( 0,1,2,3,1,0,1)
```

Key concepts:

- Five main data types in R (plus the missing value type)
- Data “class” vs data “mode”
- Rules of coercion (hierarchy of implicit casting)

Additional contexts:

Coercion also occurs when stacking data frames, for example if X1 and X2 have different data types implicit casting would be triggered (or you would get an error).

```
df1 <- data.frame(X1,Y1)
df2 <- data.frame(X2,Y2)
rbind( df1, df2 )  # row bind
```

Example questions:

- If I have data that reports the highest grade level of schooling reported for adults, stored as a factor f, and I want to know the average grade completed in the sample why would this give me the wrong answer?

  ```
  mean( as.numeric( f ) )
  ```

- If X is a logical vector with TRUE/FALSE values, explain why the following mathematical operation works on what appears to be text (“TRUE” or “FALSE”)? How would one interpret the result?

  ```
  mean( x ) = 0.56
  ```

**Coercion: Rules for implicit casting operations triggered by c(X,Y) where X and Y have different types.**

The cases below are for reference only (you are not expected to know or memorize all types and all cases for the quiz). The purpose, rather, is to demonstrate the variety of types of implicit casting that can occur and recognize the basic rules of thumb for determining the type to keep. Each case will default to the data type that retains the most information and is least likely to corrupt the data.

For example, when combining logical vectors with numeric vectors the TRUE/FALSE values get converted to numeric values because you can represent T/F as numbers (0/1), whereas you can't represent all numbers as T/F values. Thus, you are not throwing away any information through the conversion.

```
c( x=c(0,1,2,3), y=c(T,F,T) ) >>   c( 0,1,2,3,1,0,1 )
```

Preserving information is not the same as preserving the integrity of the data, however. For example, it is not a problem if a factor gets recast as a character vector because the information is identical except factors take less space in memory. We can convert a factor to a character vector, then back to a factor and nothing has been changed, added, or lost. In the example above, however, we cannot convert the new numeric vector back to a logical vector because it is not clear that 0=FALSE and 1=TRUE in the numeric vector c(0,1,2,3), and it's not possible to differentiate which values of zero and one were originally logical values.

Note the behavior if we try to recast the new vector as logical:

```
as.logical( c(0,1,2,3,4) )
[1] FALSE  TRUE  TRUE  TRUE  TRUE
```

Any value of 1 or above gets converted to TRUE, demonstrating that moving back and forth between these data types is not nearly as easy as moving between character vectors and factors.

Coercion is an important feature of R (and other data programming languages) because without it programs would break more frequently and conducting basic analysis would be a lot more challenging. But the convenience does not come without a cost because not all conversions are smooth. The creators of R decided upon the rules you see below as the least bad options.

There is a clear hierarchy in the rules – when text is combined with anything else the new vectors always default to character. After that numeric is the next best option because

logical vectors, dates, and factors can all be represented as numbers but not vice versa. But in many cases changing the representation of the information is a process that has a high likelihood of distorting or corrupting the data.

Pay attention to when data types change in your analysis, try to identify the specific steps where coercion occurs, and always ask yourself if anything is being lost or changes as a result of the casting.

## Basic Data Types

- Six basic types in R:
    - Numbers (numeric)
    - Text (character)
    - Groups (factor)
    - T/F (logical)
    - Dates & Time (Date)
    - Missing values (NA), empty values (NULL)
- Checking data types:
    - Difference between class(x) and mode(x)
    - Mode tells you how data is stored by the computer
        - For example, factors and integers are both stored as numbers
        - The mode is determined by which representation is most “efficient”, i.e. will require the least amount of memory to store
    - Class tells you how the object will behave
        - R has default operations for each class (a summary of a categorical variable would look like a table, whereas a summary of a number would consist of summary statistics)
        - Class is the way R knows what to do with objects like vectors -

### Taxonomy of R Vector Types

**1. Numbers**

- **Numeric (double precision)**
    - class: "numeric"
    - mode: "numeric"
- **Integer**
    - class: "integer"
    - mode: "numeric"

- **Complex**
    - class: "complex"
    - mode: "complex"

---

**2. Logical (booleans)**

- **Logical**
    - class: "logical"
    - mode: "logical"

---

**3. Strings**

- **Character**
    - class: "character"
    - mode: "character"

---

**4. Group / Categorical variables**

- **Factor (unordered categories)**
    - class: "factor"
    - mode: "numeric" (internally stored as integers with labels)
- **Ordered Factor (ordinal categories)**
    - class: "ordered", "factor"
    - mode: "numeric"

---

**5. Dates and Times**

- **Date (calendar date)**
    - class: "Date"
    - mode: "numeric" (days since 1970-01-01)
- **POSIXct (date-time, numeric form)**
    - class: "POSIXct", "POSIXt"
    - mode: "numeric" (seconds since epoch)

---

**6. Special**

- **NULL** (represents empty placeholder or “no value”)
    - class: "NULL"
    - mode: "NULL"
- **NA** (missing value, is type-specific)
    - There is a different NA for each data type (logical, integer, numeric, character)
    - class: inherits from main vector type (logical, integer, numeric, character)

- mode: matches context as well

**Object-Oriented Programming:** Helpful Background on Why Types Matter

- Each object has a "type" called its "class"
  - For example, data frames, numbers, and dates
- You can check an object's class using the class() function:
  - class(v): numeric
  - class(f): factor
  - class(df): data.frame
- You can list all active objects using ls()
- In the object-oriented framework classes are used to make programming more efficient by allowing you to use one function to do lots of things; the behavior of R is determined by the class of the objects
  - plot(x): density plot
  - plot(x,y): scatter plot
  - plot( table(f) ): barplot
  - plot( f, x ): box and whisker plot
- Functions are called "methods" in object-oriented programming. A function that has different behaviors for different types of objects is called a method with a superclass (the default behavior), and subclasses (alternative behavior that overrides the default if the object type matches the subclass).

## The Hierarchy in Implicit Casting Rules

**Legend**

- **N** = numeric (integer/double)
- **L** = logical
- **T** = text/character
- **F** = factor
- **D** = date/time (covers Date and POSIXct)
- **NA** = missing (type depends on context; NA_real_, NA_character_, etc.)

**All 15 Cases of Implicit Casting:** `c(type1,type2) → resulting type`

*Note order would not matter: c(N,L) = c(L,N)*

1. **c(N, L) → numeric**
   Logical promoted to numeric (FALSE→0, TRUE→1).
   Reasoning: Least loss; numerics kept exact.

2. **c(N, T) → character**
   Numbers converted to strings.
   Reasoning: Safe to print numbers as text; reverse isn't safe.

3. **c(N, F) → numeric**
   Factors drop class, leaving their integer codes.
   Reasoning: Keeps raw numbers; labels can't survive without a factor-only vector.

4. **c(N, D) → numeric**
   Dates are stored as numbers internally (days/seconds since epoch).
   Reasoning: Class drops because not all elements are Date.

5. **c(N, NA) → numeric**
   NA takes on numeric type (NA_real_).
   Reasoning: Preserves numeric info.

6. **c(L, T) → character**
   Logical TRUE/FALSE become "TRUE"/"FALSE".
   Reasoning: Avoids losing the string.

7. **c(L, F) → numeric**
   Factor becomes integer, logical becomes 0/1.
   Reasoning: Both reduce to numbers.

8. **c(L, D) → numeric**
   Dates drop to numbers; logical coerced to 0/1.
   Reasoning: Lowest common denominator is numeric.

9. **c(L, NA) → logical**
   Missing takes on logical form.
   Reasoning: Preserves logical semantics.

10. **c(T, F) → character**
    Factors converted via as.character(levels[x]).
    Reasoning: Labels preserved (better than dropping to integers).

11. **c(T, D) → character**
    Dates converted to readable strings.
    Reasoning: Keeps both dates and arbitrary text.
12. **c(T, NA) → character**
    NA becomes NA_character_.
    Reasoning: Avoids losing the text.
13. **c(F, D) → numeric**
    Factor's integers + date's underlying numeric.
    Reasoning: No coherent way to preserve both classes.
14. **c(F, NA) → numeric**
    Factor class dropped; NA coerced to NA_integer_ → then vector is numeric.
    Reasoning: Only path to keep values together.
15. **c(D, NA) → numeric**
    Dates drop to underlying numeric with NA_real_.
    Reasoning: Class dropped since not all elements are date/time.

---

**Rules of Thumb and Edge Cases:**

- **Character always wins**: any text present → whole vector is character.
- **"Least loss" rules of thumb:**
    - If **any T** (text) is present → **character** (you can always print numbers/dates as text; the reverse isn't safe).
    - Since the mode of logicals, factors, and dates is numeric then when combining with numbers it defaults to numeric
- **Homogeneous classes are preserved**:
    - c(Date, Date, ...) → class "Date" stays.
    - However, if c(factor, factor) → factor stays **only if** levels are **identical**; otherwise R typically **drops to integer** (or may combine via as.character() if character appears anywhere).
- **NULL is special**: it *vanishes* in c() because it's no longer empty.
- **NA is a chameleon**: it takes the type of what it's combined with.

- **Some classes are fragile**: factor, Date, POSIXct only survive if the vector is *homogeneous* (factors have same levels, dates have same format) otherwise they might both be converted to characters.