W2: Working with data structures

https://bit.ly/ir-w2

W1: Clearest Points / Positives

> The responses to all the questions is fantastic. I really feel like I can ask anything which is so helpful!

> Having someone in the chat helping those who are remote and the instructor is comforting

> In general, coding is new to me (I work with data entry, so I do know the relevance), but I learned more details of how it a code is a language to the computer and will output information/answers.

> Understanding how R/R Studio/Posit imports data, and runs from the console.

Be patient. It takes some time getting used to R coming from elsewhere.

https://bit.ly/ir-w2

Clearest Points

> That in R you can use the equal sign to assign variables instead of the arrow

Yup, this ultimately is a matter of style. If you're coming to R from another language, I encourage you to continue to use the `=` sign. Just know that `<-` is pretty common among R users.

https://bit.ly/ir-w2

Muddy Points

> I feel like I don’t know what is unclear just yet as I am still very new to this concept.

> I’m a very beginner, most stuff was unclear so I need time to practice.

> My brain wants to know what every line of code is doing, otherwise I get hung up on trying to figure it out and can’t process what’s happening next. R feels like a very unintuitive language to me after using SAS for many years.

> Still not super clear how Quarto works.

Great point, it’s hard to know what you need to know when you’ve encountered something brand new. Be patient and curious, and please reach out when you need help.

https://bit.ly/ir-w2

Muddy Points

> The layout of Posit Cloud was a little confusing. The more we used it, the more I understood it, but it would have helped to have a little bit more of a tour of what all the windows are from the beginning.

If you haven't already, I have recorded a video here that is a guided tour of the RStudio interface:
https://www.youtube.com/watch?v=oFmjHxl28H0

> When to type into the console (bottom left) vs the source .qmd files (top left)

In general, you want to save your work, so I usually write my code in code chunks in a `.qmd` (Quarto Markdown file).

https://bit.ly/ir-w2

Ask Questions on Slido: https://bit.ly/ir-w2

https://bit.ly/ir-w2

Review of Exercise 1

https://bit.ly/ir-w2

Last week: Data types

- **Numeric**: 18, -21, 65, 1.25
- **Character**: “ATCG”, “Whatever”, “948-293-0000”
- **Logical**: TRUE, FALSE

https://bit.ly/ir-w2

Learning Objectives for Today

By the end of today, you should be able to:

- **Explain** how to construct vectors
- **Utilize** *explicit* and *implicit* subsetting on vectors
- **Define** what a data frame is and how it is related to vectors
- **Apply** *implicit* subsetting to data.frames

https://bit.ly/ir-w2

Data structures

Data structures stores information about data types.
Vector is a ordered collection of a data type. Each *element* of a vector contains a data type, and all elements of a vector must be the same type, such as numeric, character, or logical.
So, you can have **numeric vector**, **character vector**, and **logical vector**.

https://bit.ly/ir-w2

Defining a vector

We use `c()` - the *combine* function to declare a vector

```
staff = c("chris", "ted", "jeff")
chrNum = c(2, 3, 1)
```

What happens when you mix data types in a vector? One of the **data types is *coerced* to the other data type**.

```
new_vector = c("chris", 1, 2)
new_vector
```

```
[1] "chris" "1"     "2"
```

https://bit.ly/ir-w2

Using operations on vectors

You can use operations and functions on vectors.

```
chrNum = c(2, 3, 1)
chrNum3 = chrNum * 3
chrNum3
```

```
[1] 6 9 3
```

Multiplication has a new meaning when used on a numeric vector and numeric data type! This is called **operator overloading**.
A lot of operations/functions in R are *vectorized* - you can apply a function to a single value or a whole vector. This will become very important later on.
😮

https://bit.ly/ir-w2

Operator overloading

How about numeric vector multiplied by numeric vector?

```
1 chrNum * c(2, 2, 0)
```

```
[1] 4 6 0
```

a numeric vector added to a character vector:

```
1 #chrNum + staff
```

https://bit.ly/ir-w2

Functions on vectors

How many elements are in this vector?

```
1 length(staff)
```

```
[1] 3
```

https://bit.ly/ir-w2

Subsetting vectors

We subset vectors using the bracket [ ] operation.
Inside the bracket is either a single numeric value or an a **numerical indexing vector** containing numerical values. They dictate which elements of the vector to return.

```
1 staff[2]
```

```
[1] "ted"
```

```
1 staff[c(1, 2)]
```

```
[1] "chris" "ted"
```

```
1 small_staff = staff[c(1, 2)]
2 small_staff
```

```
[1] "chris" "ted"
```

https://bit.ly/ir-w2

Subsetting vectors

Alternatively, instead of using numerical indexing vectors, we can use a **logical indexing vector**. The logical indexing vector must be the *same length* as the vector to be subsetted, with TRUE indicating an element to keep, and FALSE indicating an element to drop.

```
staff[c(TRUE, FALSE, FALSE)]
```

```
[1] "chris"
```

```
staff[c(TRUE, TRUE, FALSE)]
```

```
[1] "chris" "ted"
```

```
small_staff = staff[c(TRUE, TRUE, FALSE)]
small_staff
```

```
[1] "chris" "ted"
```

https://bit.ly/ir-w2

Explicit subsetting

- **Explicit subsetting**: Given a length 10 vector of people's ages, and subset to the 1st, 3rd, and 9th elements.

```
age = c(89, 70, 64, 90, 66, 71, 55, 60, 30, 16)
```

```
age[c(1, 5, 9)]
```

```
[1] 89 66 30
```

```
age[c(TRUE, FALSE, FALSE, FALSE, TRUE, FALSE, FALSE, FALSE, TRUE, FALSE)]
```

```
[1] 89 66 30
```

https://bit.ly/ir-w2

Quick note about sequences

You can also declare a vector sequence using the `:` operator:

```
1 1:5
```

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

https://bit.ly/ir-w2

Implicit Subsetting

- **Implicit subsetting**: Given a length 10 vector of people's ages, and subset to elements greater than age 50.

  We don't know which elements to subset off the top of our head! If we know which elements are > 50, then we can give the elements for an explicit subset.
  Use a **comparison operator** to create a logical indexing vector:

```
1 age > 50
```

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

  Can you guess what we do next?

```
1 age[age > 50]
```

```
[1] 89 70 64 90 66 71 55 60
```

https://bit.ly/ir-w2

Implicit subsetting steps

Subset a vector **implicitly**, in 3 steps:

1. Come up with a criteria for subsetting: "I want to subset to values greater than 50".
2. Use a **comparison operator** to create a **logical indexing vector** that fits this criteria.

```
age > 50
```

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

3. Use this logical indexing vector to subset.

```
age[age > 50]
```

```
[1] 89 70 64 90 66 71 55 60
```

Alternatively,

```
idx = age > 50
age[idx]
```

```
[1] 89 70 64 90 66 71 55 60
```

https://bit.ly/ir-w2

Comparison operators:

`<` less than
`<=` less or equal than
`>` greater than
`>=` greater than or equal to
`==` equal to
`!=` not equal to
Subset `staff` to not have “chris” in it.

```
staff != "chris"
```

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

```
staff[staff != "chris"]
```

```
[1] "ted"  "jeff"
```

https://bit.ly/ir-w2

Ask Me Two Questions

https://bit.ly/ir-w2

https://bit.ly/ir-w2

`data.frame`s

A `data.frame` is a table such that

- each column is a vector
- each column has a data type
- order matters

This is the data structure that we will be working with for the rest of class.

```
1 load(url("https://github.com/fhdsl/Intro_to_R/raw/main/classroom_data/CCLE.RData"))
2 head(metadata)
```

https://bit.ly/ir-w2

`data.frame` Properties

Rows:

```
1 nrow(metadata)
```

```
[1] 1864
```

Columns:

```
1 ncol(metadata)
```

```
[1] 30
```

Dimension (rows x columns)

```
1 dim(metadata)
```

```
[1] 1864   30
```

https://bit.ly/ir-w2

Useful Tools for Peeking at `data.frame`s

```
head(metadata)
View(metadata)
```

https://bit.ly/ir-w2

Column names of `data.frame`s

The **column name** is a character vector that corresponds to the columns of the `data.frame`.

```
1 colnames(metadata)
```

```
 [1] "ModelID"               "PatientID"             "CellLineName"
 [4] "StrippedCellLineName"  "Age"                   "SourceType"
 [7] "SangerModelID"         "RRID"                  "DepmapModelType"
[10] "AgeCategory"           "GrowthPattern"         "LegacyMolecularSubtype"
[13] "PrimaryOrMetastasis"   "SampleCollectionSite"  "Sex"
[16] "SourceDetail"          "LegacySubSubtype"      "CatalogNumber"
[19] "CCLEName"              "COSMICID"              "PublicComments"
[22] "WTSIMasterCellID"      "EngineeredModel"       "TreatmentStatus"
[25] "OnboardedMedia"        "PlateCoating"          "OncotreeCode"
[28] "OncotreeSubtype"       "OncotreePrimaryDisease" "OncotreeLineage"
```

https://bit.ly/ir-w2

Subsetting a column from a `data.frame`

The `dataframe$<column_name>` operation selects for a column by its column name and returns the column as a vector:

```
1 metadata$OncotreeLineage
```

```
 [1] "Ovary/Fallopian Tube"      "Myeloid"
 [3] "Bowel"                     "Myeloid"
 [5] "Myeloid"                   "Myeloid"
 [7] "Bowel"                     "Skin"
 [9] "Bowel"                     "Bladder/Urinary Tract"
[11] "Lung"                      "Ovary/Fallopian Tube"
[13] "Skin"                      "Lung"
[15] "Kidney"                    "Breast"
[17] "Bladder/Urinary Tract"     "Breast"
[19] "Lymphoid"                  "Lung"
[21] "Pancreas"                  "Pancreas"
[23] "Lymphoid"                  "CNS/Brain"
[25] "Bladder/Urinary Tract"     "CNS/Brain"
[27] "Breast"                    "Lung"
[29] "Lung"                      "Pancreas"
[31] "Lymphoid"                  "Lung"
[33] "Myeloid"                   "Lung"
[35] "CNS/Brain"                 "Soft Tissue"
[37] "Lymphoid"                  "Bone"
[39] "CNS/Brain"                 "Bone"
[41] "Pancreas"                  "Fibroblast"
[43] "Breast"                    "Myeloid"
[45] "Kidney"                    "Esophagus/Stomach"
```

https://bit.ly/ir-w2

Subsetting a column from a `data.frame`

```
1 metadata$Age[1:5]
```

```
[1] 60 36 72 30 30
```

Challenge!

```
1 metadata$Age[metadata$OncotreeLineage == "Myeloid"]
```

```
 [1] 36 30 30 64 35 10 77 55 38 35 NA 66 29 63 65  1 64 31 29 71 33 24 58  7 23
[26]  5 47 29 44 73 57 20 77 76 59 35 37 45 40  3 68 46 53 13  0 22  0 62  7 36
[51] 32  0  0 64 64 41 20 63 35 56 37 22 53 33 NA NA  2 28 81 26 52 33 69 51 37
[76]  5 40
```

https://bit.ly/ir-w2

Subsetting columns and rows from a `data.frame`

`dataframe[row_idx, col_idx]` subsets the `data.frame` by a row indexing vector `row_idx`, and a column indexing vector `col_idx`.

```
1 metadata[1:5, c(1, 3)]
```

https://bit.ly/ir-w2

Subsetting `data.frame`s

We can refer to the column names directly:

```
1 metadata[1:5, c("ModelID", "CellLineName")]
```

https://bit.ly/ir-w2

Subsetting columns or rows from a `data.frame`

We can leave the column index or row index empty to just subset columns or rows.

```
1 metadata[1:5, ]
```

https://bit.ly/ir-w2

Subsetting Columns or Rows

```
1 metadata[, c("ModelID", "CellLineName")]
```

https://bit.ly/ir-w2

The crucial bit:

```
1 metadata[metadata$OncotreeLineage == "Myeloid",]
```

https://bit.ly/ir-w2

Ask me two questions on Slido

https://bit.ly/ir-w2

https://bit.ly/ir-w2

Week 2 Check-in

https://docs.google.com/forms/d/e/1FAIpQLScByNkcK4xMVosHkIxSXBTrzvSpUqUx01I-_ICvjs3NgZcePg/viewform?usp=sf_link

https://bit.ly/ir-w2