VIDO BIOINFO Bioinformatics and Systems Biology Lab

Canadian Bioinformatics Hub

Canadian Bioinformatics Workshops

Introduction to R Programming for Bioinformatics

Day 1- Module 2A: Getting your data into R

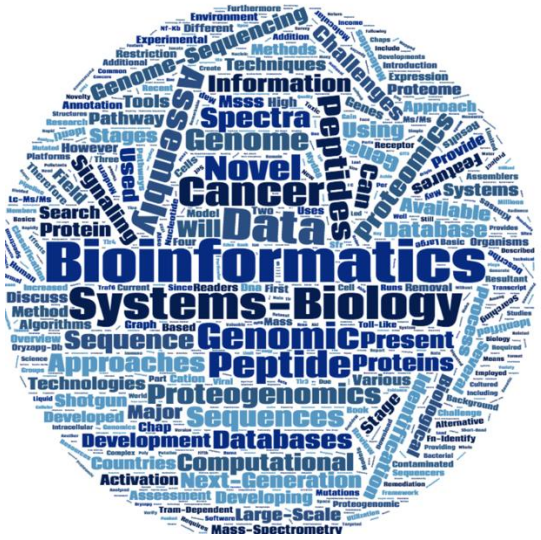
Mohamed Helmy, PhD
Principal Scientist and Adjunct Professor
Bioinformatics and Systems Biology Lab
VIDO, University of Saskatchewan

6-7 October 2025, VIDO, Saskatoon

Why Data Import Matters?

- Biomedical data almost never comes preloaded
- Must be imported before analysis
- Multiple file formats: CSV, TSV, Excel, TXT, FASTA, GFF3
- Goal
  - bring raw data into R as clean, analyzable objects
- Examples
  - patient metadata, omics data, clinical measurements, protein sequence, gene expression data

File Handling & Plotting in Bioinformatics



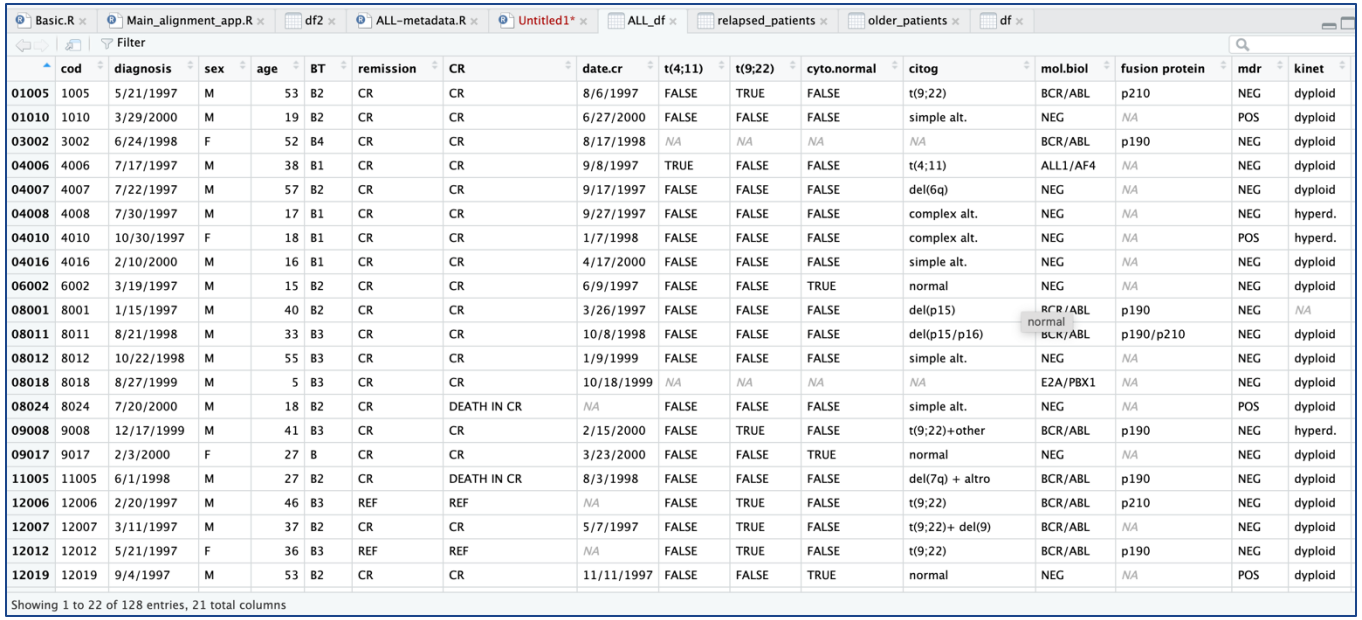
- Bioinformatics data = large, structured, diverse
- Reading & visualizing data = essential first step
- Applies to:
  - Sequence data (FASTA, FASTQ)
  - Variant data (VCF)
  - Clinical/metadata (CSV, Excel)
  - Public repositories (GEO, TCGA, UniProt)

Common File Formats

- **CSV (Comma-Separated Values)**
  - Widely used, easy to exchange
  - Example: PatientID, Age, Sex, Diagnosis

- **TSV (Tab-Separated Values)**
  - Common in genomics & transcriptomics
  - Excel (.xls / .xlsx)

- **Used in labs, supports multiple sheets**
  - TXT (plain text)
  - Flexible but may need parsing



Basic.R | Main_alignment_app.R | df2 | ALL-metadata.R | Untitled1* | ALL_df | relapsed_patients | older_patients | df

Filter

	cod	diagnosis	sex	age	BT	remission	CR	date.cr	t(4;11)	t(9;22)	cyto.normal	citog	mol.biol	fusion protein	mdr	kinet
01005	1005	5/21/1997	M	53	B2	CR	CR	8/6/1997	FALSE	TRUE	FALSE	t(9;22)	BCR/ABL	p210	NEG	dyploid
01010	1010	3/29/2000	M	19	B2	CR	CR	6/27/2000	FALSE	FALSE	FALSE	simple alt.	NEG	NA	POS	dyploid
03002	3002	6/24/1998	F	52	B4	CR	CR	8/17/1998	NA	NA	NA	NA	BCR/ABL	p190	NEG	dyploid
04006	4006	7/17/1997	M	38	B1	CR	CR	9/8/1997	TRUE	FALSE	FALSE	t(4;11)	ALL1/AF4	NA	NEG	dyploid
04007	4007	7/22/1997	M	57	B2	CR	CR	9/17/1997	FALSE	FALSE	FALSE	del(6q)	NEG	NA	NEG	dyploid
04008	4008	7/30/1997	M	17	B1	CR	CR	9/27/1997	FALSE	FALSE	FALSE	complex alt.	NEG	NA	NEG	hyperd.
04010	4010	10/30/1997	F	18	B1	CR	CR	1/7/1998	FALSE	FALSE	FALSE	complex alt.	NEG	NA	POS	hyperd.
04016	4016	2/10/2000	M	16	B1	CR	CR	4/17/2000	FALSE	FALSE	FALSE	simple alt.	NEG	NA	NEG	dyploid
06002	6002	3/19/1997	M	15	B2	CR	CR	6/9/1997	FALSE	FALSE	TRUE	normal	NEG	NA	NEG	dyploid
08001	8001	1/15/1997	M	40	B2	CR	CR	3/26/1997	FALSE	FALSE	FALSE	del(p15)	BCR/ABL	p190	NEG	NA
08011	8011	8/21/1998	M	33	B3	CR	CR	10/8/1998	FALSE	FALSE	FALSE	del(p15/p16)	BCR/ABL	p190/p210	NEG	dyploid
08012	8012	10/22/1998	M	55	B3	CR	CR	1/9/1999	FALSE	FALSE	FALSE	simple alt.	NEG	NA	NEG	dyploid
08018	8018	8/27/1999	M	5	B3	CR	CR	10/18/1999	NA	NA	NA	NA	E2A/PBX1	NA	NEG	dyploid
08024	8024	7/20/2000	M	18	B2	CR	DEATH IN CR	NA	FALSE	FALSE	FALSE	simple alt.	NEG	NA	POS	dyploid
09008	9008	12/17/1999	M	41	B3	CR	CR	2/15/2000	FALSE	TRUE	FALSE	t(9;22)+other	BCR/ABL	p190	NEG	hyperd.
09017	9017	2/3/2000	F	27	B	CR	CR	3/23/2000	FALSE	FALSE	TRUE	normal	NEG	NA	NEG	dyploid
11005	11005	6/1/1998	M	27	B2	CR	DEATH IN CR	8/3/1998	FALSE	FALSE	FALSE	del(7q) + altro	BCR/ABL	p190	NEG	dyploid
12006	12006	2/20/1997	M	46	B3	REF	REF	NA	FALSE	TRUE	FALSE	t(9;22)	BCR/ABL	p210	NEG	dyploid
12007	12007	3/11/1997	M	37	B2	CR	CR	5/7/1997	FALSE	TRUE	FALSE	t(9;22)+ del(9)	BCR/ABL	NA	NEG	dyploid
12012	12012	5/21/1997	F	36	B3	REF	REF	NA	FALSE	TRUE	FALSE	t(9;22)	BCR/ABL	p190	NEG	dyploid
12019	12019	9/4/1997	M	53	B2	CR	CR	11/11/1997	FALSE	FALSE	TRUE	normal	NEG	NA	POS	dyploid

Showing 1 to 22 of 128 entries, 21 total columns

Common File Formats in Bioinformatics

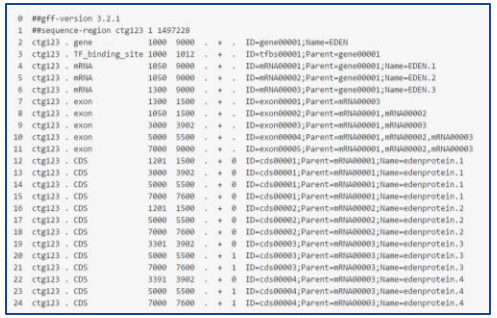
- **FASTA**
  - nucleotide/protein sequences
- **FASTQ**
  - sequence + quality scores
- **VCF**
  - genetic variants
- **GTF/GFF3**
  - gene annotations
- **SAM/BAM**
  - sequence alignments
- **CSV/TSV**
  - tabular clinical or metadata

Knowing the format = knowing the right tool

FASTA

```
>sp|P31946|1433B_HUMAN 14-3-3 protein beta/alpha OS=Homo sapiens
MTMDKSELVQKAKLAEQAERYDDMAAAMKAVTEQGHELSNEERNLLSVAYKNVVGARRSS
WRVISSIEQKTERNEKKQQMGKEYREKIEAELQDICNDVLELLDKYLIPNATQPESKVFY
LKMKGDYFRYLSEVASGDNKQTTVSNSQQAYQEAFEISKKEMQPTHPIRLGLALNFSVFY
YEILNSPEKACSLAKTAFDEAIAELDTLNEESYKDSTLIMQLLRDNLTLWTSENQGDEGD
AGEGEN
>sp|P62258|1433E_HUMAN 14-3-3 protein epsilon OS=Homo sapiens
MDDREDLVYQAKLAEQAERYDEMVESMKKVAGMDVELTVEERNLLSVAYKNVIGARRASW
RIISSIEQKEENKGGEDKLKMIREYRQMVETELKLICCDILDVLDKHLIPAANTGESKVF
YYKMKGDYHRYLAEFATGNDRKEAAENSLVAYKAASDIAMTELPPTHPIRLGLALNFSVF
YYEILNSPDRACRLAKAAFDDAIAELDTLSEESYKDSTLIMQLLRDNLTLWTSDMQGDGE
EQNKEALQDVEDENQ
>sp|Q04917|1433F_HUMAN 14-3-3 protein eta OS=Homo sapiens GN=YWHAH
MGDREQLLQRARLAEQAERYDDMASAMKAVTELNEPLSNEDRNLLSVAYKNVVGARRSSW
RVISSIEQKTMADGNEKKLEKVKAYREKIEKELETVCNDVLSLLDKFLIKNCNDFQYESK
VFYLKMKGDYYRYLAEVASGEKKNSVVEASEAAYKEAFEISKEQMQPTHPIRLGLALNFS
VFYYEIQNAPEQACLLAKQAFDDAIAELDTLNEDSYKDSTLIMQLLRDNLTLWTSDQQDE
EAGEGN
...
...
...
```

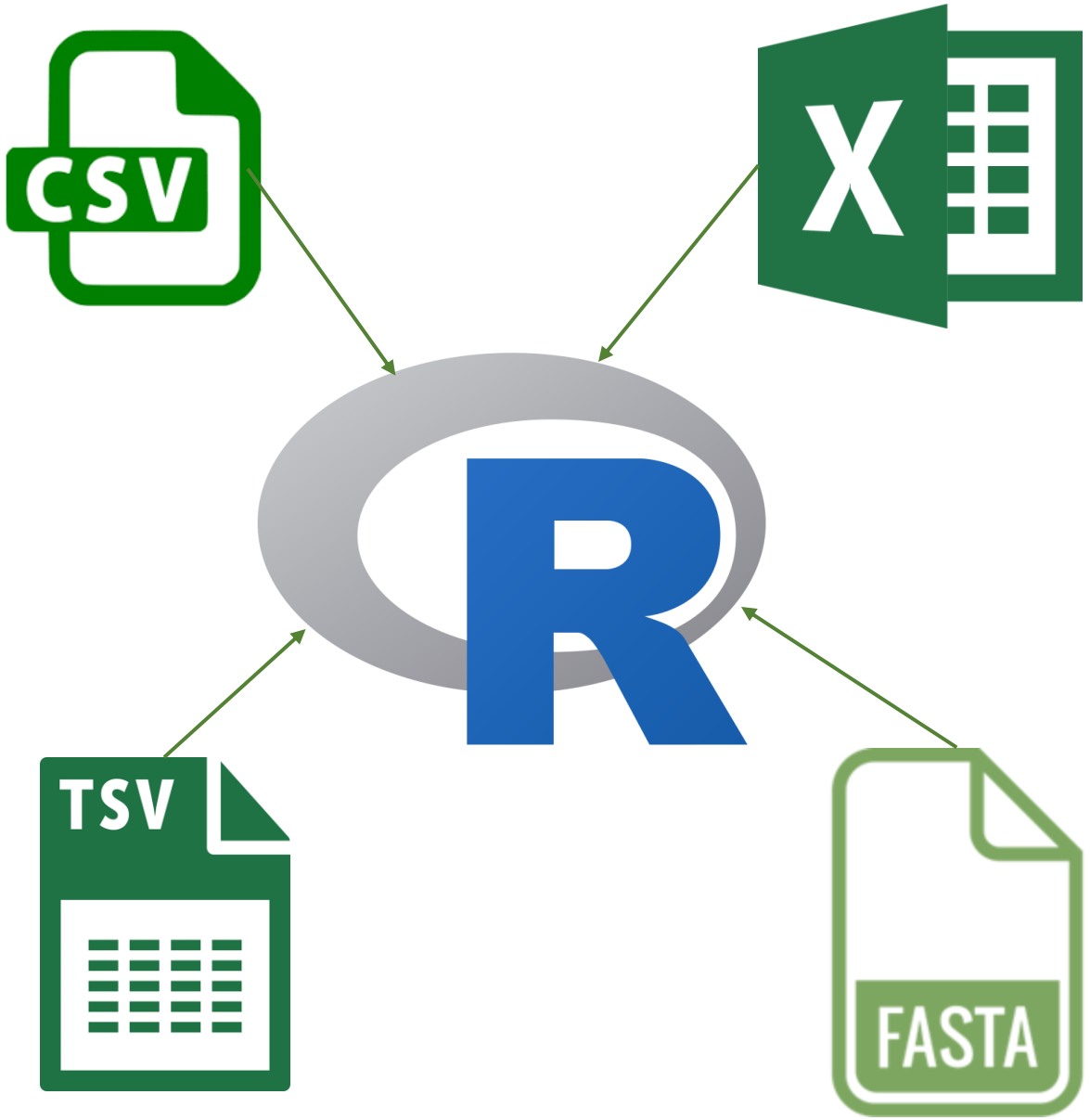
Identifier — @SRR566546.970 HWUSI-EAS1673_11067_FC7070M:4:1:2299:1109 length=50
Sequence — TTGCCTGCCTATCATTTTAGTGCCTGTGAGGTGGAGATGTGAGGATCAGT
'+' sign — +
Quality scores — hhhhhhhhhhghhghhhhhfhhhhhfffffe'ee['X]b[d[ed'[Y[^Y
Identifier — @SRR566546.971 HWUSI-EAS1673_11067_FC7070M:4:1:2374:1108 length=50
Sequence — GATTTGTATGAAAGTATACAACTAAAACTGCAGGTGGATCAGAGTAAGTC
'+' sign — +
Quality scores — hhhhgfhhcghghggfcffdhfehhhhcehdchhdhahehffffde'bVd



```
##fileformat=VCFv4.1
##fileDate=20140930
##source=23andme2vcf.pl https://github.com/arrogantrobot/23
##reference=file://23andme_v3_hg19_ref.txt.gz
##FORMAT=<ID=GT,Number=1,Type=String,Description="Genotype"
#CHROM  POS ID  REF ALT QUAL  FILTER  INFO  FORMAT  GEN
chr1  82154  rs4477212  a  .  .  .  .  GT  0/0
chr1  752566  rs3094315  g  A  .  .  .  GT  1/1
chr1  752721  rs3131972  A  G  .  .  .  GT  1/1
chr1  798959  rs11240777  g  .  .  .  .  GT  0/0
chr1  800007  rs6681049  T  C  .  .  .  GT  1/1
chr1  838555  rs4970383  c  .  .  .  .  GT  0/0
chr1  846808  rs4475691  C  .  .  .  .  GT  0/0
chr1  854250  rs7537756  A  .  .  .  .  GT  0/0
chr1  861808  rs13302982  A  G  .  .  .  GT  1/1
chr1  873558  rs1110052  G  T  .  .  .  GT  1/1
chr1  882033  rs2272756  G  A  .  .  .  GT  0/1
chr1  888659  rs3748597  T  C  .  .  .  GT  1/1
chr1  891945  rs13303106  A  G  .  .  .  GT  0/1
```

R Tools for Data I/O

- **Base R functions**
  - read.csv(), read.csv2() # read CSV files
  - read.table() # read table from file
  - readChar() # read charchters from a connection (i.e., URL)
- **readr:**
  - read_csv()
  - read_tsv()
- **readxl: read Excel files**
  - read_xls()
  - read_xlsx()
- **Bioconductor packages:**
  - Biostrings (FASTA)
  - VariantAnnotation (VCF)
  - rtracklayer (GTF/GFF)
- **data.table:**
  - fast reading for large data



Demo: reading files into R

- **CSV Example using base are functions:**
  - read.csv2()
- **CSV Example using *readr* functions :**
  - read_csv()

```r
# Read data in to R
# read CSV - base functions
bp <- read.csv2("Desktop/R/data/BloodPressure_Data.csv") # no sepqration
# take a quick look at the data
head(bp)

bp <- read.csv2("Desktop/R/data/BloodPressure_Data.csv", sep = ",") # no sepqration
# take another look at the data
head(bp)
str(bp)

# readr functions
library(readr)

# Read the CSV file
bp_data <- read_csv("Desktop/R/data/BloodPressure_Data.csv")

# Take a quick look at the data
head(bp_data)
str(bp_data)
```

Handling Dates and Columns

- **Why it's Important**
  - Dates track sample collection, diagnosis, or treatment over time.
  - Columns hold key variables (patient ID, age, diagnosis, etc.).
  - Correct handling ensures accurate, reproducible analysis.

- **Common Challenges**
  - Inconsistent date formats (1997-05-12, 12/05/1997, May 12, 1997).
  - Columns read as wrong data types (text vs. number).
  - Missing or inconsistent values.
  - Multiple variables in a single column (Age_Sex).
  - Large datasets need selective subsetting.

- **Takeaway**
  - Proper handling = accuracy, consistency, and meaningful insights.

Hand on: Working with dates and columns

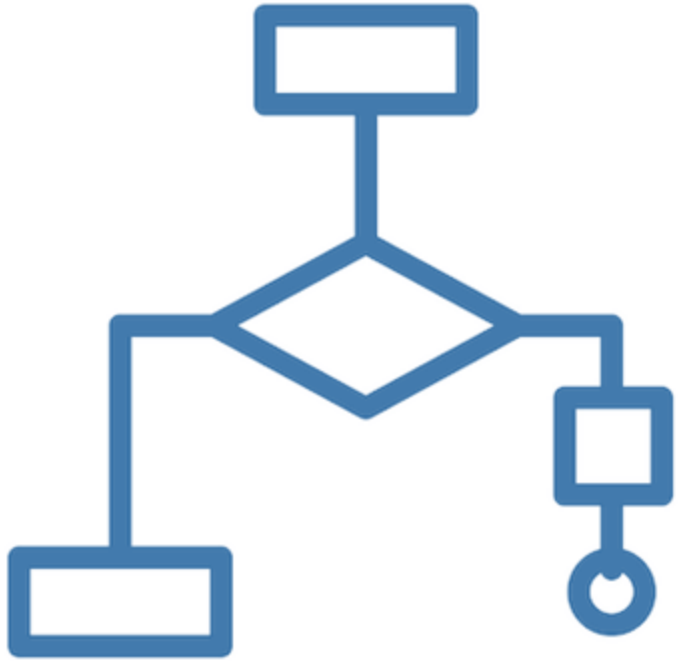
- **Use the blood pressure dataset**
  - Read the file into R
  - Make sure all the entries of the “Date” column are in the YMD format
  - Create a new column and store the year in this column
  - Filter the patients based on the year and sex

```r
# Work with date
library(readr)
library("lubridate")

# read ALL data
bp <-  read.csv2("Desktop/R/data/BloodPressure_wDates.csv", sep = ",")

# Convert date column and extract year
bp$Date <- ymd(bp$Date)
bp$Year <- year(bp$Date)

# Filtering blood pressure patients by year and gender
subset(bp, Year == 2003 & Gender == "f")
```

Conditions and Loops in R

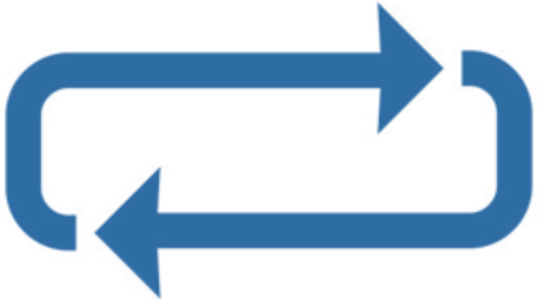
Why Do We Need Conditions & Loops?

- **Conditions = let R make decisions.**
- **Loops = let R repeat tasks automatically.**



- **Essential for bioinformatics workflows:**
  - Filter patient/sample data.
  - Apply the same operation to multiple files or genes.
  - Automate repetitive analysis steps.

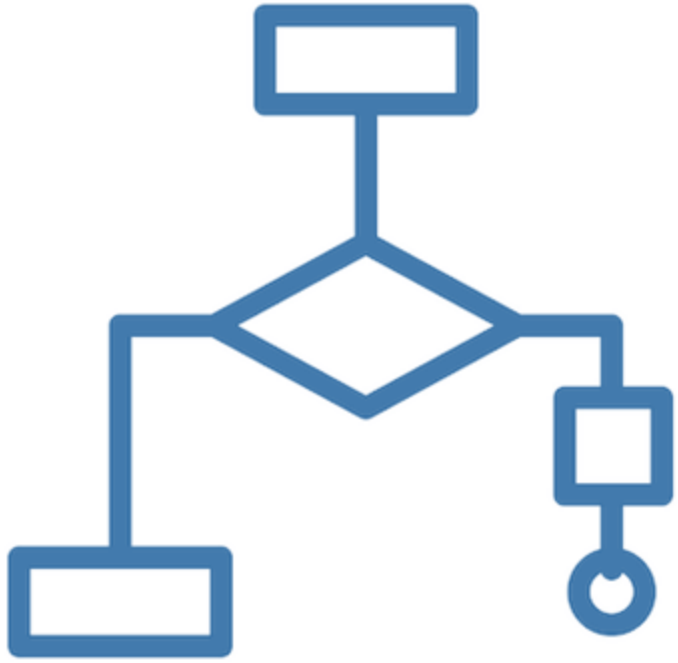
- **Example:**
  - “Find all patients diagnosed after 2002. Then update the Age by +1 year”
  - “Apply normalization to every sample in a dataset.”



If {} Else {}Statements

- **Syntax**

```
# If {} else {} statement
if (condition) {
  # code if TRUE
} else {
  # code if FALSE
}
```



- **Example**

```
# If else example
age <- 55
if (age > 50) {
  print("Older patient")
} else {
  print("Younger patient")
}
```

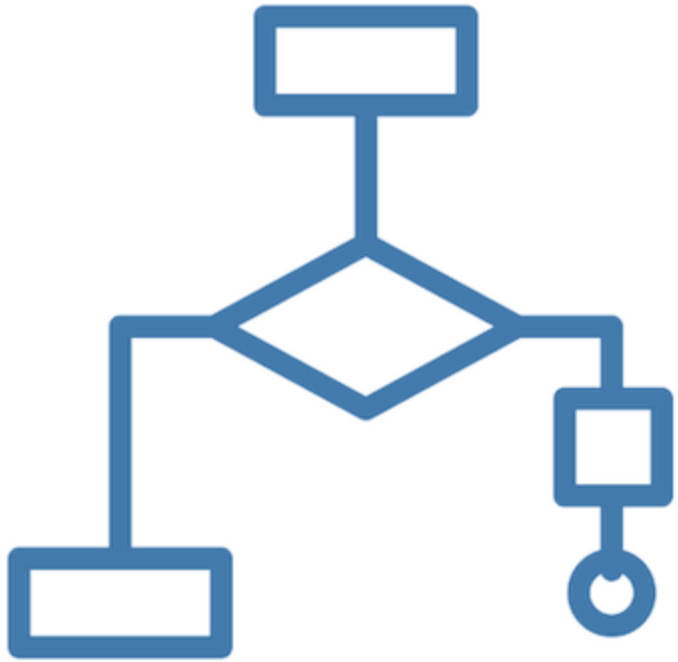
If {} Else {}Statements, multiple conditions

- **If {} Else If {} Else Statements**
- **Syntax**

```
# If {} else if {} else statement

if (condition1) {
  # code if condition1 is TRUE
} else if (condition2) {
  # code if condition2 is TRUE
} else {
  # code if none are TRUE
}
```

- **Example**

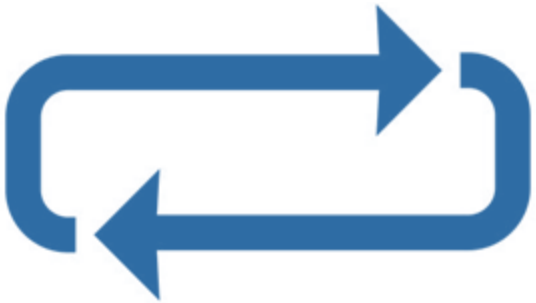
```
# If else if example
age <- 35

if (age < 18) {
  print("Child")
} else if (age >= 18 & age < 60) {
  print("Adult")
} else {
  print("Senior")
}
```



For loop

- **Syntax**

```r
# for loops
for (i in 1:5) {
  print(i)
}
```



- **Example**

```r
# for loop example
patients <- c("P1", "P2", "P3")
for (p in patients) {
  print(paste("Processing:", p))
}
```

Loops vs. Vectorization in R

- **for loops work, but they are slow for large datasets in R.**
- **R is optimized for vectorized operations.**
- **We use the apply() family of functions.**
- **Good practice:**
  - Use for loops for learning, testing your code and small tasks.
  - Use apply(), lapply(), sapply() for efficiency on real biological data.

```r
# the apply () family vs. loops
# Using a for loop
m <- matrix(1:9, nrow=3)
row_sums <- c()
for (i in 1:nrow(m)) {
  row_sums[i] <- sum(m[i, ])
}

# Using apply()
row_sums2 <- apply(m, 1, sum)
```

The Apply Family in R

- **apply(X, MARGIN, FUN)**
  - Apply a function to rows (1) or columns (2) of a matrix/dataframe.
- **lapply(X, FUN)**
  - Apply function to each element of a list; returns a list.
- **sapply(X, FUN)**
  - Same as lapply(), but tries to simplify output to a vector or matrix.
- **tapply(X, INDEX, FUN)**
  - Apply function to subsets of a vector, defined by a factor.
- **mapply(FUN, …)**
  - Multivariate version of sapply(). Applies a function in parallel to multiple vectors.

```r
# the apply() family
# apply()
apply(m, 1, sum)   # row sums
apply(m, 2, mean)  # column means

# lapply()
lapply(list(1:3, 4:6), mean)

# sapply()
sapply(list(1:3, 4:6), mean)

# lapply()
lapply(list(1:3, 4:6), mean)

# tapply()
ages <- c(21, 25, 30, 40, 35)
gender <- c("M", "M", "F", "F", "M")
tapply(ages, gender, mean)   # mean age by gender


# mapply()
nums1 <- 1:5
nums2 <- 6:10
mapply(sum, nums1, nums2)   # adds 1+6, 2+7, … 5+10
```

Hands-on: Loops & Conditions with

We will use the Blood Pressure data

- **Task 1 – Basic Filtering**
  - Use a for loop with if/else conditions
  - Go through each row of the dataset and:
  - Print a message if the patient has High BP (> 140) # Tip: use the paste() function
  - Print a message if the patient has Low BP (< 90)
  - Otherwise, mark them as Normal

- **Task 2 – Rewrite the same logic using apply()**
  - Instead of looping, create a new column (BP_Status) in the dataset:
  - Assign "HIGH", "LOW", or "NORMAL" to each patient.

```r
# read data
bp_data <- read.csv2("Desktop/R/data/BloodPressure_wDates.csv", sep = ",")

# Use for loop
for (i in 1:nrow(bp_data)) {
  if (bp_data$BloodPressure[i] > 140) {
    print(paste("Patient", bp_data$ID[i], "has HIGH blood pressure"))
  } else if (bp_data$BloodPressure[i] < 90) {
    print(paste("Patient", bp_data$ID[i], "has LOW blood pressure"))
  } else {
    print(paste("Patient", bp_data$ID[i], "is NORMAL"))
  }
}


# Use apply() instead of loops
bp_data$BP_Status <- apply(bp_data, 1, function(row) {
  if (as.numeric(row["BloodPressure"]) > 140) {
    "HIGH"
  } else if (as.numeric(row["BloodPressure"]) < 90) {
    "LOW"
  } else {
    "NORMAL"
  }
})
```

THANK YOU

VIDO

VACCINE AND INFECTIOUS DISEASE ORGANIZATION

VIDO.ORG

UNIVERSITY OF SASKATCHEWAN