R for bioinformatics

Lesson 1

January 09, 2022

Who I am

- My name is Dmitri. Dr. Dmitri Kazmin if you want to get formal.
- I speak both English and Russian
- I have a PhD in biochem
- I started as a wet-lab biologist and gradually transitioned to bioinformatics
- I have been a professional bioinformatician for ~15 years
- I do quite a bit of coding, but I don’t consider myself a programmer
- I am a biologist, who uses the tools of bioinformatics to seek answers to biological questions

Class structure

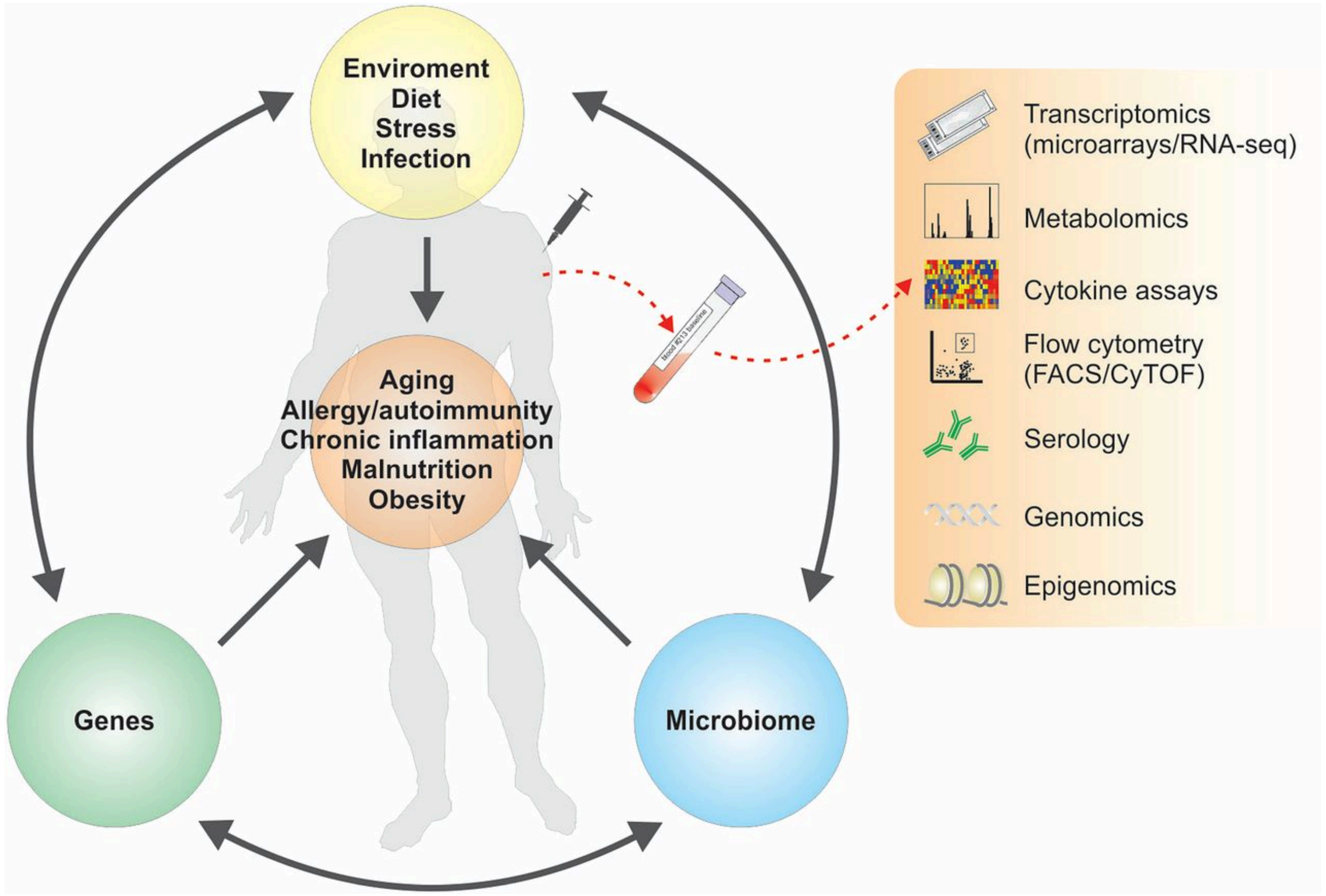
- Each class period will contain a (brief) presentation and lots of hands-on practice
- Source data, PDFs and codes for each class will be posted on Github (https://github.com/DmitriKazmin/R.class)
- Participants are encouraged to submit their homework by uploading to Github instead of using email
- The course will use actual data from a novel immunological clinical study
- We will explore and analyze the real-world data and will come up with biological hypotheses about the mechanisms of immunity to influenza vaccine

Course syllabus

Week	Topics
1	Introduction to bioinformatics; description of the clinical study; introduction to basic concepts in programming (variables, functions, arguments)
2	Data classes in R and variable types. Scalars, vectors, matrices, data frames and lists. Data manipulation and extraction.
3	More advanced R syntax: FOR loops and IF / ELSE statements. Data input and output.
4	Exploring built-in datasets; basic statistical analysis, parametric vs. non-parametric tests of significance.
5	Plotting in R.
6	Gene expression dataset: finding differentially regulated genes, and graphically representing the effects of vaccination on the blood transcriptome
7	Gene set enrichment analysis: making biological sense of the gene expression patterns
8	Antibody data: analyzing and representing the effects of vaccine on accumulation of antibodies against the flu virus
9	Cellular data: analyzing and plotting the changes in cellular populations in the blood of vaccine receptients
10	Ouchie! Vaccine reactogenicity: redness, swelling, fever. Which genes control the appearance of adverse effects to vaccine
11	Putting it all together: Finding the important correlates of vaccine efficacy

Now onto the science!

Complexity of a human model



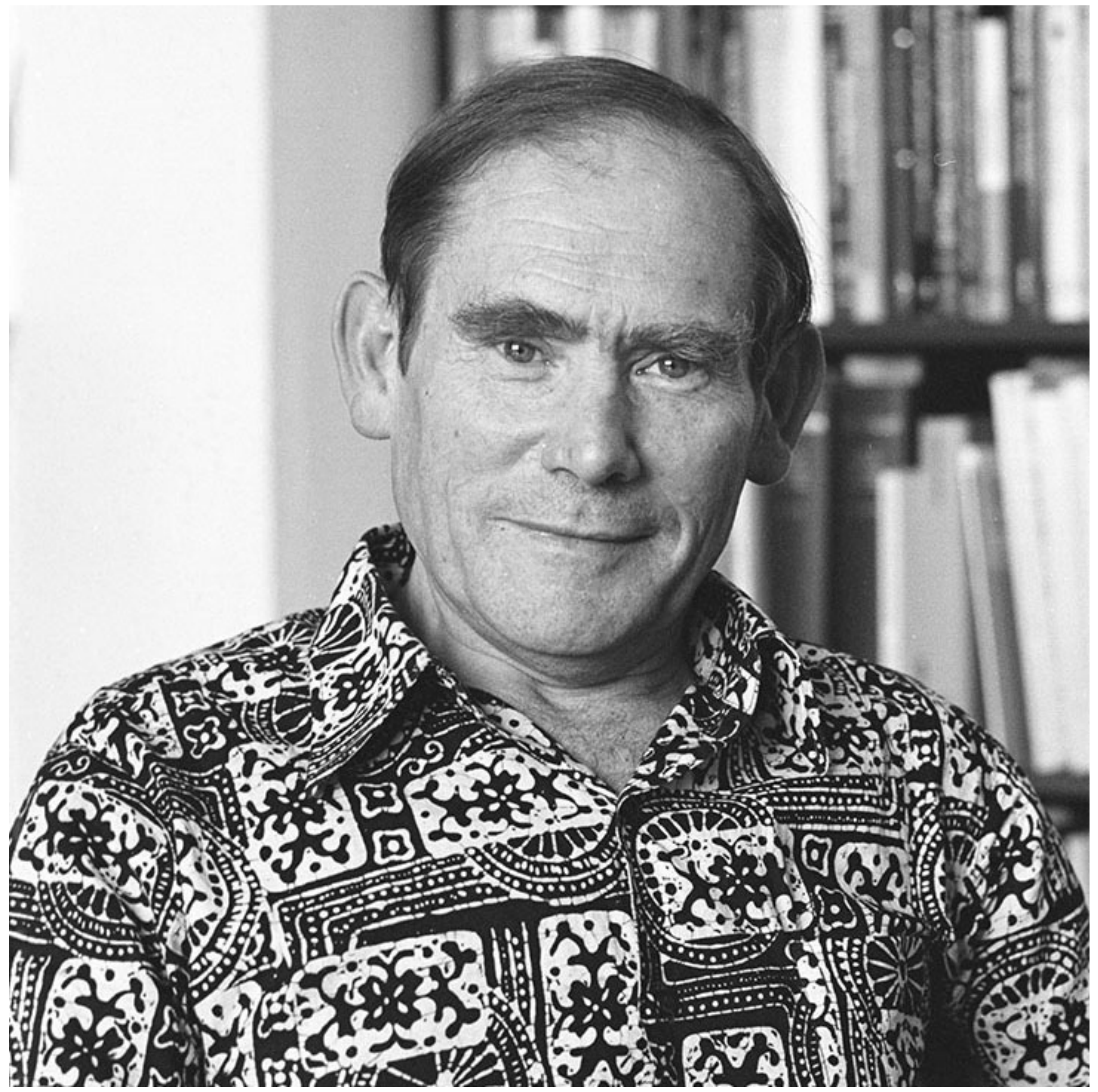
Vaccines provide a convenient way to perturb the human immune system in a controlled way. Responses to vaccine are controlled by multiple individual factors, which, combined, determine the development of protective immunity to vaccine.

The **diversity** of a human immune system can be probed by multiple **assays**. In this slide we illustrate the main lines of evidence collected in a typical systems vaccinology study:

1. Changes in gene expression in peripheral blood
2. Metabolic changes
3. Changes in the abundance of specific immune signaling molecules
4. Changes in the frequency of specific subsets of immune cells
5. Changes in antibody concentrations (titers)
6. Individual differences in genome
7. Changes in epigenetic reprogramming of immune cells

Pulendran, B. (2014). Systems vaccinology: probing humanity's diverse immune systems with vaccines. *Proceedings of the National Academy of Sciences*, *111*(34), 12300-12306.

Data vs. knowledge



Sydney Brenner
2002 Nobel Prize winner
For his contributions to the discovery of the genetic code

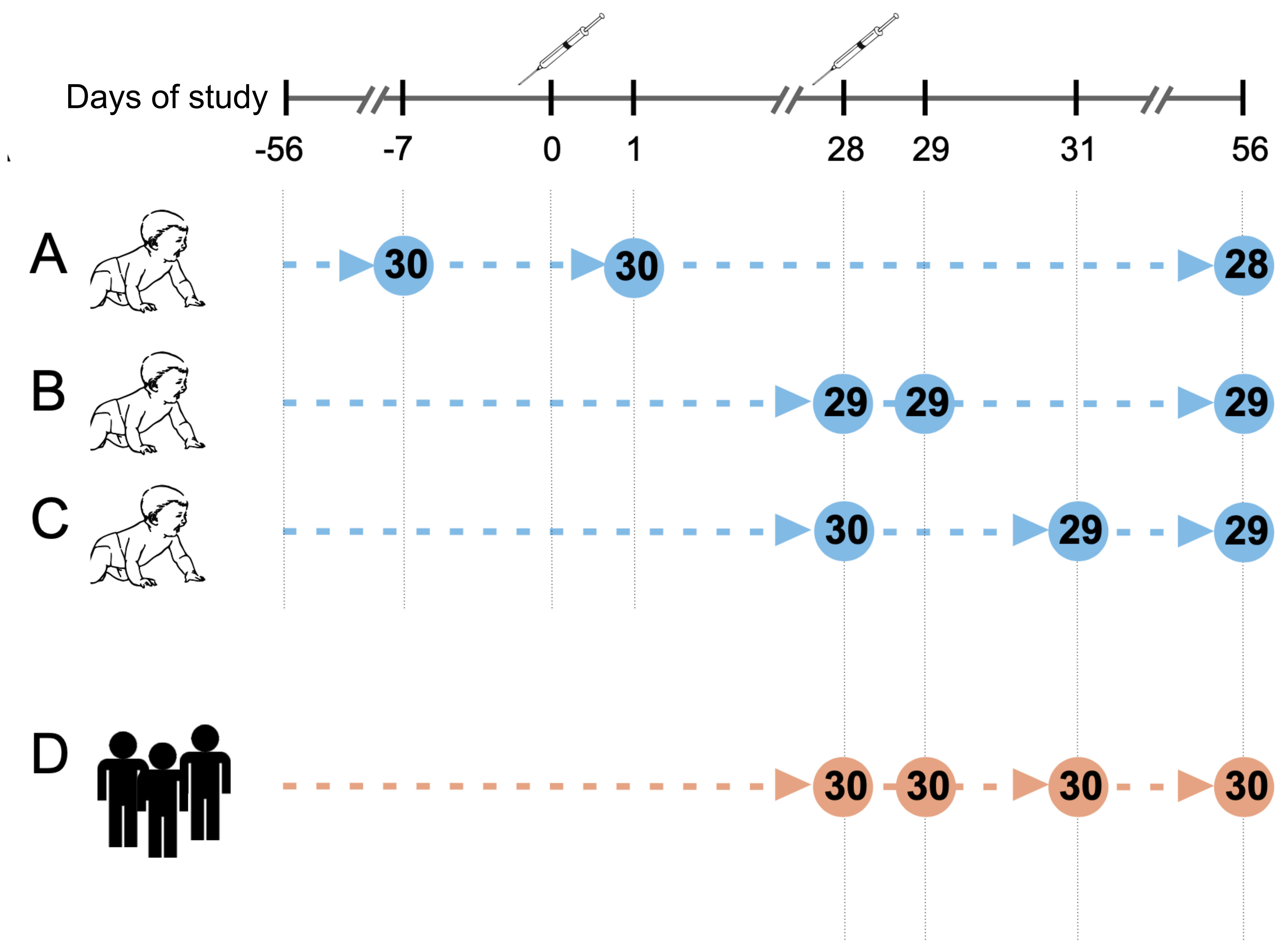
“We are drowning in a sea of data and starving for knowledge. The biological sciences have exploded, largely through our unprecedented power to accumulate descriptive facts… We need to turn data into knowledge and we need a framework to do it. “

Nobel prize lecture

The definition and purpose of bioinformatics

Wikipedia definition	My definition
Bioinformatics is an interdisciplinary field that develops methods and software tools for understanding biological data, in particular when the data sets are large and complex.	Bioinformatics is a science (or art, rather) of extracting bits of meaningful biological information from large and complex datasets, which may be noisy, incomplete, underpowered and otherwise consist of mostly irrelevant signals.
Emphasis on tool development	Emphasis on finding a pearl in a dung pile

The study



The vaccine:
Approved seasonal trivalent inactivated influenza vaccine (not a live virus)

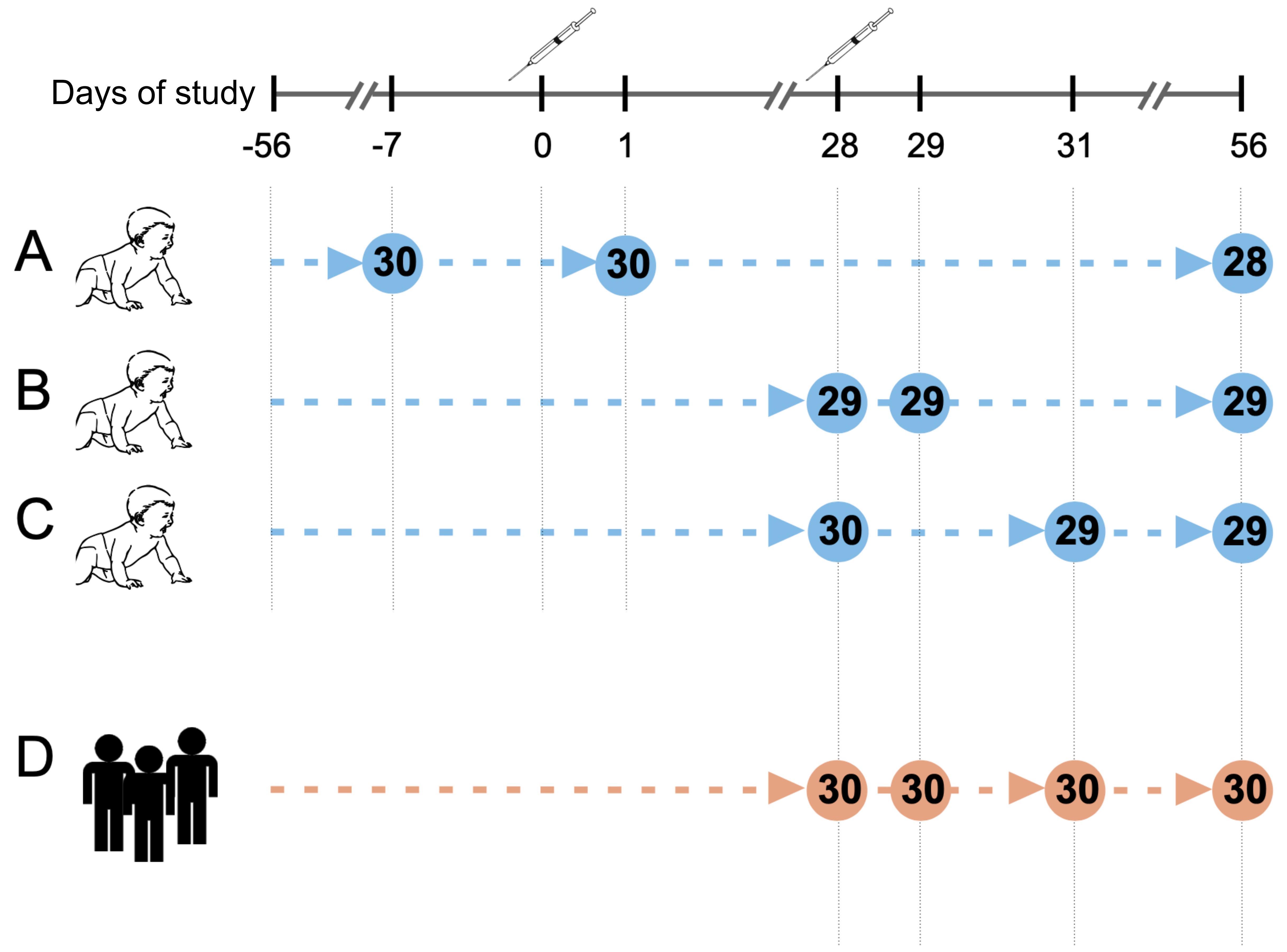
“Trivalent” means that the vaccine contains protein components of three viral strains:

- A/California H1N1
- A/Switzerland H3N2
- B/Phuket

Adjuvant:
An adjuvant is a chemical compound (or a mix of compounds) that is administered with the main intervention to augment its effects. In the case of vaccines, an adjuvant is mixed together with the vaccine and is injected at the same time.

This vaccine includes **MF-59** adjuvant, which is an oil-in-water emulsion. Addition of MF-59 has been shown to increase vaccine immunogenicity (that is, to increase the antibody titers accumulated after the vaccination).

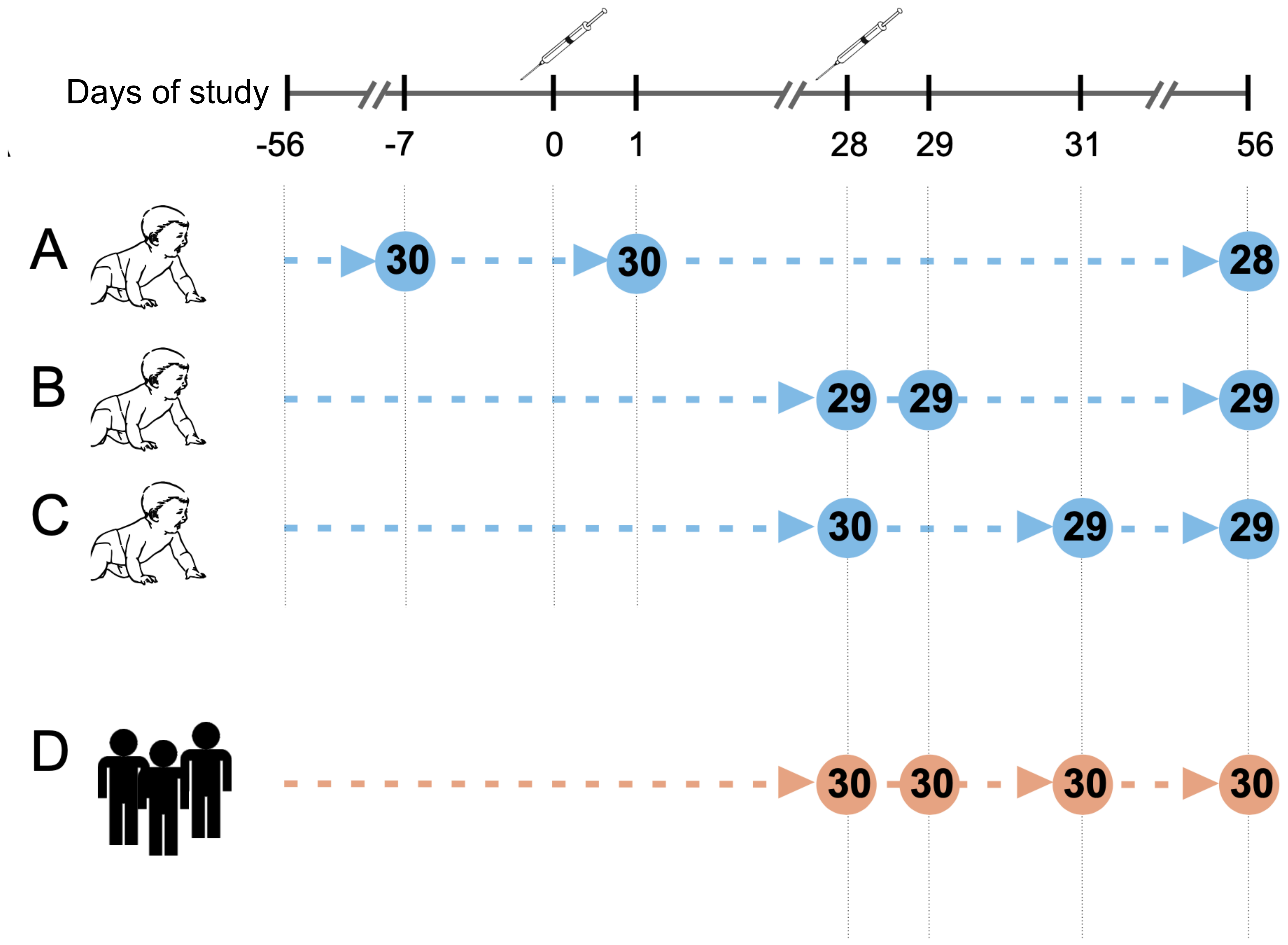
The study



What types of data have been collected?

- **Gene expression** from peripheral blood mononuclear cells (PBMCs). These cells include all white blood cells except granulocytes
- **HAI titers**, which is a measure of antibody (serology) responses to vaccine. HAI titers were collected at baseline for cohort A, at pre-boost baseline for Cohorts B and C, and at day 56 (28 days post vaccination) for all cohorts.
- **Cellular responses**. Multiple immune cell subsets were profiled pre- and post-vaccination, including dendritic cells, monocytes, macrophages and granulocytes. Changes in frequency (abundance) of the responding cell subsets give a clue to which cells play major roles in the development of an immune response to vaccine
- **Reactogenicity data (adverse effects)**. Measures such as fever, pain, redness, swelling, irritability were recorded at the baseline, the day of vaccination and up to day 7 post vaccination.

Scope and novelty



Scope:

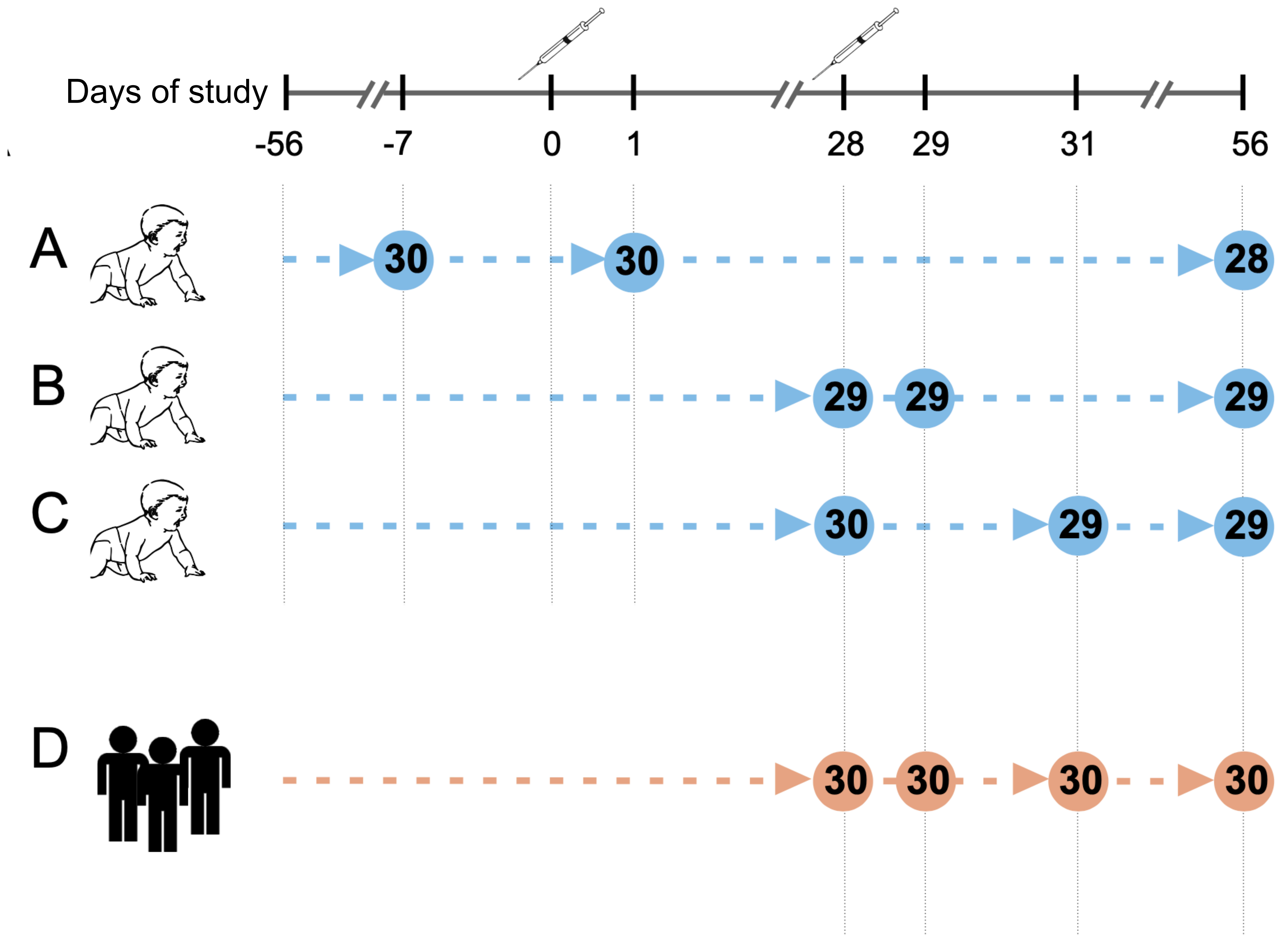
Finding the universal correlates of vaccine immunogenicity. Not all vaccine recipients respond to vaccination in the same way. Some develop high antibody titers, and are better protected against the infection, while others fail to respond. Understanding the molecular mechanisms that lead to an effective immune response is the holy grail of systems immunology.

Novelty:

Young children cohort. Children are the main recipients of vaccines, yet very little is known about the mechanisms of immune responses in children and whether they are different from those in adults.

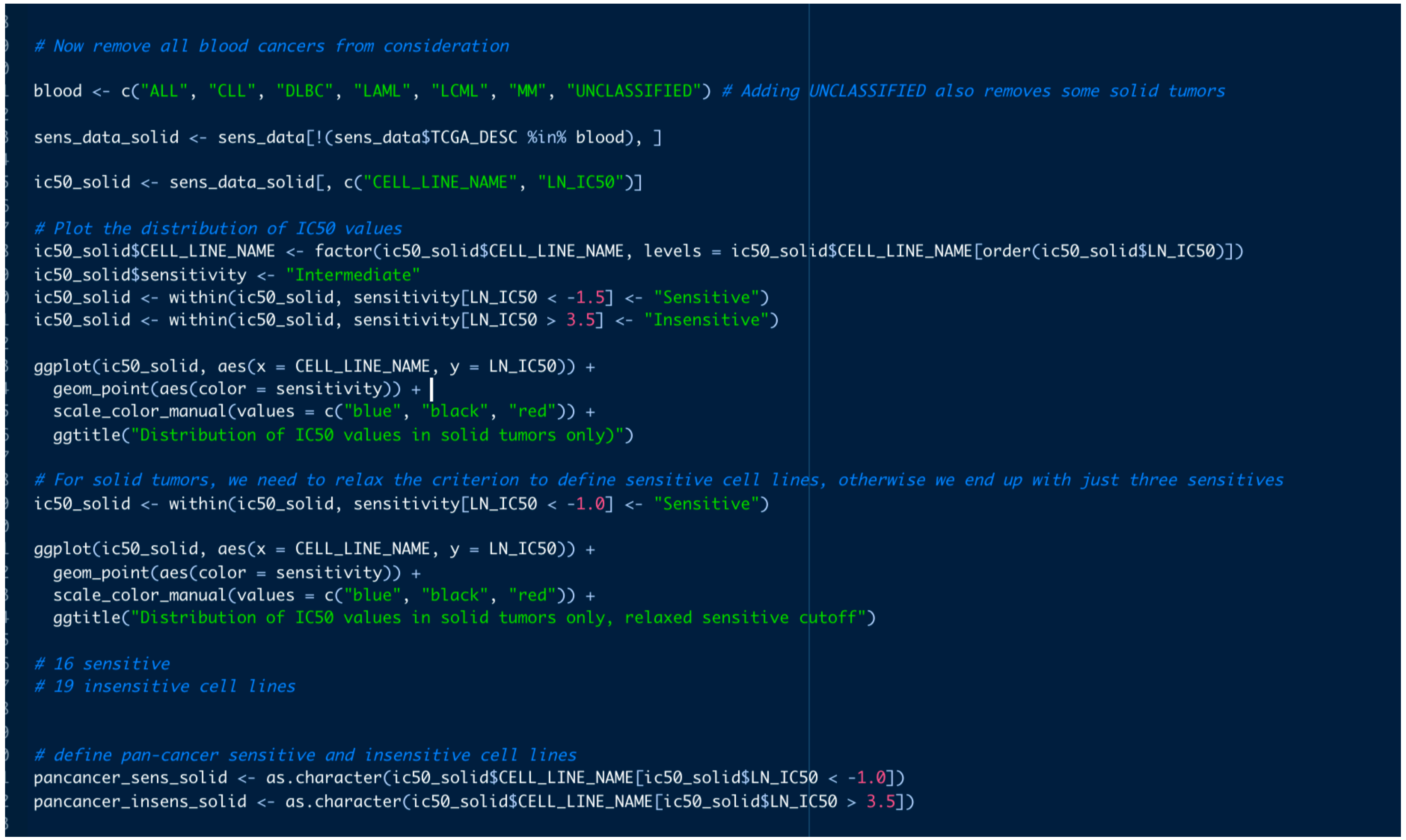
Novel vaccine. Many studies have addressed the mechanisms of action of a regular TIV vaccine, yet only one study had investigated the effects of the adjuvant co-administered with the TIV.

What do we hope to accomplish?



- **Finding the transcriptional correlates of vaccine immunogenicity** (serological responses). We can use these correlates to
  - Predict which patient will develop protective immunity to vaccine and which will not
  - Understand the basic immunological mechanisms that lead to successful immune response
- **Integration of transcriptional and cellular data** to gain the high level understanding of the immune responses to vaccination
- **Finding the correlates and predictors of vaccine adverse effects**. What are the molecular mechanisms that control adverse effects of vaccination? Can they be prevented? Can they be predicted?

How are we going to accomplish this?



```
# Now remove all blood cancers from consideration

blood <- c("ALL", "CLL", "DLBC", "LAML", "LCML", "MM", "UNCLASSIFIED") # Adding UNCLASSIFIED also removes some solid tumors

sens_data_solid <- sens_data[!(sens_data$TCGA_DESC %in% blood), ]

ic50_solid <- sens_data_solid[, c("CELL_LINE_NAME", "LN_IC50")]

# Plot the distribution of IC50 values
ic50_solid$CELL_LINE_NAME <- factor(ic50_solid$CELL_LINE_NAME, levels = ic50_solid$CELL_LINE_NAME[order(ic50_solid$LN_IC50)])
ic50_solid$sensitivity <- "Intermediate"
ic50_solid <- within(ic50_solid, sensitivity[LN_IC50 < -1.5] <- "Sensitive")
ic50_solid <- within(ic50_solid, sensitivity[LN_IC50 > 3.5] <- "Insensitive")

ggplot(ic50_solid, aes(x = CELL_LINE_NAME, y = LN_IC50)) +
  geom_point(aes(color = sensitivity)) +
  scale_color_manual(values = c("blue", "black", "red")) +
  ggtitle("Distribution of IC50 values in solid tumors only)")

# For solid tumors, we need to relax the criterion to define sensitive cell lines, otherwise we end up with just three sensitives
ic50_solid <- within(ic50_solid, sensitivity[LN_IC50 < -1.0] <- "Sensitive")

ggplot(ic50_solid, aes(x = CELL_LINE_NAME, y = LN_IC50)) +
  geom_point(aes(color = sensitivity)) +
  scale_color_manual(values = c("blue", "black", "red")) +
  ggtitle("Distribution of IC50 values in solid tumors only, relaxed sensitive cutoff")

# 16 sensitive
# 19 insensitive cell lines


# define pan-cancer sensitive and insensitive cell lines
pancancer_sens_solid <- as.character(ic50_solid$CELL_LINE_NAME[ic50_solid$LN_IC50 < -1.0])
pancancer_insens_solid <- as.character(ic50_solid$CELL_LINE_NAME[ic50_solid$LN_IC50 > 3.5])
```

By doing some coding… lots of coding

Why coding and not Excel?

120 participants
Six time points
Thousands of genes
Dozens of cellular subsets
Three vaccine strains
A spectrum of adverse effects
Four orthogonal data sets

The data is affected by subject-to-subject variation, technical noise and missing values.

We will attempt to find biologically informative patterns and correlations in this multi-dimensional cloud of data points. This is best accomplished by coding your analysis (think also reportability and reproducibility).

So, let’s start coding!

Variables

Any piece of code deals with data

In computer codes, pieces of data (or information) are stored in bits called variables.

Variables are used to store information to be referenced and manipulated in a computer program. They also provide a way of labeling data with a descriptive name, so our programs can be understood more clearly by the reader and ourselves. It is helpful to think of variables as containers that hold information. Their sole purpose is to label and store data in memory. This data can then be used throughout your code.

We can create variables throughout the code and give them names that can be descriptive of their contents. When we create a variable in R, we first type the name of the variable, followed by an assignment symbol   <-   followed by the value we want to assign to that variable.

For instance,

a <- 5

Creates a variable named “a” and assigns to it a numerical value 5. We can give a variable a more descriptive name, for instance:

five <- 5

Variables, cont’d

You can assign not only a number, but also a word to a variable. A letter or a sequence of letters and non-numeric symbols is called “a **string**”

MyString <- “Hello world!”

Here we have created a variable called “MyString” and gave it a value “Hello World!”

You can always see the contents of a variable by typing its name in the console. Names of variables are case-sensitive.

In addition to numbers and strings, you can assign a **logical** value (TRUE or FALSE) to a variable:

MyLogical <- TRUE

Manipulations with variables

Variables containing numerical values can be manipulated using mathematical operators just like the numbers they contain. For instance:

a <- 10
b <- 25
c <- b - a

Now the variable c contains the number 15.

Variables and functions

Manipulations with variables

Printing a variable:

In the above example:

MyString <- “Hello world!”

We can print the contents of the variable MyString to the console:

print(MyString)

“print” is an example of a **function**. Functions are "self contained" modules of code that accomplish a specific task. Functions usually "take in" data, process it, and "return" a result. In R a function name is always followed by round brackets (). Within these round brackets are **arguments** to the function. Arguments can be either the data that the function is designed to operate on, or modifiers, that control the exact behavior of the function. In this case “print” is a function that prints the content of its argument to the console. The variable MyString is the argument to the print function.

paste() is another example of a function. It takes its arguments and combines them into a single string, with individual elements separated by a specified by the modifier “sep=“. For instance:

x <- paste(MyString, “My name is Dmitri”, sep = “ “)

Will produce a variable x, which will contain a single string: “Hello world! My name is Dmitri”. Pay attention that there is a white space between “Hello world!” And “My name is Dmitri”, since we specified the white space as a separator. We could have replaced it with anything else, for example, the word “COOTIES”. In that case, the variable x would have the value “Hello world!COOTIESMy name is Dmitri”

Functions, cont’d

Base R and multiple available optional libraries contain hundreds, if not thousands of functions designed to perform various tasks. Sometimes, however, we want to create our own functions. For instance,

```
zscore <- function(x) {
z <- (x - mean(x)) / sd(x)
return(z)
}
```

Will create a function that will calculate a z-score for each element of an array x, containing several numbers. You can call this function from anywhere in the code by typing, for example:

```
MyZscore <- zscore(c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10))
```

In this example, c() is also a function, that combines several elements (in this case, numbers from 1 to 10) into a single array. An expression “*zscore(c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10))*” is an example of nested functions.

Homework

1. Create a variable “a” and assign it a value “My name”
2. Create a variable “b” and assign it a value corresponding to your first name followed by a dot.
3. Using the paste() function with separator “ is “ (mind the white spaces), generate a variable “c” with a value “My name is YourFirstName.”
4. Create a variable “d” with a numerical value corresponding to your age in years.
5. Create a variable “e” with a value corresponding to your birth year by subtracting the variable “d” from 2021 (assuming that you didn’t have a birthday in 2022).
6. Using the paste() function combine variables “c” and “e” and a string “I was born in” to create a variable “f” with the value “My name is YourFirstName. I was born in YourBirthYear”. Be sure that the individual elements in this sentence are separated by a white space.
7. Print the variable “f” to the console.

Save your code to a file. Be sure that the file name contains your name and the string “lesson1”. Upload your homework to the “homework” branch in the Github repository.

If you get stuck with the code and cannot get the desired output, open an issue under the “issues” tab on Github. Be sure to include your code, and we will see why it isn’t working as intended.

Have fun!