Welcome to DATA 151

I'm so glad you're here!

DATA 151: CLASS 4A
INTRODUCTION TO DATA SCIENCE (WITH R)

BASICS OF PROGRAMMING

NOTES PREPARED BY PROF. KITADA SMALLEY (FALL 2022)

ANNOUNCEMENTS

RELEVANT READING

Introduction to Data Science:

- Tuesday:
  - Ch 3: Programming basics
- Thursday:
  - Ch 4: Tidyverse

HOMEWORK REMINDER

Due this week:

- *HW #3: DC Introduction to Programming in R*
  - ***No submission on WISE necessary, do on DataCamp***
- *Project Milestone #1: Project Proposal*
  - ***Due on WISE 9/22***
  - *One submission per group*
  - *Start scheduling group meetings*

TAKING TIME TO CHECK-IN

CHECKING IN

We care about you and we're here support you!

HOW CAN WE HELP?

- ***How do I navigate WISE?***
- ***How do I submit assignments to WISE?***
- ***How do I do tutorials on DataCamp?***
- ***Where can I get help on content?***

THE QUAD CENTER

Quantitative Understanding, Analysis and Design

FORD 224

HOURS:

Monday 6:00 - 9:00 PM
Tuesday 6:00 - 9:00 PM
Wednesday 3:00 - 9:00 PM
Thursday 3:00 - 9:00 PM
Friday 3:00 - 4:30 PM
Sunday 3:00 - 6:00 PM

R MARKDOWN

R MARKDOWN

R Markdown files are designed to be used in three ways:

1. For communicating to decision makers, who want to focus on the conclusions, not the code behind the analysis.
2. For collaborating with other data scientists (including future you!), who are interested in both your conclusions, and how you reached them (i.e. the code).
3. As an environment in which to *do* data science, as a modern day lab notebook where you can capture not only what you did, but also what you were thinking.

TYPES OF CONTENT

It contains three important types of content:

1. An (optional) **YAML header** surrounded by **---**s.
2. **Chunks** of R code surrounded by ```.
3. Text mixed with simple text formatting like # heading and _italics_.

HOT KEYS!

- Insert new chunk
  - Cmd/Ctrl + Alt + I
- Run code in chunk
  - Cmd/Ctrl + Shift + Enter
- Knit entire document
  - Cmd/Ctrl + Shift + K

FORMATTING

```
Text formatting

--------------------------------------------------


*italic*  or _italic_

**bold**   __bold__

`code`

superscript^2^ and subscript~2~
```

FORMATTING

```
Headings

---------------------------------------------------------


# 1st Level Header


## 2nd Level Header


### 3rd Level Header
```

FORMATTING

```
Lists
---------------------------------------------------------


*   Bulleted list item 1

*   Item 2

    * Item 2a

    * Item 2b

1.  Numbered list item 1

1.  Item 2. The numbers are incremented automatically in the output.
```

CHUNK OPTIONS

Option	Run code	Show code	Output	Plots	Messages	Warnings
`eval = FALSE`	-		-	-	-	-
`include = FALSE`		-	-	-	-	-
`echo = FALSE`		-				
`results = "hide"`			-			
`fig.show = "hide"`				-		
`message = FALSE`					-	
`warning = FALSE`						-

RECALL THE EXPERIMENT

LEARNING OBJECTIVES

Learning Objective:

In this session students will learn how to create an R markdown document that basics of programming through a case study of creating a randomized experiment.

- How to create an R Markdown document: `.Rmd`
    - Creating headers at different levels to organize and navigate the document
    - Creating and working with code chunks
- Basics of programming:
    - Creating a sequence of consecutive integers
    - Creating and working with matrices: `matrix()`
        - Matrix indexing
    - Creating random values from R: `sample()`
    - Working with `for` loops
    - Using the `%in%` operator to define group membership
    - Using conditional statements: `if()`
    - How to concatenate values together to form a vector: `c()`

THE MOTIVATION

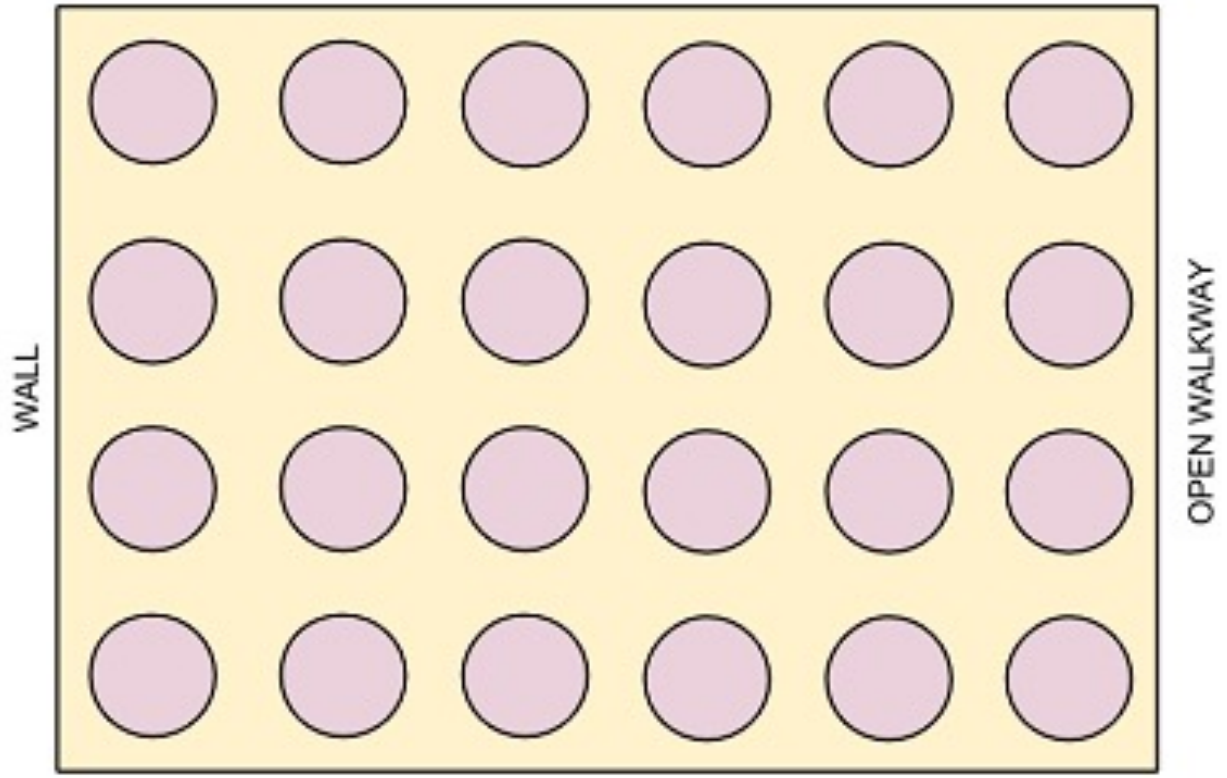
Consider the set up:

- In a greenhouse experiment we want to study a single factors (fertilizer) with 4 levels
- We have enough space for 24 experimental units (a potted plant)
- To maintain balance in the experiment, we will have 6 replications of each treatment

Source: https://newonlinecourses.science.psu.edu/stat502/node/175/

COMPLETELY RANDOMIZED DESIGN

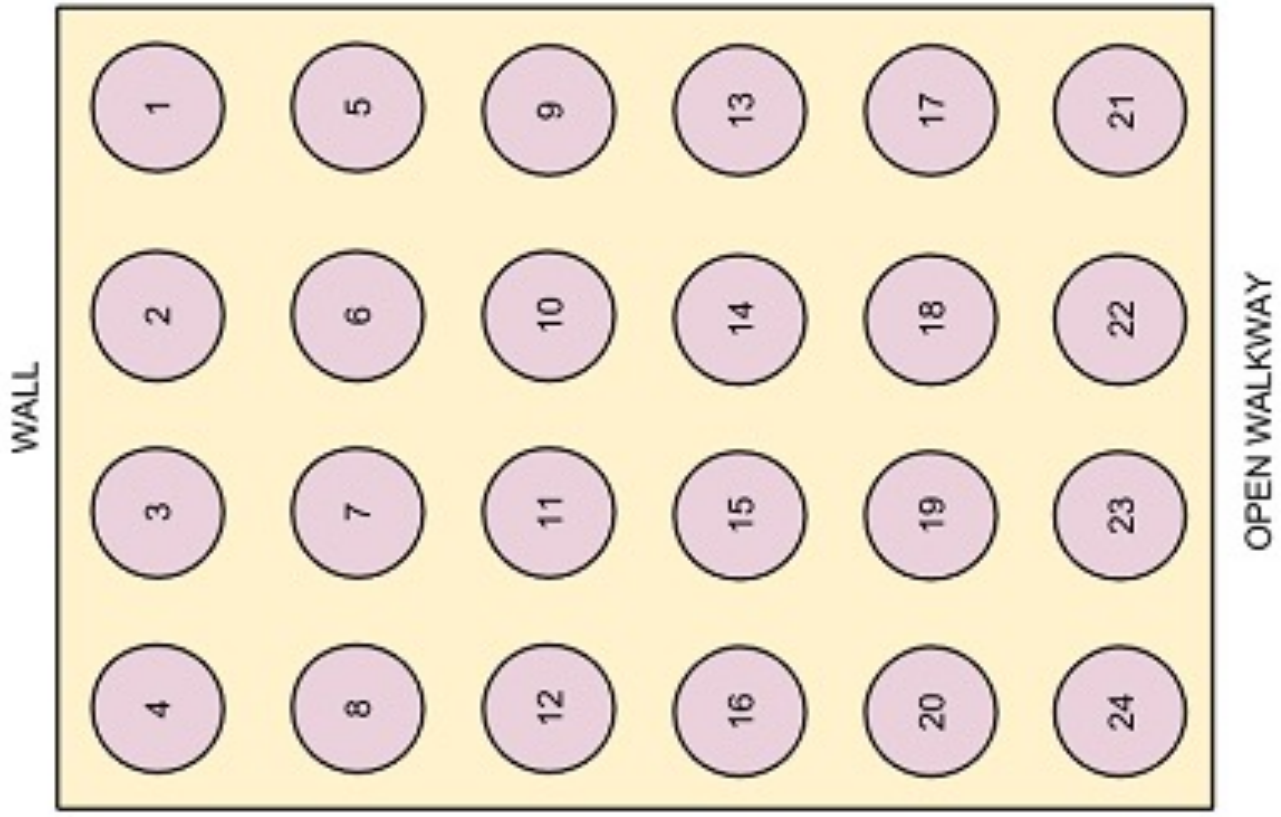
Greenhouse Diagram and bench used for the experiment (viewed from above):



Source: https://newonlinecourses.science.psu.edu/stat502/node/175/

COMPLETELY RANDOMIZED DESIGN

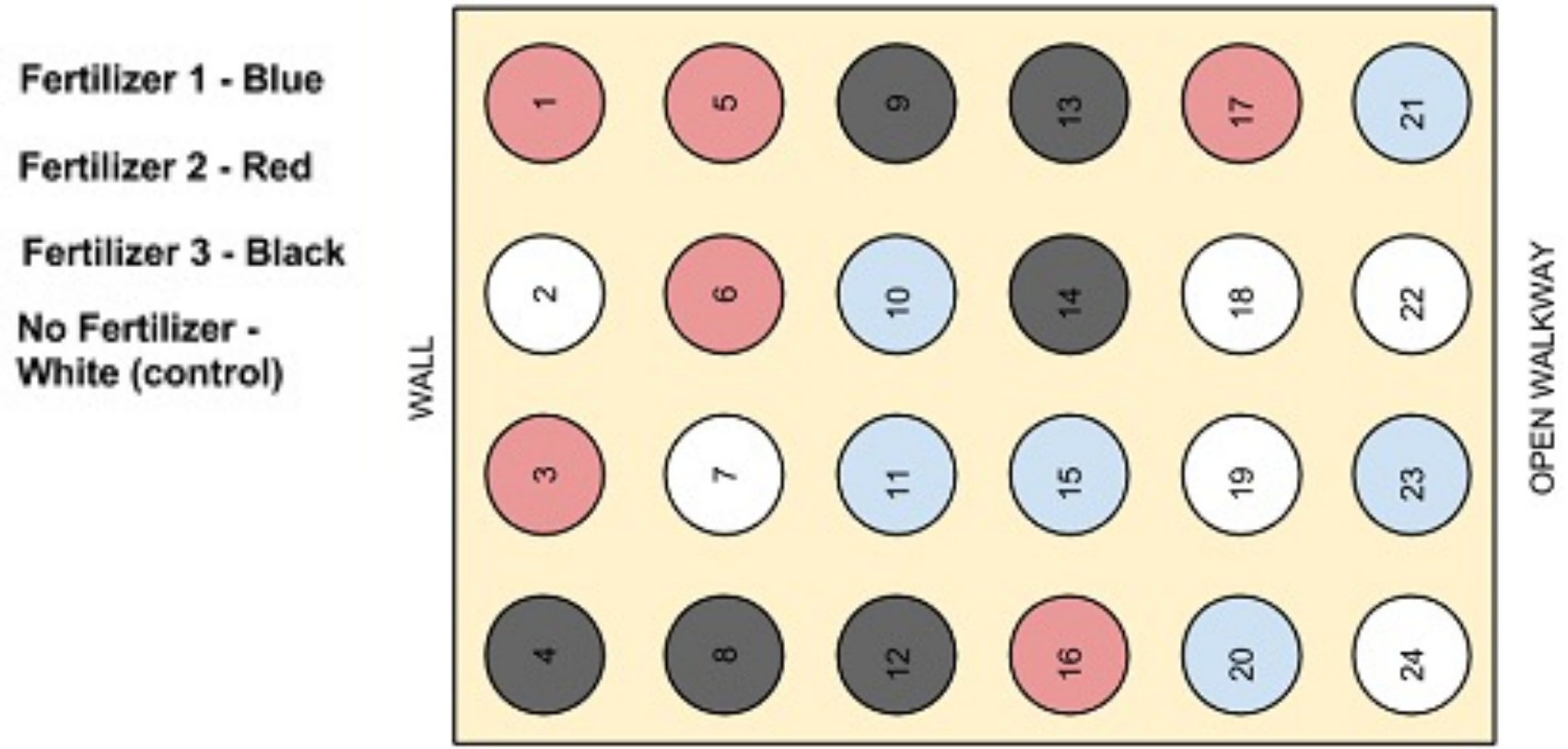
Step 1: Assign it experimental unit a unique id

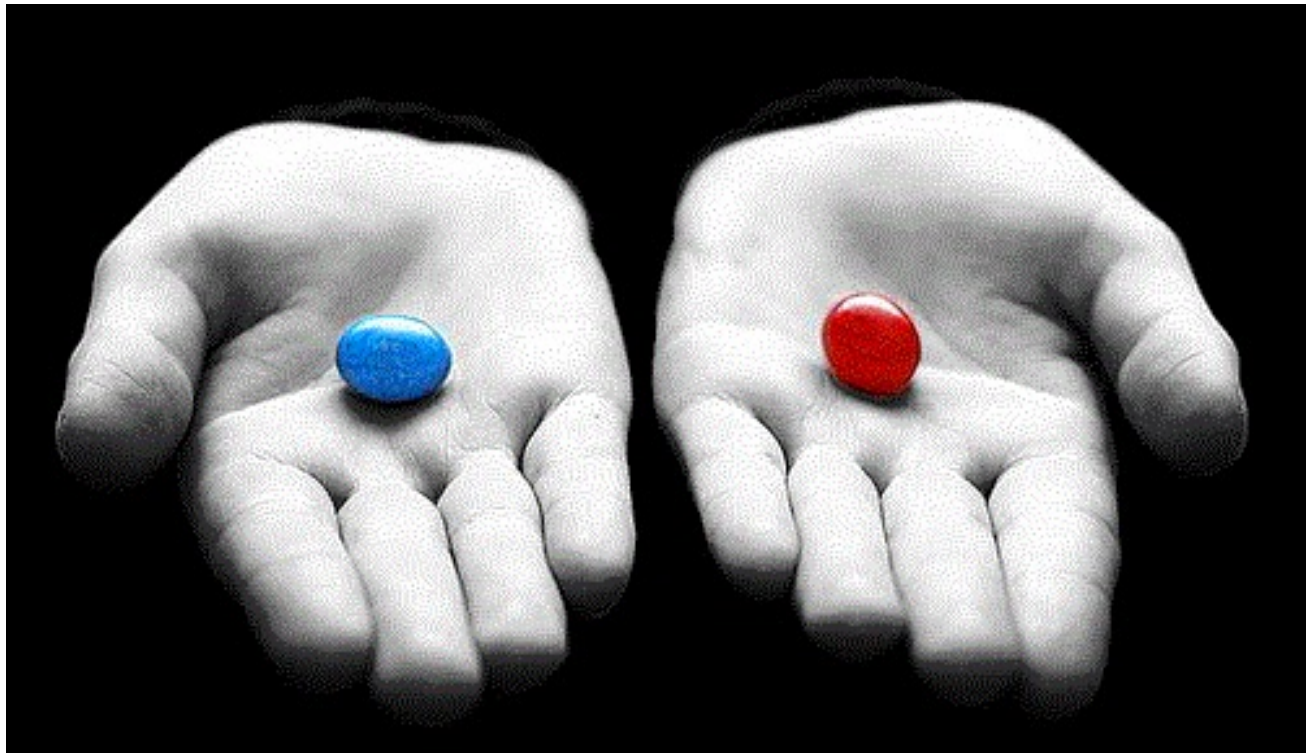


Source: https://newonlinecourses.science.psu.edu/stat502/node/175/

COMPLETELY RANDOMIZED DESIGN

Step 2: Randomly assign each experimental units to treatments





LET'S ACT THIS OUT!

LET'S ACT THIS OUT!

I need 6 volunteers please ☺

LET'S ACT THIS OUT!

Step 1: Each student volunteer gets an ID

LET'S ACT THIS OUT!

Step 2: ID's are randomly mixed up

LET'S ACT THIS OUT!

Step 3: ID's are separated into two groups

LET'S ACT THIS OUT!

Step 4: Students are assigned treatments

- **Group 1 will get a blue card**
- **Group 2 will get a red card**

PART II: BASICS OF PROGRAMMING

HOW CAN WE USE R TO DESIGN AN EXPERIMENT?

SEQUENCE OF INTEGERS

1. Creating ID's

We want to give each experimental unit an ID. Since ultimately we arae going to randomly draw from this list of IDs we can just assign number IDs from 1 to 24. This can be done by using the colon (`:`). The syntax for the colon function is `starting integer : ending integer` .

```
# STEP 1: Giving Id's to Experimental Units
# recall in our example from class there were 24 plants (experimental units)
# the colon is a way to create a vector of consecutive integers

ids<-1:24
# note that we are storing the output as a vector
ids
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
```

LEARNING BY DOING! (#1)

Learning by doing:

- Q1: What happens when you go from a bigger number to a smaller number?

```
# INSERT CODE HERE #
```

- Q2: What happens when you go have a postitive and negative number?

```
# INSERT CODE HERE #
```

WORKING WITH MATRICES

2. Organizing Your Lab Experiment

In the example we saw, the lab bench had 4 rows and 6 columns. We can use the `matrix` function to organize all our experimental units. Matrices have two dimensions, rows and columns (in that order). A matrix is a very useful way to store numbers there are also special mathematical operations that can be performed on matrices.

Start by reading the documentation about matrices:

```
# let's read the documentation about matrix function to learn about it arguments
?matrix

# the inputs of this function are the data, nrow, and ncol
```

WORKING WITH MATRICES

Now let's organize our experimental units:

```
## STEP 2: Organizing the experimental units into rows and columns
# in the example we saw, the lab bench had 4 rows and 6 columns
labBench<-matrix(ids, nrow=4, ncol=6)
labBench
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    1    5    9   13   17   21
## [2,]    2    6   10   14   18   22
## [3,]    3    7   11   15   19   23
## [4,]    4    8   12   16   20   24
```

```
# this is a matrix!
```

LEARNING BY DOING! (#2)

Learning by doing:

- What would happen if the number of columns was omitted? (ie don't include `ncol`)

```
# INSERT CODE HERE #
```

- What if product of the dimensions `nrow` × `ncol` is not equal to the length of the data?

```
# INSERT CODE HERE #
```

RANDOMLY ASSIGNING TREATMENTS

3. Randomly Assigning Treatments

We want to randomly assign treatments to our experimental units to avoid confounding. We can use R to help us with this task.

The simplest form of an experimental design is a *Completely Randomized Design*. In this design we choose ID's and randomly assign to treatments. In order to choose which IDs will go in which treatments, we can use the `sample()` function.

First, let's learn about the sample function:

```
?sample
```

LEARNING BY DOING! (#3)

Learning by doing: `sample`

Let's try it!

```
crd_samp<-sample(ids, replace=FALSE)
crd_samp
```

```
##  [1] 11 12  3 16  5 17 23 20  8 10 13 15 19 14 18 24 21  1  7  2 22  6  9  4
```

- What happened?

```
ANSWER HERE:
```

- Is the order of numbers that same as your neighbor's?

```
ANSWER HERE:
```

COMPLETELY RANDOMIZED DESIGN

- Also known as **CRD**
- The simplest experimental design, in terms of analysis and convenience
- Subjects are randomly assigned to treatments
- Typically done by listing treatment levels and randomly assigning random numbers to each

SAMPLING TO RANDOMLY ASSIGN TREATMENTS

Now let's try assigning our IDs to treatments using a matrix. Here `nrow` will correspond to the number of treatment.

Let's say that

- Row 1 = Treatment A
- Row 2 = Treatment B
- Row 3 = Treatment C
- Row 4 = Treatment D (Control)

SAMPLING TO RANDOMLY ASSIGN TREATMENTS

A. Completely Randomized Design

```
## Completely randomized design
## choose ID's and randomly assign to treatments

# nrow will be the number of treatments
crd_mat<-matrix(crd_samp, nrow=4)
crd_mat
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]   11    5    8   19   21   22
## [2,]   12   17   10   14    1    6
## [3,]    3   23   13   18    7    9
## [4,]   16   20   15   24    2    4
```

SAMPLING TO RANDOMLY ASSIGN TREATMENTS

```
# we can also rename the rows
rownames(crd_mat)<-c("Treat A", "Treat B", "Treat C", "Treat D")

crd_mat
```

```
##         [,1] [,2] [,3] [,4] [,5] [,6]
## Treat A   21   10   18   16    5   22
## Treat B    8   14    3    9    7   15
## Treat C   12   17    2    1    6   20
## Treat D   13   23    4   19   24   11
```

BUT WAIT A MINUTE!!

Making an Experiment Map!

We can use the matrix from the previous step to know which experimental units are in their respective treatments; however, it might be easier if we made a map that showed where their treatments were located. To accomplish this task we'll need to understand `for` loops, the `%in%` operator, and conditions.

WHAT DOES A LOOP DO?

CODE

OUTPUT

```
## FOR LOOPS
for(i in 1:5){
  print(i)
}
```

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

What does a "for loop" do?

THE %IN% OPERATOR

```
> ## %in% operator
> 1 %in% c(1, 2, 3)
[1] TRUE
> 5 %in% c(1, 2, 3)
[1] FALSE
```

Based on these examples, what do you think the %in% operator does?

Bringing it all together!

This is a little complicated, so I will provide the code. Let's pay careful attention to how each piece is working together.

```
## Making a map of this design
## here we will learn to create a loop and how to write conditional
s
treats<-matrix(nrow=24)
for(i in 1:24){
  if(i %in% crd_mat[1,]){
    treats[i]<-"A"
  }
  if(i %in% crd_mat[2,]){
    treats[i]<-"B"
  }
  if(i %in% crd_mat[3,]){
    treats[i]<-"C"
  }
  if(i %in% crd_mat[4,]){
    treats[i]<-"D"
  }
}

treats
```

MAPPING TREATMENTS TO IDS

```
## make the map!
expDes<-matrix(treats, nrow=4)
expDes
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,] "C"  "A"  "B"  "D"  "C"  "A"
## [2,] "C"  "C"  "A"  "B"  "A"  "A"
## [3,] "B"  "B"  "D"  "B"  "D"  "D"
## [4,] "D"  "B"  "C"  "A"  "C"  "D"
```

Food for Thought:

- How does this look?
- Does it appear “random”?

MAPPING TREATMENTS TO IDS

Note: We might be tempted to think that just because there are clusters of the same treatment together that the design is not random; however, we used a random mechanism to assign IDs to treatment. Using a random mechanism does not guarantee that there won't be these clusters.

MATRIX INDEXING

Matrix Indexing

We have been using matrices a lot! When working with matrices its a good idea to know can we can call specific subsets within the matrix. Every cell of a matrix has an address which is defined by the row and column that its in.

In the following examples we will see how this can be used:

LEARNING BY DOING! (#4)

Learning by doing!

- What treatments is to be assigned in row 3 column 2?

```
# EXAMPLE:
expDes[3,2]
```

```
## [1] "B"
```

- What treatments are to be assigned in row 3?

```
# INSERT CODE HERE #
```

- What treatments are to be assigned in column 4?

```
# INSERT CODE HERE #
```

LEARNING BY DOING! (#4) – CHALLENGE!

- If a cat knocked over the plants in the 4th row what would the rest of my experiment look like?

```
# INSERT CODE HERE #
```

RANDOMIZED (COMPLETE) BLOCK DESIGN

B. Blocked Design

If we know that there is a natural gradient or more homogeneous subgroups within our experiment we might consider blocking to improve our design. The hallmark of a ***randomized complete block design*** is that every treatment must be present in ever block. This allows us to avoid confounding treatment with block.

In this example columns are used as blocks. We will randomly assign where the treatments are placed within which block.

RANDOMIZED (COMPLETE) BLOCK DESIGN

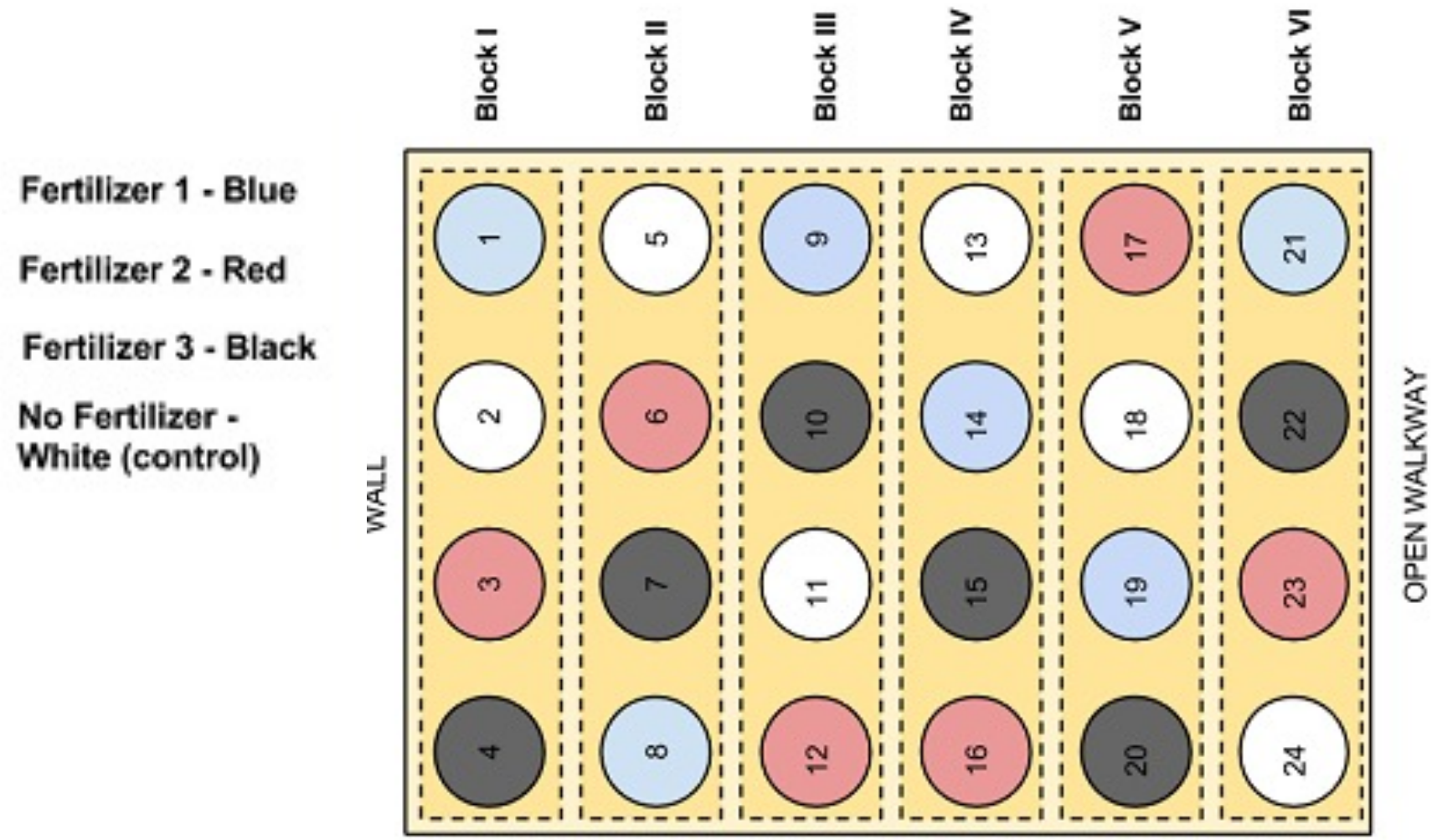
In a block design, the random assignment of experimental units to treatments is carried out within each block

What are the steps in performing a blocked experiment?

1. Form groups (blocks)
   - All individuals within each block should be similar in regard to the lurking variable

2. Within each block, randomly assign experimental units to each treatment.

RANDOMIZED (COMPLETE) BLOCK DESIGN



RANDOMIZED BLOCK DESIGN AND CONCATENATION

```r
# we will learn how to concatenate here
# start with an emply list
blockTreats<-c()
for(i in 1:6){
  thisSample<-sample(c("A", "B", "C", "D"), replace=FALSE)
  blockTreats<-c(blockTreats, thisSample)
}

blockDes<-matrix(blockTreats, nrow=4)
blockDes
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,] "B"  "B"  "D"  "A"  "A"  "A"
## [2,] "A"  "C"  "A"  "D"  "B"  "C"
## [3,] "C"  "D"  "B"  "C"  "D"  "D"
## [4,] "D"  "A"  "C"  "B"  "C"  "B"
```