Statistical Analysis using R

IITA
Transforming African Agriculture

Data Wrangling

Ibnou Dieng
Kayode Fowobaje
Sam Ofodile
Moshood Bakare
Trushar Shah
Margaret Karanja

IITA Biometrics Unit

01-02 & 07-09 November 2021

Course Overview

1. ~~Short Introduction to R and RStudio~~
2. ~~Preparation of Data for Statistical Analysis~~
3. Data wrangling
4. Experimental Designs for Plant Breeding
5. ANOVA and MET analysis
6. Multivariate analysis
7. Graphics in R with ggplot2

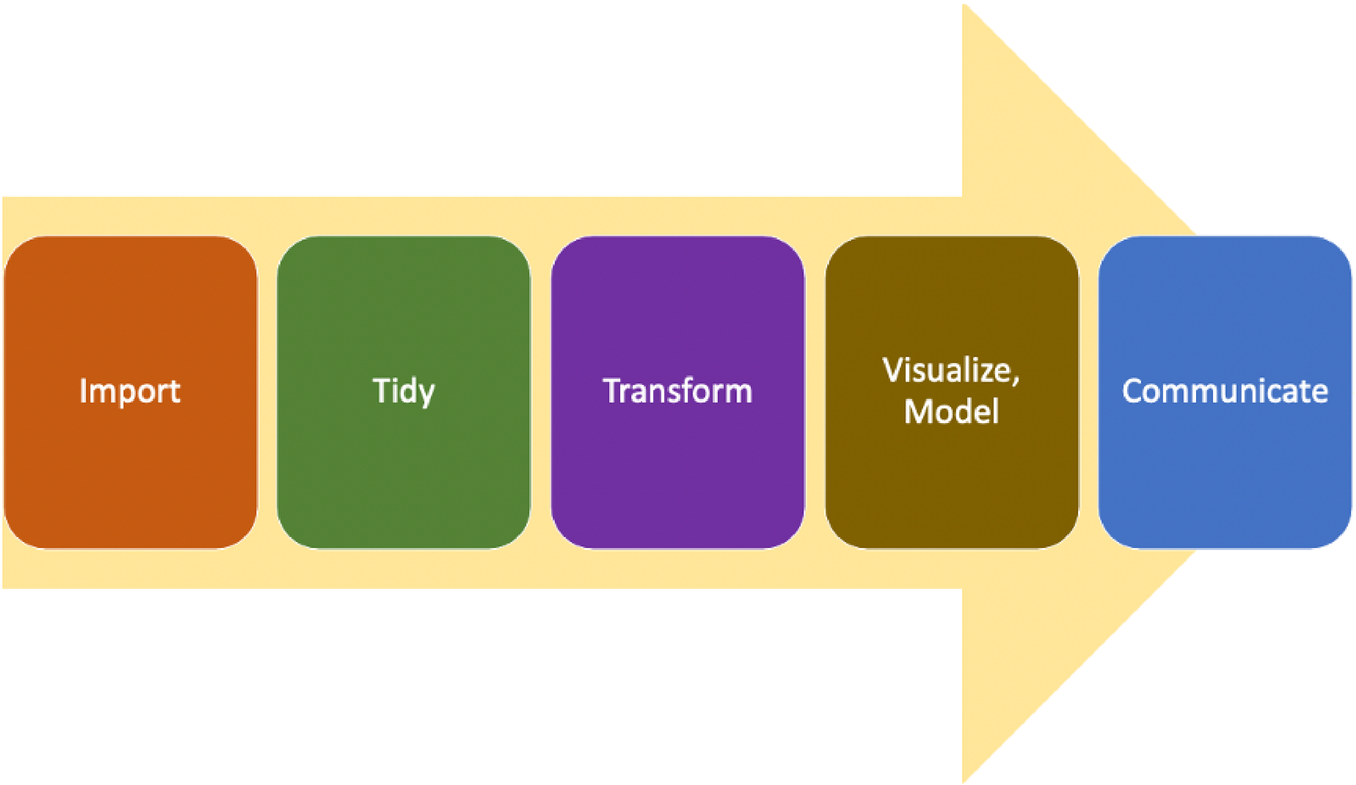
Recap

- **R** is a free software environment for **statistical** computing and graphics
    - We are not *only* learning **R**
    - We are not *only* learning **statistics**
- We want to
    - understand/solve statistical problems from the agricultural sector using **R**
    - interpret results generated by **R**
- What have we learned so far:
    - **R** is the engine and **RStudio** is the dashboard
    - **RStudio** interface divided into four quadrants: **Q1** (scripts); **Q2** (console); **Q3** (environment); **Q4** (help, plot)
    - What's a **Package**: like a third-party apps on your phone
    - What's a Project: "RStudio projects make it straightforward to divide your work into multiple contexts, each with their own working directory, workspace, history, and source documents"

IITA Transforming African Agriculture

Introduction

The Data Workflow (Hadley Wickham)



Data Frame

- A vector is a variable
- A factor is a categorical variable
- A data frame is a table composed with one or several vectors and/or factors all of the same length. It's a two-dimensional array in which each column contains values of one variable and each row contains one set of values from each column
- Following are the characteristics of a data frame:
  - The column names should be non-empty
  - The row names should be unique
  - The data stored in a data frame can be of numeric, factor or character type.
  - Each column should contain same number of data items

Data Frame: Example

env	loc	year	gen	yield	height	lodging	size	protein	oil
L70	Lawes	1970	G01	2.387	1.445	4.25	8.45	36.7	20.895
L70	Lawes	1970	G02	2.282	1.45	4.25	9.95	37.55	20.74
L70	Lawes	1970	G03	2.567	1.46	3.75	10.85	37.8	21.295
L70	Lawes	1970	G04	2.877	1.26	3.5	10.05	38.45	21.99
L70	Lawes	1970	G05	2.392	1.335	3.5	11	37.5	22.13
L70	Lawes	1970	G06	2.408	1.36	4	11.75	38.25	21.16
L70	Lawes	1970	G07	2.699	1.3	3	11.75	37.35	21.7
L70	Lawes	1970	G08	2.457	0.955	3.25	10	35.2	21.145
L70	Lawes	1970	G09	2.567	1.03	3	11.25	35.9	21.495
L70	Lawes	1970	G10	2.984	1.155	3.75	10.85	39.7	20.43
L70	Lawes	1970	G11	1.663	1.42	4.5	6.95	40.25	19.09
L70	Lawes	1970	G12	1.964	1.435	4.25	8.35	40.3	18.745
L70	Lawes	1970	G13	1.472	1.585	4.5	9.3	41.15	19.18
L70	Lawes	1970	G14	2.72	1.33	4	8.25	37.4	20.76
L70	Lawes	1970	G15	2.22	1.37	4.25	9.3	36.65	20.685
L70	Lawes	1970	G16	1.655	1.7	4.75	9.15	39.65	20.435
L70	Lawes	1970	G17	1.722	1.28	4.25	8.4	43.7	17.455
L70	Lawes	1970	G18	1.432	1.495	4.5	7.8	42.4	17.4

- A data frame is a matrix-like structure

```
mydata[1:nrow, 1:ncol]
```

Data Frame

- Let's create a data frame using the **RStudio** console with the command

```
trial01 <- data.frame(variety = c("G01-US234", "G05-BT456", "Ind01",
    "G11-DR234"), yield = c(6323.3, 2515.2, 5611, 7729, 7843.25), he
    95.2, 113, 89.45, 145.67))
```

- We have a data frame with 3 variables (variety, yield and height) and 5 observations. The data frame is stored under an object named **trial01**

- We can print the data frame

```
trial01
```

```
##     variety   yield height
## 1 G01-US234 6323.30 123.30
## 2 G05-BT456 2515.20  95.20
## 3     Ind01 5611.00 113.00
## 4 G11-US244 7729.00  89.45
## 5 G11-DR234 7843.25 145.67
```

Data Frame

- We can extract the first three rows

```
trial01[1:3, ]
```

```
##     variety  yield height
## 1 G01-US234 6323.3  123.3
## 2 G05-BT456 2515.2   95.2
## 3     Ind01 5611.0  113.0
```

- We can extract the first two columns

```
trial01[, 1:2]
```

```
##     variety   yield
## 1 G01-US234 6323.30
## 2 G05-BT456 2515.20
## 3     Ind01 5611.00
## 4 G11-US244 7729.00
## 5 G11-DR234 7843.25
```

Data Frame

- We can extract "from 3rd to 5th row" with "2nd and 3rd column"

```
trial01[3:5, 2:3]
```

```
##     yield height
## 3 5611.00 113.00
## 4 7729.00  89.45
## 5 7843.25 145.67
```

- We can extract specific column from a data frame using column name

```
trial01$yield
```

```
## [1] 6323.30 2515.20 5611.00 7729.00 7843.25
```

Data Frame

- We can add a column vector using a new column name

```
trial01$flowering <- c(87, 101, 88, 120, 90)
trial01
```

```
##     variety   yield height flowering
## 1 G01-US234 6323.30 123.30        87
## 2 G05-BT456 2515.20  95.20       101
## 3     Ind01 5611.00 113.00        88
## 4 G11-US244 7729.00  89.45       120
## 5 G11-DR234 7843.25 145.67        90
```

IITA
Transforming African Agriculture

The tidyverse package

- Data can be entered from the console and stored in data frame
- Usually, you must import your data into **R**. This typically means that you take data stored in a file, database, etc., and load it into a data frame in **R**
- We'll work with **tibbles** instead of **R**'s traditional **data.frame**
- **R** is an old language, and what was useful 10 or 20 years ago can now get in your way
- We opt for the **tidyverse** package, an opinionated collection of **R** packages designed for data science, mainly data wrangling (import and transform) – much more intuitive and easier
- All **tidyverse** packages share an underlying design philosophy, grammar, and data structures

The tidyverse package

- We can install the complete **tidyverse** with

```
install.packages("tidyverse")
```

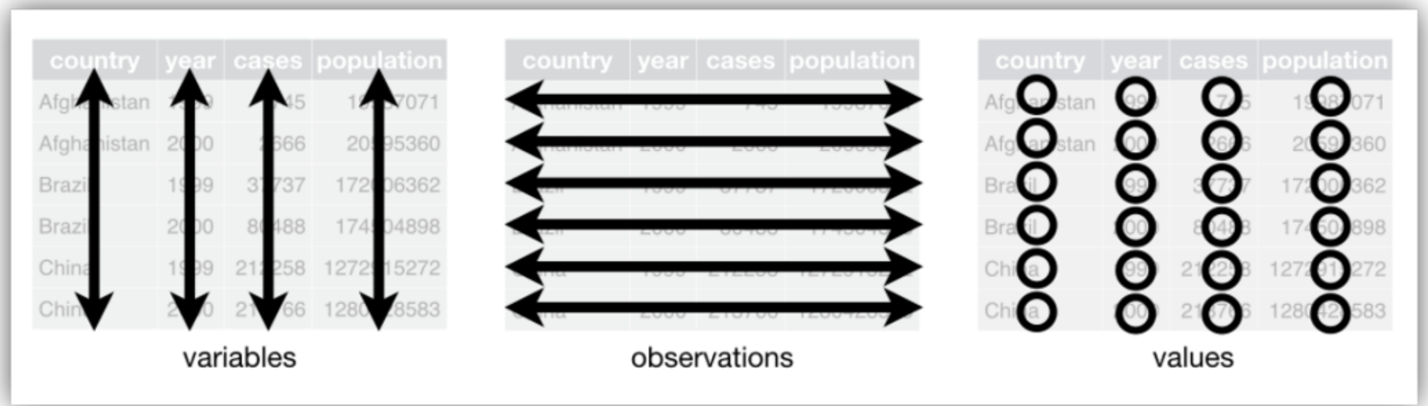
- The core tidyverse includes these packages: `ggplot2`, `tibble`, `tidyr`, `readr`, `purr`, `dplyr`, `stringr`, `forcats`
- The 8 packages are launched togother via the same call

```
library(tidyverse)
```

- `tidyverse` also includes other packages with more specialized usage. They are not loaded automatically with `library(tidyverse)`, need to load each one with its own call to `library(package_name)`
- **Tidy** data is a standard way of mapping the meaning of a dataset to its structure.
- Tibbles are Data Frames, but they make life a little easier

The tidyverse package

- In tidy data:
  - Each variable forms a column
  - Each observation forms a row



The tidyverse package

- There are two main differences in the usage of a **data frame** vs a **tibble**: printing, and subsetting
    - **Tibbles** show only the first 10 rows, and all the columns that fit on screen. This makes it much easier to work with large dataset
    - In addition to its name, each column reports its type, a nice feature borrowed from `str()`
- Let's display the `trial01` created above

```
trial01
```

```
##     variety   yield height flowering
## 1 G01-US234 6323.30 123.30        87
## 2 G05-BT456 2515.20  95.20       101
## 3     Ind01 5611.00 113.00        88
## 4 G11-US244 7729.00  89.45       120
## 5 G11-DR234 7843.25 145.67        90
```

The tidyverse package

- Let's look at the structure of `trial01`

```
str(trial01)
```

```
## 'data.frame':	5 obs. of  4 variables:
##  $ variety  : chr  "G01-US234" "G05-BT456" "Ind01" "G11-US244" ...
##  $ yield    : num  6323 2515 5611 7729 7843
##  $ height   : num  123.3 95.2 113 89.5 145.7
##  $ flowering: num  87 101 88 120 90
```

- `trial01` is a **data frame**
- In **R**, we can convert a **data frame** object to a **tibble** object
- For that, we need the function `as_tibble` from the **tidyverse** `package`

The tidyverse package

- A **function** of a **package** is accessible only when the package is launched
- Then we can convert `trial01` to a `tibble` and save the new created object into `trial01.tibble`

```
library(tidyverse)
trial01.tibble <- as_tibble(trial01)
```

- Let's display the data in **tibble** (left) and **data frame** (right) formats

```
## # A tibble: 5 x 3
##   variety   yield height
##   <chr>     <dbl>  <dbl>
## 1 G01-US234 6323.  123. 
## 2 G05-BT456 2515.   95.2
## 3 Ind01     5611   113  
## 4 G11-US244 7729    89.4
## 5 G11-DR234 7843.  146. 
```

```
##     variety   yield height
## 1 G01-US234 6323.30 123.30
## 2 G05-BT456 2515.20  95.20
## 3     Ind01 5611.00 113.00
## 4 G11-US244 7729.00  89.45
## 5 G11-DR234 7843.25 145.67
```

Data Import

- There are different ways to import data into **R**. We choose the `readr` and `readxl` packages, part of `tidyverse`

- `readr` is part of the core `tidyverse` and supports seven file formats with seven `read_functions`

- The common `read_function` is: `read_csv()` for comma delimited files

- To use `read_csv()`, supply the path to a file and you get the data into R

```
mydata <- read_csv("C:/Documents/R-basics/mydata.csv")
```

- For path names with **R**, use forward slashes / or put two backslashes \\

- If a project is created and we are working within the project, then

```
mydata <- read_csv("mydata.csv")
```

Data Import

- The `readxl` package makes it easy to get data out of Excel into R, easy to install for all operating systems
- `readxl` supports both the **.xls** and the **.xlsx** format
- `readxl` is not a core `tidyverse` package. To be loaded explicity by `library(readxl)`
- The main function of `readxl` is `read_excel()`
- To use `read_excel()`, supply the path of the Excel file along with the sheet name

```
mydata <- read_excel("mydata.xlsx", sheet = "Sheet1")
```

Data Import: Summary

- To import a **csv** file into **R**

```
library(tidyverse)
mydata <- read_csv("mydata.csv")
```

- To import a **xlsx** file into **R**

```
library(readxl)
mydata <- read_excel("mydata.xlsx", sheet = "Sheet1")
```

Data Import: Practical

05:00 minutes

- Import **Example-01.csv** to R and save it to an object (choose an appropriate object name)
- Display the data into R
- Extract and display:
  - the two first columns
  - the three last rows
  - from row 1 to 6 with all columns

Data Transformation

- When working with data, need to:
  - create new variables,
  - make summaries,
  - rename the variables,
  - reorder the observations, etc.
- This can be achieved with the package `dplyr`, a core package of `tidyverse`

Data Transformation

- There are five key **dplyr** functions in **tidyverse** to solve most of data manipulation challenges:
    - Pick observations by their values – **filter()**
    - Reorder (sort) the rows – **arrange()**
    - Pick variables by their names – **select()**
    - Create new variables with functions of existing variables – **mutate()**
    - Collapse many values down to a summary – **summarize()**

IITA
Transforming African Agriculture

Data Transformation: Filter

- **filter()** allows to subset observations based on their values

```
filter(data, expressions)
```

- The first argument is the name of the data frame
- The second and subsequent arguments are the expressions that filter the data frame
- Let's import **Example-02.csv** to R
- First: we need to copy the **csv** file to the project working directory

Data Transformation: Filter

- Second: we need to import the file into **R**. As this is a **csv** file, let's use the **readr** package, a core package of **tidyverse** to import the data

```
library(tidyverse)
example02 <- read_csv("Example-02.csv")
example02
```

```
## # A tibble: 464 x 10
##    env   loc    year gen   yield height lodging  size protein   oil
##    <chr> <chr> <dbl> <chr> <dbl>  <dbl>   <dbl> <dbl>   <dbl> <dbl>
##  1 L70   Lawes  1970 G01    2.39  1.44     4.25  8.45    36.7  20.9
##  2 L70   Lawes  1970 G02    2.28  1.45     4.25  9.95    37.6  20.7
##  3 L70   Lawes  1970 G03    2.57  1.46     3.75 10.8     37.8  21.3
##  4 L70   Lawes  1970 G04    2.88  1.26     3.5  10.0     38.4  22.0
##  5 L70   Lawes  1970 G05    2.39  1.34     3.5  11       37.5  22.1
##  6 L70   Lawes  1970 G06    2.41  1.36     4    11.8     38.2  21.2
##  7 L70   Lawes  1970 G07    2.70  1.3      3    11.8     37.4  21.7
##  8 L70   Lawes  1970 G08    2.46  0.955    3.25 10       35.2  21.1
##  9 L70   Lawes  1970 G09    2.57  1.03     3    11.2     35.9  21.5
## 10 L70   Lawes  1970 G10    2.98  1.16     3.75 10.8     39.7  20.4
## # … with 454 more rows
```

Data Transformation: Filter

- We can filter the data for 1970

```
library(tidyverse)
example02.70 <- filter(example02, year == 1970)
example02.70
```

```
## # A tibble: 232 x 10
##    env   loc    year gen   yield height lodging  size protein   oil
##    <chr> <chr> <dbl> <chr> <dbl>  <dbl>   <dbl> <dbl>   <dbl> <dbl>
##  1 L70   Lawes  1970 G01    2.39  1.44     4.25  8.45    36.7  20.9
##  2 L70   Lawes  1970 G02    2.28  1.45     4.25  9.95    37.6  20.7
##  3 L70   Lawes  1970 G03    2.57  1.46     3.75 10.8     37.8  21.3
##  4 L70   Lawes  1970 G04    2.88  1.26     3.5  10.0     38.4  22.0
##  5 L70   Lawes  1970 G05    2.39  1.34     3.5  11       37.5  22.1
##  6 L70   Lawes  1970 G06    2.41  1.36     4    11.8     38.2  21.2
##  7 L70   Lawes  1970 G07    2.70  1.3      3    11.8     37.4  21.7
##  8 L70   Lawes  1970 G08    2.46  0.955    3.25 10       35.2  21.1
##  9 L70   Lawes  1970 G09    2.57  1.03     3    11.2     35.9  21.5
## 10 L70   Lawes  1970 G10    2.98  1.16     3.75 10.8     39.7  20.4
## # … with 222 more rows
```

Transforming African Agriculture

Data Transformation: Filter

- We can filter the ata for one location :

```
library(tidyverse)
filter(example02, loc == "Lawes")
```

```
## # A tibble: 116 x 10
##    env   loc    year gen   yield height lodging  size protein   oil
##    <chr> <chr> <dbl> <chr> <dbl>  <dbl>   <dbl> <dbl>   <dbl> <dbl>
##  1 L70   Lawes  1970 G01    2.39  1.44     4.25  8.45    36.7  20.9
##  2 L70   Lawes  1970 G02    2.28  1.45     4.25  9.95    37.6  20.7
##  3 L70   Lawes  1970 G03    2.57  1.46     3.75 10.8     37.8  21.3
##  4 L70   Lawes  1970 G04    2.88  1.26     3.5  10.0     38.4  22.0
##  5 L70   Lawes  1970 G05    2.39  1.34     3.5  11       37.5  22.1
##  6 L70   Lawes  1970 G06    2.41  1.36     4    11.8     38.2  21.2
##  7 L70   Lawes  1970 G07    2.70  1.3      3    11.8     37.4  21.7
##  8 L70   Lawes  1970 G08    2.46  0.955    3.25 10       35.2  21.1
##  9 L70   Lawes  1970 G09    2.57  1.03     3    11.2     35.9  21.5
## 10 L70   Lawes  1970 G10    2.98  1.16     3.75 10.8     39.7  20.4
## # … with 106 more rows
```

Data Transformation: Filter

- We can filter the data with multiple criteria

```
filter(example02, yield > 3.2, loc == "Lawes")
```

```
## # A tibble: 8 x 10
##   env   loc    year gen   yield height lodging  size protein   oil
##   <chr> <chr> <dbl> <chr> <dbl>  <dbl>   <dbl> <dbl>   <dbl> <dbl>
## 1 L70   Lawes  1970 G26    3.21  1.18     4     9.55    39.8  20.1
## 2 L70   Lawes  1970 G45    3.26  0.785    2.75 23.2     37    23.7
## 3 L70   Lawes  1970 G48    4.38  0.76     3.75 18.8     38.2  24.4
## 4 L70   Lawes  1970 G50    3.38  0.835    2.75 17.9     39.9  24.6
## 5 L71   Lawes  1971 G45    3.82  0.56     1.25 19.6     36.3  23.7
## 6 L71   Lawes  1971 G48    3.57  0.545    1.25 17.0     39.1  24.1
## 7 L71   Lawes  1971 G49    3.44  0.66     1.25 18.6     37.5  23.1
## 8 L71   Lawes  1971 G57    3.39  0.685    1.5  19.4     38.6  22.8
```

Data Transformation: Filter

- What does the following command do?

```
filter(example02, loc == "Lawes" | loc == "Brookstead")
```

```
## # A tibble: 232 x 10
##    env   loc    year gen   yield height lodging  size protein   oil
##    <chr> <chr> <dbl> <chr> <dbl>  <dbl>   <dbl> <dbl>   <dbl> <dbl>
##  1 L70   Lawes  1970 G01    2.39  1.44     4.25  8.45    36.7  20.9
##  2 L70   Lawes  1970 G02    2.28  1.45     4.25  9.95    37.6  20.7
##  3 L70   Lawes  1970 G03    2.57  1.46     3.75 10.8     37.8  21.3
##  4 L70   Lawes  1970 G04    2.88  1.26     3.5  10.0     38.4  22.0
##  5 L70   Lawes  1970 G05    2.39  1.34     3.5  11       37.5  22.1
##  6 L70   Lawes  1970 G06    2.41  1.36     4    11.8     38.2  21.2
##  7 L70   Lawes  1970 G07    2.70  1.3      3    11.8     37.4  21.7
##  8 L70   Lawes  1970 G08    2.46  0.955    3.25 10       35.2  21.1
##  9 L70   Lawes  1970 G09    2.57  1.03     3    11.2     35.9  21.5
## 10 L70   Lawes  1970 G10    2.98  1.16     3.75 10.8     39.7  20.4
## # … with 222 more rows
```

Transforming African Agriculture

Data Transformation: Practical

08:00 minutes

- Import **Example-02.csv** to R and save it to an object named **example02**
- Display the **example02** object
- filter the data by considering:
  - locations `Nambour` and `RedlandBay`
  - genotypes `G01`, `G57`, and `G58`, location `Brookstead` for the year `1970`
  - location `Lawes` yield between `2` and `3` inclusive, oil greater than `22`

Data Transformation: Arrange

- **arrange()** changes the order of the rows (sorting): it takes a data frame and a set of column names (or more complicated expressions) to order by

- We can arrange `example02` by year, loc, gen

```
arrange(example02, year, loc, gen)
```

```
## # A tibble: 464 x 10
##    env   loc          year gen   yield height lodging  size protein   oil
##    <chr> <chr>       <dbl> <chr> <dbl>  <dbl>   <dbl> <dbl>   <dbl> <dbl>
##  1 B70   Brookstead  1970 G01   1.25    1.01    3.25  8.85    39.5  18.8
##  2 B70   Brookstead  1970 G02   1.17    1.13    2.75  8.9     38.6  19.8
##  3 B70   Brookstead  1970 G03   0.468   1.16    2.25 10.8     37.8  20.4
##  4 B70   Brookstead  1970 G04   1.44    1.24    1.5  10.6     38.7  20.4
##  5 B70   Brookstead  1970 G05   1.34    1.12    2    12.0     37.8  20.8
##  6 B70   Brookstead  1970 G06   0.913   1.10    2.25 11       37.4  19.9
##  7 B70   Brookstead  1970 G07   1.24    1.13    2    10.2     37.8  20.3
##  8 B70   Brookstead  1970 G08   0.385   1.12    2.25  6.15    38.5  17.9
##  9 B70   Brookstead  1970 G09   1.11    1.04    1.75  8.3     37.9  20.0
## 10 B70   Brookstead  1970 G10   1.80    1.04    2    11.8     38.4  19.7
## # … with 454 more rows
```

Data Transformation: Arrange

- We can use the function **desc()** to re-order by a column in descending order

```
arrange(example02, desc(yield))
```

```
## # A tibble: 464 x 10
##    env   loc         year gen   yield height lodging  size protein   oil
##    <chr> <chr>      <dbl> <chr> <dbl>  <dbl>   <dbl> <dbl>   <dbl> <dbl>
##  1 L70   Lawes       1970 G48    4.38 0.76      3.75  18.8    38.2  24.4
##  2 R70   RedlandBay  1970 G56    4.13 0.56      2.25  19.0    38    24.0
##  3 R71   RedlandBay  1971 G49    4.00 0.905     1.75  17.2    36.6  22.6
##  4 B71   Brookstead  1971 G49    3.90 1.00      2.25  21.6    37.5  22.2
##  5 L71   Lawes       1971 G45    3.82 0.56      1.25  19.6    36.3  23.7
##  6 B71   Brookstead  1971 G53    3.82 0.675     1.75  21.9    39.0  22.1
##  7 B71   Brookstead  1971 G45    3.75 0.735     2.5   23.6    37.9  22.1
##  8 R70   RedlandBay  1970 G57    3.67 0.545     1.75  16.6    38.2  23.7
##  9 R70   RedlandBay  1970 G49    3.62 1.02      3.5   14.2    36.3  23.4
## 10 B71   Brookstead  1971 G50    3.61 0.75      3     19.3    40.2  23.3
## # … with 454 more rows
```

IITA
Transforming African Agriculture

Data Transformation: Arrange

- When working with many variables, it can be a good practice to narrow the dataset and consider only few variables for analysis. Let's only consider the location, year, genotype, yield and height

```
example02.short <- select(example02, loc, year, gen, yield, height)
example02.short
```

```
## # A tibble: 464 x 5
##    loc    year gen   yield height
##    <chr> <dbl> <chr> <dbl>  <dbl>
##  1 Lawes  1970 G01    2.39  1.44
##  2 Lawes  1970 G02    2.28  1.45
##  3 Lawes  1970 G03    2.57  1.46
##  4 Lawes  1970 G04    2.88  1.26
##  5 Lawes  1970 G05    2.39  1.34
##  6 Lawes  1970 G06    2.41  1.36
##  7 Lawes  1970 G07    2.70  1.3
##  8 Lawes  1970 G08    2.46  0.955
##  9 Lawes  1970 G09    2.57  1.03
## 10 Lawes  1970 G10    2.98  1.16
## # … with 454 more rows
```

Data Transformation: Arrange

- We can be interested to move one or more variables to the start of the data frame. For that, we can use **select()** and **everything()**

```
select(example02, year, everything())
```

```
## # A tibble: 464 x 10
##     year env   loc   gen   yield height lodging  size protein   oil
##    <dbl> <chr> <chr> <chr> <dbl>  <dbl>   <dbl> <dbl>   <dbl> <dbl>
##  1  1970 L70   Lawes G01    2.39  1.44     4.25  8.45    36.7  20.9
##  2  1970 L70   Lawes G02    2.28  1.45     4.25  9.95    37.6  20.7
##  3  1970 L70   Lawes G03    2.57  1.46     3.75 10.8     37.8  21.3
##  4  1970 L70   Lawes G04    2.88  1.26     3.5  10.0     38.4  22.0
##  5  1970 L70   Lawes G05    2.39  1.34     3.5  11       37.5  22.1
##  6  1970 L70   Lawes G06    2.41  1.36     4    11.8     38.2  21.2
##  7  1970 L70   Lawes G07    2.70  1.3      3    11.8     37.4  21.7
##  8  1970 L70   Lawes G08    2.46  0.955    3.25 10       35.2  21.1
##  9  1970 L70   Lawes G09    2.57  1.03     3    11.2     35.9  21.5
## 10  1970 L70   Lawes G10    2.98  1.16     3.75 10.8     39.7  20.4
## # … with 454 more rows
```

Data Transformation: Add new variables

- We can add new columns that are functions of existing columns with `mutate()` which always adds new columns at the end of the dataset

```
mutate(example02.short, yield_kg_ha = yield * 1000)
```

```
## # A tibble: 464 x 6
##    loc    year gen   yield height yield_kg_ha
##    <chr> <dbl> <chr> <dbl>  <dbl>       <dbl>
##  1 Lawes  1970 G01    2.39 1.44         2387
##  2 Lawes  1970 G02    2.28 1.45         2282
##  3 Lawes  1970 G03    2.57 1.46         2567
##  4 Lawes  1970 G04    2.88 1.26         2877
##  5 Lawes  1970 G05    2.39 1.34         2392
##  6 Lawes  1970 G06    2.41 1.36         2408
##  7 Lawes  1970 G07    2.70 1.3          2699
##  8 Lawes  1970 G08    2.46 0.955        2457
##  9 Lawes  1970 G09    2.57 1.03         2567
## 10 Lawes  1970 G10    2.98 1.16         2984
## # … with 454 more rows
```

Transforming African Agriculture

Data Transformation: Summaries

- `summarize()` collapses a data frame to a single or few row(s)

```
summarize(example02, yield_all = mean(yield, na.rm = TRUE))
```

```
## # A tibble: 1 x 1
##   yield_all
##       <dbl>
## 1      2.05
```

- `summarize()` is useful with `group_by()`

```
by_treatment <- group_by(example02, loc)
summarise(by_treatment, yield_treatment = mean(yield, na.rm = TRUE))
```

```
## # A tibble: 4 x 2
##   loc        yield_treatment
##   <chr>                <dbl>
## 1 Brookstead            2.01
## 2 Lawes                 2.37
## 3 Nambour               2.09
## 4 RedlandBay            1.72
```

Data Transformation: The pipe %>%

- The following code:

```r
by_treatment <- group_by(example02, loc)
summarise(by_treatment, yield_treatment = mean(yield, na.rm = TRUE))
```

- is equivalent to:

```r
example02 %>%
    group_by(loc) %>%
    summarise(yield_treatment = mean(yield, na.rm = TRUE))
```

IITA
Transforming African Agriculture

Data Transformation: Count

- When doing aggregation, it's a good idea to include a count
- It might be useful before analyzing our data to check how many datapoints we have for each location

```
example02 %>%
    group_by(loc) %>%
    summarise(n = n())
```

```
## # A tibble: 4 x 2
##   loc            n
##   <chr>      <int>
## 1 Brookstead   116
## 2 Lawes        116
## 3 Nambour      116
## 4 RedlandBay   116
```

Data Transformation: Factors

- Factors are used to work with categorical variables, with a fixed and known set of possible values. The values of a factor are called the levels

- Let's display `example02`: `env`, `loc` and `gen` are **character** but should be considered categorical variables (**factors**)

```
example02
```

```
## # A tibble: 464 x 10
##    env   loc    year gen   yield height lodging  size protein   oil
##    <chr> <chr> <dbl> <chr> <dbl>  <dbl>   <dbl> <dbl>   <dbl> <dbl>
##  1 L70   Lawes  1970 G01    2.39  1.44     4.25  8.45    36.7  20.9
##  2 L70   Lawes  1970 G02    2.28  1.45     4.25  9.95    37.6  20.7
##  3 L70   Lawes  1970 G03    2.57  1.46     3.75 10.8     37.8  21.3
##  4 L70   Lawes  1970 G04    2.88  1.26     3.5  10.0     38.4  22.0
##  5 L70   Lawes  1970 G05    2.39  1.34     3.5  11       37.5  22.1
##  6 L70   Lawes  1970 G06    2.41  1.36     4    11.8     38.2  21.2
##  7 L70   Lawes  1970 G07    2.70  1.3      3    11.8     37.4  21.7
##  8 L70   Lawes  1970 G08    2.46  0.955    3.25 10       35.2  21.1
##  9 L70   Lawes  1970 G09    2.57  1.03     3    11.2     35.9  21.5
## 10 L70   Lawes  1970 G10    2.98  1.16     3.75 10.8     39.7  20.4
## # … with 454 more rows
```

Data Transformation: Factors

- The function `as.factor()` convert a variable to a **factor**

```
example02$env <- as.factor(example02$env)
example02$loc <- as.factor(example02$loc)
example02$gen <- as.factor(example02$gen)
```

- This is equivalent to the code below using `mutate()` and the pipe `%>%`

```
example02 %>%
    mutate(env = factor(env), loc = factor(loc), gen = factor(gen))
```

- or the same but in more clean code (my preference)

```
example02 %>%
  mutate(
    env=factor(env),
    loc=factor(loc),
    gen=factor(gen)
  )
```

Data Transformation: Factors

- To display the levels of a factor

```
levels(example02$loc)
```

```
## [1] "Brookstead" "Lawes"      "Nambour"    "RedlandBay"
```

- To get the number of levels of a factor

```
nlevels(example02$loc)
```

```
## [1] 4
```

Data Wrangling: Practical

30:00 minutes

- Import **Example-03.xlsx** to R and save to an object named **example03**
- How many observations do we have per location?
- What are the variables? What are the factors? -- convert them to factors if any
- How many locations? genotypes? Display the list of locations and the list of genotypes respectively?
- Display the data where the yield was less than `150`?
- Display the mean of `earht` per genotype, sorted in descending order
- Select `loc`, `gen`, `yield`, `flower` to be saved in a new object `example03.short` and create a new variable `flower_new` by subtracting 10 days to `flower`

Data Wrangling: Practical

30:00 minutes

- Import **Example-02.csv** to R and save to an object named **example02**
- How many locations do we have per `year`?
- Calculate the number of observations, min, max, mean, variance and standard deviation of `size` by `location` and `year`
- Select all the observations where `oil` is greater than 20, `lodging` less than 3, and `yield` greater than 3 in the location `Brookstead` and sort the height by descending order