Session 2: Building our analysis dataset

Turry Ouma, Meklit Chernet

Let's EXPLORE our data:

- The function str() is one of the functions you will need to apply before you work on any new dataset. It gives you more insight into the structure of your dataset. Type str(dataVAL_FR_PO) in the console and see the result. You will get the total number of variables, observations, data type of each variable, the first observations etc.
- The function head() allows us to see the first 6 rows by default. You can also specify the number of rows: head(dataVAL_FR_PO, 10)
- The function summary() shows the data type of each variable and a few other attributes that are useful for numeric attributes. It also displays min, 1st quartile, median, mean, 3rd quartile and max values: summary(dataVAL_FR_PO)

R’s basic data types are:

- Decimal values are called numerics in R
- Whole numbers like are called integers. Integers are also numerics.
- Boolean values e.g TRUE or FALSE are called logical.
- Text/string values are called characters.

Selection of data frame elements

- Since we have specific parameters of interest for this exercise (Identifier information, location data, selected agronomic events and dates & yield data), let's proceed to subset our data. e.g.
- Now if we knew the exact column number it would be easier to use square brackets in base R, as in example 1 below:

Example 1: Selecting Columns by Index

```
dataVAL_FR_PO [ ,1:8] #we use the sequence to select all the metadata columns of our dataset
dataVAL_FR_PO [2,2] #to select the value at the second row and second column
dataVAL_FR_PO [1:2,6:10] #to select rows 1, 2 and columns 6,7,8,9,10
```

OR using dplyr:

```
dataVAL_FR_PO %>%
  select(c(2, 5, 6))
```

Example 2: Selecting Specific Columns by their Names

- But often times and as you have seen when exploring this dataset, we have numerous columns so it is hard to select by row/column. In our case, it is better to select the columns by name.
- We use names(dataVAL_FR_PO) to identify the names of the columns, then put all the column names in a vector:

```
#subset/select
dataVAL_FR_PO [, c("ENID", "HHID", "geopoint-Latitude","geopoint-Longitude",
"plantingDetails/plantingDate", "plantingDetails/variety", "harvest/intHarvestDate_CON",
"harvest/effHarvestDate_CON", "harvest/tuberizedMarketableRootsFW_CON",
harvest/intHarvestDate_SSR", "harvest/effHarvestDate_SSR",
harvest/tuberizedMarketableRootsFW_SSR")]
```

- #OR use dplyr

```
yield_datFR <- dataVAL_FR_PO %>%
                select(c("ENID", "HHID", "geopoint-Latitude","geopoint-Longitude",
"plantingDetails/plantingDate", "plantingDetails/variety",
"harvest/intHarvestDate_CON","harvest/effHarvestDate_CON",
"harvest/tuberizedMarketableRootsFW_CON", "harvest/intHarvestDate_SSR",
"harvest/effHarvestDate_SSR", "harvest/tuberizedMarketableRootsFW_SSR"))
```

Assigning new field names

- As you may have noticed, the variable names are quite lengthy and may be hard to remember. So, we are going to change that by assigning new names using the function colnames()

```
colnames(yield_datFR) = c("ENID", "HHID", "Latitude", "Longitude",
"plantingDate", "variety",  "intHarvestDate_CON", "HarvestDate_CON",
"yield_CON","intHarvestDate_SSR", "HarvestDate_SSR","yield_SSR")
```

today	ENID	HHID	country	geopoint-Latitude	geopoint-Longitude	harvest/intHarvestDate_CON	harvest/effHarvestDate_CON	harvest/intHarvestDate_SSR	h
24-May-18	ACEANG000240	ACHHNG002213	NG	7.154756667	5.600325				
24-May-18	ACEANG000240	ACHHNG002212	NG	7.149225	5.589766667				
11-May-18	ACEANG000162	ACHHNG002210	NG	7.264956667	5.219845				
24-May-18	ACEANG000162	ACHHNG002210	NG	6.769763333	3.906036667	3/10/2019		1/10/2019	
24-May-18	ACEANG000162	ACHHNG002210	NG	6.769726667	3.906088333				
24-May-18	ACEANG000162	ACHHNG002201	NG	6.768298333	3.904696667				

Repetitive coordinates

When you explore this data in order to get unique identifiers, you will realize that in some cases there were several records from the same HH because the gps coordinates were captured multiple times, probably due to poor visibility.

In order to clean this data, we round off the gps coordinates to drop repetitive ones:

```
yield_datFR$Latitude <- round(yield_datFR$Latitude,digits=3)#drop some repetitive gps coordinates
yield_datFR$Longitude <- round(yield_datFR$Longitude,digits=3)
```

No yield data

- Since our main concern is with data that has yield data, we now proceed to drop any records that have no yield.
- Here we apply the ifelse statement:

```
yield_datFR$yieldInfo <- ifelse( is.na(yield_datFR$plantingDate) &
is.na(yield_datFR$variety)  & is.na(yield_datFR$HarvestDate_SSR)  &
is.na(yield_datFR$HarvestDate_CON)  & is.na(yield_datFR$yield_CON)  &
is.na(yield_datFR$yield_SSR), "NO YIELD", "YIELD") #drop rows without any yield
information
yield_datFR2 <- droplevels(yield_datFR[!yield_datFR$ yieldInfo == "NO YIELD", ])
```