Session 3: Data Wrangling

Turry Ouma, Meklit Chernet

Working with dates

The more you learn about dates and times, the more complicated they seem to get. To warm up, try these three seemingly simple questions:

Does every year have 365 days?

Does every day have 24 hours?

Does every minute have 60 seconds?

Explore the structure of dfFR, specifically the ‘harvest date’ variable for SSR and Control treatments.

Discuss how the issue (if any) can be best resolved

Working with dates

- The ‘harvest date’ is given as a character type when in reality it takes in data in the date format. In order to correct this, we are going to combine two lines of functions as below:
- We will use the **lubridate package** to change the type and then get the difference between harvest dates and planting dates by creating a new variable using **mutate**.

```
The order of our dates is this: 10-Jun-19 so dmy suffices.
dfFR <- dfFR %>% mutate(
  diffeffH = lubridate::dmy(dfFR$HarvestDate_SSR) - lubridate::dmy(dfFR$plantingDate),
  diffeffC = lubridate::dmy(dfFR$HarvestDate_CON) - lubridate::dmy(dfFR$plantingDate)
  )
```

Merging data

- For the next step in getting graphics out of this data, we need to know from which region the data came from. Since this was not in our original questionnaire, we are lucky that we have a separate registration dataset that has this information linked to the HHID and ENID. We use the merge function to get this information:
- dfFR2 <- droplevels(unique(merge(dsENHH, dfFR, by=c("HHID", "ENID"))))

Discuss how to join data frames (inner, outer, left, right) in R

Dates AGAIN! ... and FOR loops

#For loops
For Loop is a type of control statement used to iterate over items of a sequence.
For loop Syntax:

```
for (value in sequence)
{
  statement
}
```

Discuss instances in which the **For loop** can be applied

- In order to generate bar graphs of yield by month we need to convert the dates into month abbreviations. We achieve this by splitting the dates to get the month number then we use the function month.abb (an inbuilt R constant) to convert this number into the month abbreviation:

```
#work on dates to get m, d, y


        onpldate <- NULL
        for(hids in unique(dfFR2$HHID)){
          hdata <- dfFR2[dfFR2$HHID == hids, ]
          hdata$pmnth <- lubridate::month(dmy(hdata$plantingDate))
          hdata$pday <- lubridate::day(dmy(hdata$plantingDate))
          hdata$pyear <- lubridate::year(dmy(hdata$plantingDate))
          hdata$rtdmy <- ifelse(is.na(hdata$pmnth), NA, paste(hdata$pyear, hdata$pmnth, hdata$pday, sep = "/"))
          hdata$rtdmy <-  as.Date(hdata$rtdmy)
          hdata <- hdata[order(hdata$rtdmy, decreasing = TRUE), ][1,]
          onpldate <- rbind(onpldate, hdata)
        }


head(onpldate)
str(onpldate)
onpldate$f0m <- month.abb[onpldate$pmnth]
```

Reshaping your data frame

- Often times you will want to reshape your data frame from wide to 'longer' layouts or vice versa. In our case for plotting we need to have a column that gives us the yield (by treatment type) alongside the value in a separate column. Therefore, since our data is in wide format, we use the tidyr function "gather" to reshape our data into long format, favorable for plotting.

```
#change the layout of data
#gather
pldate <- onpldate %>% tidyr::gather(harvest, yield, yield_SSR, yield_CON)
pldate$harvest <- as.character(pldate$harvest)
pldate$harvest[pldate$harvest == "yield_SSR"] <- "SSR"
pldate$harvest[pldate$harvest == "yield_CON"] <- "Control"
hvstpldate <- pldate %>% tidyr::gather(diff, days, diffeffH, diffeffC)
head(hvstpldate)

hvstpldate2 <- hvstpldate %>% tidyr::gather(hvstype, daterc, HarvestDate_SSR, HarvestDate_CON)
head(hvstpldate2)
```

Discuss how to use the tidyr function "spread"

Factors & levels

- If you asses the regions/states where the data was collected, you will see that these are abbreviated. In order to plot by region, we need to provide correct full names.
- When you check str(rti) you see that region_state is listed as character. In order to rename the levels, we have to convert this to factor and then recode the factor levels:

```
rti$region3 <- as.factor(rti$region_state)
levels(rti$region_state) #asses the levels
levels(rti$region_state) <- c("Kwara", "Ogun", "Ondo",
"Oyoo") #rename the levels
```