Day 2

Oyedele Oluwafemi

14/11/2021

Data Types in R

1 Vector A vector is a basic data structure in R. It contains elemnet of the same type. It is further classified into 6 data types which is called atomic vectors. The names of the six atomic vectors are as follows. - Numeric - Integer - Logical - Character - Complex - Raw

let us create a simple vector

N/B We can create a vector with the c(), function where the c stands for combine or concatenate. Since a vector must have elements of the same type, this function will try and coerce elements to the same type, if they are different.

```
vec1<-c(1,2,5) # This is a numeric vector
class(vec1)
```

```
## [1] "numeric"
```

```
typeof(vec1)
```

```
## [1] "double"
```

```
class(vec1)
```

```
## [1] "numeric"
```

Let us create a character vector

```
vec2<-c('Black','Yellow','White')
class(vec2)
```

```
## [1] "character"
```

```
typeof(vec2)
```

```
## [1] "character"
```

```
length(vec2)
```

```
## [1] 3
```

```
nchar(vec2)
```

```
## [1] 5 6 5
```

2. List List is a data structure having components of mixed data types. i.e a vector having elements of different type is called a list. # Let us create a list

```
list1<-list(1,'Blue',2.4,'Green')
typeof(list1)
```

```
## [1] "list"
```

```
class(list1)
```

```
## [1] "list"
```

```
length(list1)
```

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

3. Matrix Matrix is a two dimentional data structure in R programming. Matrix is similar to vector but additionally contains the dimension attribute.

```
mat1<-matrix(1:9,nrow = 3,ncol = 3,byrow = TRUE)
class(mat1)
```

```
## [1] "matrix" "array"
```

```
typeof(mat1)
```

```
## [1] "integer"
```

```
attributes(mat1)
```

```
## $dim
## [1] 3 3
```

4. Factors Factors is a data structure used for fields that takes only predefined, finite number of values (categorical data). e.g male and female. Factors is normally created with the factor function in R.

```
fact1<-factor('Male','Female')
class(fact1)
```

```
## [1] "factor"
```

5. Data frane Data frame is a two dimensional data structure in R. It is a special case of a list which has each component of equal length. Each component form the column and the contents of the component form the rows.

```
df1<-data.frame(Age=c(10,15,20,30),Month=c('Jan','Feb','Mar','Apr'))
class(df1)
```

```
## [1] "data.frame"
```

```
typeof(df1)
```

```
## [1] "list"
```

```
head(df1)
```

```
##   Age Month
## 1  10   Jan
## 2  15   Feb
## 3  20   Mar
## 4  30   Apr
```

```
tail(df1)
```

```
##   Age Month
## 1  10   Jan
## 2  15   Feb
## 3  20   Mar
## 4  30   Apr
```

Setting of Working Directory

Once you open R, your present working directory is always your root directory that you select when you where installing R. - To get your present working directory # Type the code below getwd()

Reading Data into R

- To read in a csv file into R we will use read_csv # tidyverse
- To read in an excel file into R we will use read_excel # tidyverse

```
library(tidyverse)
```

```
## Warning in as.POSIXlt.POSIXct(Sys.time()): unable to identify current timezone 'U':
## please set environment variable 'TZ'

## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.0.2     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(here)
```

```
## here() starts at C:/Users/OLUWAFEMI/Desktop/IARSAF_R_Basic_Training
```

```
library(readxl)
library(skimr)
```

```
dat1<-read_csv(here::here('data/sorghum.csv'))
```

```
## New names:
## * `` -> ...1

## Rows: 289 Columns: 7

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (4): gen, trial, env, loc
## dbl (3): ...1, yield, year

##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# Congratulation you have successfully read a csv file into R
# Now assign it to an object called dat1
```

```
dat2<-read_excel(here::here('data/oats.xlsx'))
```

Let use some functions from tidyverse to explore our data

```
dat1 %>% slice_head(n = 5)
```

```
## # A tibble: 5 x 7
##    ...1 gen   trial env   yield  year loc
##   <dbl> <chr> <chr> <chr> <dbl> <dbl> <chr>
## 1     1 G16   T2    E01     590  2001 Mieso
## 2     2 G17   T2    E01     554  2001 Mieso
## 3     3 G18   T2    E01     586  2001 Mieso
## 4     4 G19   T2    E01     738  2001 Mieso
## 5     5 G20   T2    E01     489  2001 Mieso
```

```
dat1 %>% slice_tail(n=5)
```

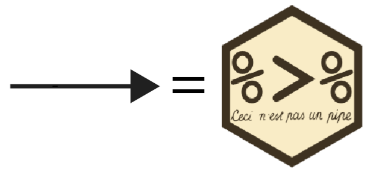
```
## # A tibble: 5 x 7
##    ...1 gen   trial env   yield  year loc
##   <dbl> <chr> <chr> <chr> <dbl> <dbl> <chr>
## 1   285 G24   T2    E13    1269  2005 Melkassa
## 2   286 G25   T2    E13    1689  2005 Melkassa
## 3   287 G26   T2    E13    1578  2005 Melkassa
## 4   288 G27   T2    E13    2038  2005 Melkassa
## 5   289 G28   T2    E13    1967  2005 Melkassa
```

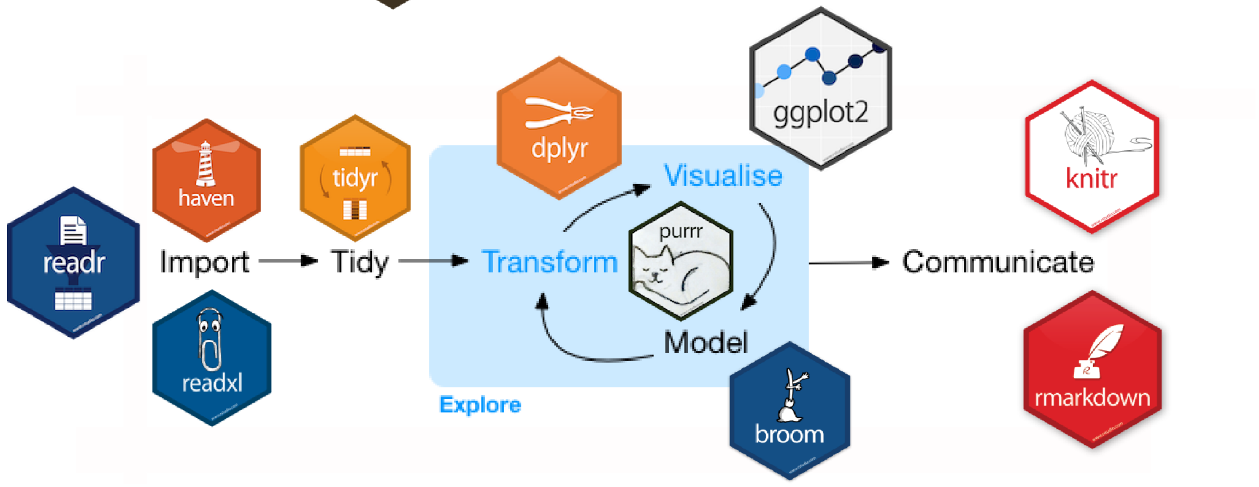
Data Cleaning using Tidyr

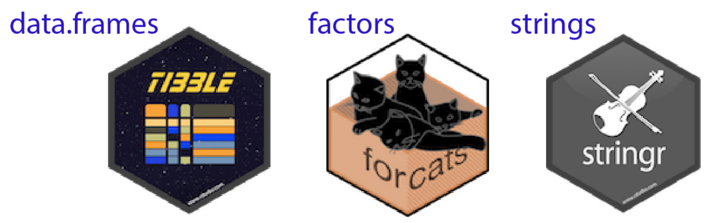
- The **tidyverse** is a collection of R packages which share a design philosophy, syntax, and data structures in which tidyr is one of them. One of the powerful function in the tidyverse is the pipes (%>%).

Pipes take the object on the **left** and apply the function on the **right**: Read out loud: "and then…" Pipes save us typing, make code readable, and allow chaining function together, so we are going to use them **all the time** when manipulating data frames.

Data Science Workflow using the Tidyverse







To install the tidyverse

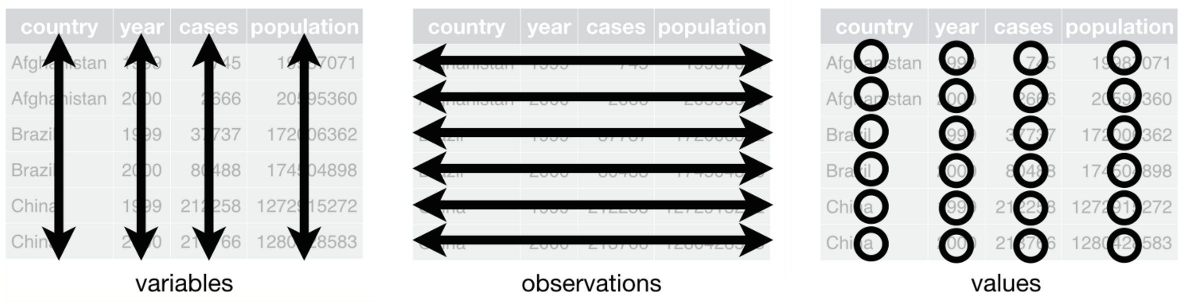
```
install.packages(tidyverse) # But you have already install this taht is why I set eval=FALSE in the cod
```

To load the tidyverse

```
library(tidyverse) # To load the tidyverse
```

Here we are going to use function from tidyr

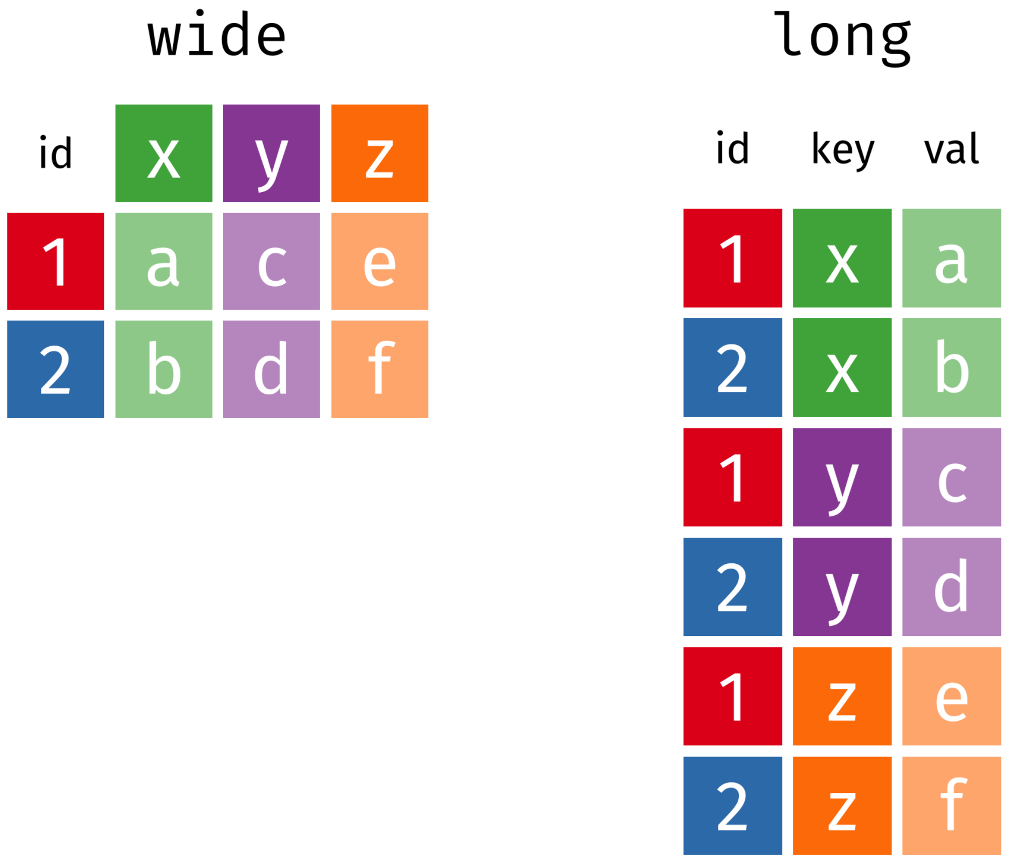
- The goal of tidyr is to help you create tidy data. Tidy data is data where:



Tidy data describes a standard way of storing data that is used wherever possible throughout the tidyverse. If you ensure that your data is tidy, you'll spend less time fighting with the tools and more time working on your analysis. Learn more about tidy data in **vignette("tidy-data")**

We are going to make use of only four function from the tidyr package

- Pivot longer
- Pivot wider
- Seperate
- Unite



Let see an example

```r
data("billboard") # this data is from tidyr package and we are going to wrangle this data and produce i
billboard %>% slice_head(n=5) # What do you observe about this dataset ?
```

```
## # A tibble: 5 x 79
##   artist   track    date.entered   wk1   wk2   wk3   wk4   wk5   wk6   wk7   wk8
##   <chr>    <chr>    <date>       <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 2 Pac    Baby Do~ 2000-02-26      87    82    72    77    87    94    99    NA
## 2 2Ge+her  The Har~ 2000-09-02      91    87    92    NA    NA    NA    NA    NA
## 3 3 Doors~ Krypton~ 2000-04-08      81    70    68    67    66    57    54    53
## 4 3 Doors~ Loser    2000-10-21      76    76    72    69    67    65    55    59
## 5 504 Boyz Wobble ~ 2000-04-15      57    34    25    17    17    31    36    49
## # ... with 68 more variables: wk9 <dbl>, wk10 <dbl>, wk11 <dbl>, wk12 <dbl>,
## #   wk13 <dbl>, wk14 <dbl>, wk15 <dbl>, wk16 <dbl>, wk17 <dbl>, wk18 <dbl>,
## #   wk19 <dbl>, wk20 <dbl>, wk21 <dbl>, wk22 <dbl>, wk23 <dbl>, wk24 <dbl>,
## #   wk25 <dbl>, wk26 <dbl>, wk27 <dbl>, wk28 <dbl>, wk29 <dbl>, wk30 <dbl>,
## #   wk31 <dbl>, wk32 <dbl>, wk33 <dbl>, wk34 <dbl>, wk35 <dbl>, wk36 <dbl>,
## #   wk37 <dbl>, wk38 <dbl>, wk39 <dbl>, wk40 <dbl>, wk41 <dbl>, wk42 <dbl>,
## #   wk43 <dbl>, wk44 <dbl>, wk45 <dbl>, wk46 <dbl>, wk47 <dbl>, wk48 <dbl>, ...
```

```r
# is it a tidydata ?
```

Let us use the pivot longer function to clean this data

```r
bill<-billboard %>% pivot_longer(-c(1:3),names_to = "Weeks",values_to = 'Record') # Assign this to an o
```

Let us seperate the date.entered into ymd

```r
bill2<-bill %>% separate(col = 'date.entered',into = c('Year','Month','Day'),sep = '-')
```

Let return the dataset into a wide format

```r
bill3<-bill %>% pivot_wider(names_from = 'Weeks',values_from = 'Record')
```

Let us unite bill 2 data frame back

```r
bill4<-bill2 %>% unite(col = 'Date', c('Year','Month','Day'))
```

You can practice on this functions by using it to solve real world question that is how you improve your data skills in R.