DDC ONLINE TRAINING SESSION:

DATA REPRESENTATION / EXPLORING DATA

DATA MANIPULATION

16/02/2023

CBS COPENHAGEN BUSINESS SCHOOL
HANDELSHØJSKOLEN

Loading the datasets

1. Load the library:

```
library(dplyr)
```

2. Load the GDP .csv file and visualize the first columns of the dataset:

```
GDP<-read.csv("GDP.csv")
head(GDP)
```

3. Load the continent .csv file and visualize the first columns of the dataset:

```
Continents<-read.csv("continent.csv")
head(Continents)
```

Combine two datasets

There are two ways to combine two datasets, the first one is by appending rows, while the other one by appending columns.

1. TASK 1: Append vertically the first three rows to the ones in position 231, 232, 233

```
rbind(df1[starting_row:ending_row , starting_column:ending column],
      df2[starting_row:ending_row , starting_column:ending column])
```

2. TASK 2: Append horizontally the first three columns to the ones in position 231, 232, 233

```
cbind(df1[starting_row:ending_row , starting_column:ending column],
      df2[starting_row:ending_row , starting_column:ending column])
```

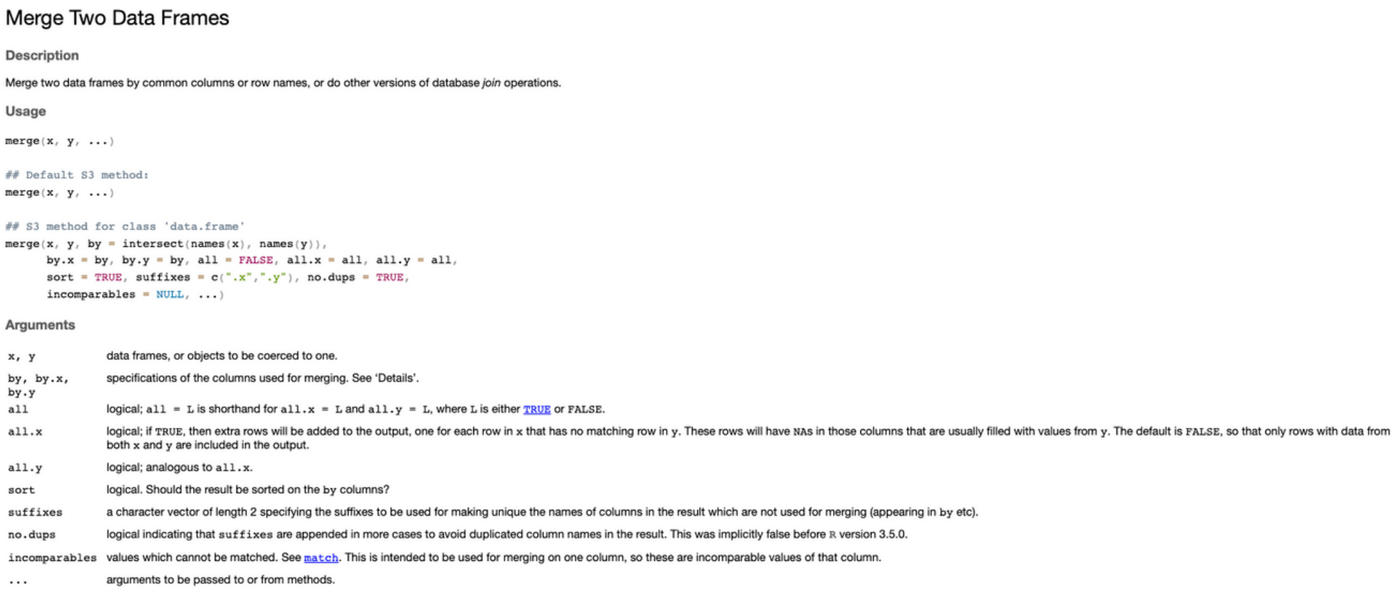
1. TASK 1: Append vertically the first three rows to the ones in position 231, 232, 233
2. TASK 2: Append horizontally the first three columns to the ones in position 231, 232, 233

```
# Append one dataset to another
# Append rows
rbind(Continents[1:3,] ,Continents[231:233,])
```

```
# Append columns
cbind(Continents[1:3,] ,Continents[231:233,])
```

merge()

Use the help() command to understand the syntax of the merge() function
[HINT: help(merge)]



Merge Two Data Frames

Description

Merge two data frames by common columns or row names, or do other versions of database *join* operations.

Usage

```
merge(x, y, ...)

## Default S3 method:
merge(x, y, ...)

## S3 method for class 'data.frame'
merge(x, y, by = intersect(names(x), names(y)),
      by.x = by, by.y = by, all = FALSE, all.x = all, all.y = all,
      sort = TRUE, suffixes = c(".x",".y"), no.dups = TRUE,
      incomparables = NULL, ...)
```

Arguments

x, y	data frames, or objects to be coerced to one.
by, by.x, by.y	specifications of the columns used for merging. See 'Details'.
all	logical; all = L is shorthand for all.x = L and all.y = L, where L is either TRUE or FALSE.
all.x	logical; if TRUE, then extra rows will be added to the output, one for each row in x that has no matching row in y. These rows will have NAs in those columns that are usually filled with values from y. The default is FALSE, so that only rows with data from both x and y are included in the output.
all.y	logical; analogous to all.x.
sort	logical. Should the result be sorted on the by columns?
suffixes	a character vector of length 2 specifying the suffixes to be used for making unique the names of columns in the result which are not used for merging (appearing in by etc).
no.dups	logical indicating that suffixes are appended in more cases to avoid duplicated column names in the result. This was implicitly false before R version 3.5.0.
incomparables	values which cannot be matched. See match. This is intended to be used for merging on one column, so these are incomparable values of that column.
...	arguments to be passed to or from methods.

TASK 3: merge the Continents and GDP datasets and display the first rows

merge()

TASK 3 SOLVED: merge the Continents and GDP datasets

```
cData <- merge(Continents, GDP)
head(cData)
```

TASK 4: merge the Continents and GDP datasets by merging on coloumn name “ISO2"

merge()

TASK 4 SOLVED: merge the Continents and GDP datasets by merging on coloumn name “ISO2"

```
merge(Continents, GDP, by = "ISO2")
```

Subset Rows Based on Column Values

From now on, use the cData dataset

TASK 5: select observations in Oceania

SOLUTION 1:

```
# displaying the first 6 columns to conserve space
# !is.na bit is required due to how R matches the == with NA's
cData[cData$Continent == "OC" & !is.na(cData$Continent),1:6]
```

SOLUTION 2 with subset():

```
subset(cData[,1:6], Continent == "OC")
```

SOLUTION 3 using filter() from the dplyr package:

```
filter(cData[,1:6], Continent == "OC")
```

Subset Rows Based on Column Values

Filtering based on multiple columns

TASK 6: select countries in Oceania that are rich (GDP greater than 3rd quartile

SOLUTION 1:

```
cData[cData$Continent == "OC" & !is.na(cData$Continent) & cData$X2011 > 23000 & !is.na(cData$X2011),]
```

SOLUTION 2 with filter():

```
filter(cData, Continent == "OC" & X2011 > 23000)
```

Selecting Certain Columns

Let us say we are interested only in the GDP figures and not in any of the country identifiers. We would want to select only certain columns.

Traditional R ways:

```
# Limiting number of rows to 3 to conserve space
cData[1:3,4:13] # Indexing by column numbers
```

With negative indexing:

```
cData[1:3,-(1:3)] # Negative indexing
```

Another way based on partial matching column name:

```
cData[1:3,grep("X", colnames(cData))]
```

Selecting Certain Columns

subset can also handle this:

```
subset(cData[1:3,], select = -c(ISO2, Continent, Country)) # Drop these columns
```

dplyr first way:

```
select(cData[1:3,], X2003:X2012) # All columns between X2003 and X2012
```

Another way with dplyr:

```
select(cData[1:3,], -(ISO2:Country))
```

Aggregating based on Groups

Let us say we want to calculate the average GDP per continent in 2011 and the number of countries in each continent.

R way:

```
Cont1 <- aggregate(cData$X2011 ~ cData$Continent, FUN=function(x)mean(x, na.rm=T))
Cont1
```

```
Cont2 <- aggregate(cData$X2011 ~ cData$Continent, FUN=function(x) length(x))
Cont2
```

```
Cont <- merge(Cont1, Cont2, by='cData$Continent')
Cont
```

```
rm(Cont1, Cont2)
```

Aggregating based on Groups

dplyr way:

```
# Create grouped data
contiData <- group_by(cData, Continent)
contiData
```

```
# Create variables on the fly
summarise(contiData, count=n(), GDP2012 = mean(X2012, na.rm = T))
```

Save dataset for later use

```
write.csv(contiData, file="contiData.csv", row.names=F)
```