Welcome to The Carpentries Etherpad!

This pad is synchronized as you type, so that everyone viewing this page sees the same text. This allows you to collaborate seamlessly on documents.

Use of this service is restricted to members of The Carpentries community; this is not for general purpose use (for that, try https://etherpad.wikimedia.org).

Users are expected to follow our **code of conduct**:
https://docs.carpentries.org/topic_folders/policies/code-of-conduct.html

---

Workshop Website: https://edcarp.github.io/2020-07-28-heriotwatt-dc-online/

Setup page: https://datacarpentry.org/spreadsheets-socialsci/setup.html

Zoom details: see email (or get in touch e.belikov@epcc.ed.ac.uk)

>>> **DAY 4 : from line 1187** <<<
Please don't forget to check in and
to complete the post workshop survey: https://www.surveymonkey.com/r/dcpostworkshopassessment?workshop_id=2020-07-28-heriotwatt-dc-online

---

Day 1: Organizing data in Spreadsheets and Data Wrangling with OpenRefine

Check in: Please add your name, area of research, whether you have done some programming, and one of your favourite songs/bands

- Muhammad Junaid Arshad, Spin Physics, I use Python
- Matt gravener biology R
- Joseph Gibbons, experimental physics,
- I often use MatLab for processing my data, I'm a Linux user (Arch distro) and have been trying to learn Python3 and improve my data skills,
- (a favourite song/band choice is too hard)
- Xiang Shi, petroleum microbiologist. I use R
- Stephen Burnside, computational fluid dynamics, I use fortran
- Kym Craig, psychology, R and PsychoPy (Python)
- Cheng Siew Goh, Construction Management, not using any programme
- Lee Picken, Research Futures Academy, No programming
- Mohammed Almoghayer, Renewable Energy, I use python
- Burhan Turgut, Mechanical Engineering. Did a bit of MATLAB
- Seyed Reza Asadolahpour (Reza), Deep Learning in Petroleum Engineering (micro-CT rock images), MATLAB and C# (in the past) also a little bit of Python and R, Sleeping Sun from

Nightwish.

- Jessica Chen-Burger, logic programming
- Amal Abbas, business management , no programming
-
- Matthew Pauley Brewing and Distilling sensory science and surveys
-
- Elena Shevchenko, Digital Marketing, very basic limited experience with SPSS, R, Python, Knocking in the Heaven's Door (you know by whom;)
- Ranjeet, Composite Materials, Extensively used Matlab - looking to use R,
-
- Christie, Renewable Energy, Matlab and a bit of python
-
- Keith Davidson, Environment Science and Chemistry , expected to use matlab, anova, probability, minitab for research satistical data anlysis. new to programming

- Kerr Samson, Materials Science, small amount of matlab
- Berke Ricketti, Physics, Matlab/Python, Husker Du
- Alexey Narykov, Electrical Engineering, Matlab, Yo La Tengo+1
- Sofie Voerman, environmental science (marine biology), use R for programming, attempted python. Listen to Ben Caplan
- Yujia Han, supply chain management, R for programming.

Data : https://figshare.com/articles/SAFI_Survey_Results/6262019

Exercise: How regularly do you use spreadsheets? Have you had a frustrationg experience with it (share in a setence if you want)?
office 365-excel, windows 10.
Not very often, once a week
a few times a semester for record keeping and basic computing
often, but mostly for administrative documents, not research data

Exercise: Download messy data (https://ndownloader.figshare.com/files/11502824), load it into the Spreadsheet app , identify any issues you find with the data and suggest how could both tabs be merged into a sigle tidy data table.

- I think the data is difficult to understand, appears to contains errors +2, and I believe would be difficult to reuse +1.
- livestock_owned_and_numbers contains 2 foms of information in only one column. should be split.+3
- The id's are not unique. Also, the columns are not the same for the two tables, so merging would result in missing data.
- A consistent way of marking/assigning values is not seen. For example, n,N,No all mean the same

thing. +1

- Room numbers couldn't be negative Negative numbers are probably a way of showing missing or unavailable data.
- Mozambique Plots key_id 9 has -999 for plots.
- Adding symbols like * causes problems with formulae
- Negative values, numerical and text data in one tab
- Multiple tables on one sheet
- Spelling mistakes will confuse the data
- gaps (empty cells) in the spreadsheet
- a mix of long and wide formatting
- no headers
- headings with snakecase (i.e., with_between_text), others without
- key_id does not represent unique id values
- But multiple tables can help interlinking the data
- factorial experimental design

Tidy data

- a single variable per column
- a single observation per row
- don't combine multiple values in single cell

Tidy data principles:

- Avoid using multiple tables within one spreadsheet.
- Avoid spreading data across multiple tabs (but do use a new tab to record data cleaning or manipulations).
- Record zeros as zeros.
- Use an appropriate null value to record missing data.
- Don’t use formatting to convey information or to make your spreadsheet look pretty.
- Place comments in a separate column.
- Record units in column headers.
- Include only one piece of information in a cell.
- Avoid spaces, numbers and special characters in column headers.
- Avoid special characters in your data.

Exercise: Download the clean dataset (https://ndownloader.figshare.com/files/11492171), check that there are many more variables included now and add below some exaples of metadata that should be included with these data.

Please record your notes/ comments for the above excercise below:

Correct link (?): https://figshare.com/articles/SAFI_Survey_Results/6262019

The data doesn't use an appropriate separater (coma or semi-colon)

SAFI data summary: https://datacarpentry.org/socialsci-workshop/data/

Feedback:

Keep:

- exercises
- I like the way that this workshop starts with principles of tidy data in an excel sheet (a tool we may have all used before)
- online format is working well for this workshop
- real-time collaborative document
- infromative
- Online format is good if you have multiple screens, I suspect for those who don't, it is a bit more challenging
- quick reply via chat
- Collaborative document to share ideas and discuss; examples of how sorting out the data; could be more useful to have more hands-on practices

collaborative documents works well, may include some visual promps though
A collaborative working enviornment is great to have

To improve:

- Place all important URLs at the beginning of the work document
- Very frustrated with Zoom and running this online, as it is not possible to look at the screen and wotk on the file at the same time. Idelayy, should run on 2 machines!! and audio on the phone.Keeps on throughing me out of the sessions on both colaborative document and zoom meeting.
- I believe i missed a general introduction/ welcome when joining the session at 1pm.
- its easy to drift off for a few seconds with the online format, would be good to have all excercises/ solutions together to look back at a later time (e.g. the splitting date column excersise, cant find it in the online documents)
- Avoid dublicating links, e.g. https://datacarpentry.org/spreadsheets-socialsci/setup.html and https://figshare.com/articles/SAFI_Survey_Results/6262019

online format well put together, think I missed the email with the links to download the data etc +1 (emailed the organisers, got lost somewhere)

Break: until 14:30 (please be back on time)

Open Refine: openrefine.org

Download/ install/ setup OpenRefine:
https://openrefine.org/download.html
https://datacarpentry.org/openrefine-socialsci/setup.html
https://github.com/OpenRefine/OpenRefine/wiki/Installation-Instructions

Dataset for demonstration:
https://ndownloader.figshare.com/files/11502815

What is OpenRefine and what is possible with this tool?
https://github.com/OpenRefine/OpenRefine/wiki/Screencasts

Exercise: Using faceting, find out how many different *interview_date* values there are in the survey results. During what period were most of the interviews collected?

Hint: to produce a timeline display for *interview_date*. You will need to use **Edit cells** > **Common transforms** > **To date** to convert this column to dates (click on the drop-down in the column header)

Clustering info: https://github.com/OpenRefine/OpenRefine/wiki/Clustering-In-Depth

Exercise: remove the left square (**[**), the single quote marks ('), the right square brackets (**]**), and spaces from the *items_owned* column (using **Edit cells** > **Transform**...).

GREL (https://guides.library.illinois.edu/openrefine/grel **) commands**:

value.replace("'", "")
value.replace("]", "")
value.replace(" ", "")

- You should now have a list of items separated by semi-colons (;).

One can also 'daisychain' methods:
value.replace().replace().replace()

Additional source - illustrates the power of OpenRefine:
http://davidhuynh.net/spaces/nicar2011/tutorial.pdf

Exercise: Which two items are the most commonly owned? Which are the two least commonly owned?

Hint: split items_owned on semicolon (**;**), then use **Sort by** functionality.

Exercise: looking at *months_no_water* variable, which month relects most issues wrt water use?

Hint: use replace() and slpit() in custom text facet and Sort by (Sort ... then click count inside the facet window on the left)

Exercise: what village is the one associated with 49? What data could you use to identify it?
I noticed that 49 was situated in a row surrounded by Chirodzo.... perhaps the correct answer?
Using the date and times that the other interviews were carried out would suggest it's Chirodzo
49 is the number of entry (if that was asked)

Months no water
- value.replace("[","")

Exercise: Sort the data by *gps_Altitude*. Can you find any issues?

- value "0" (Yes)

How do you think the zero got recorded?
Missing data?
- loss of GPS signal
- Perhaps the value was unknown, and 0 (zero) was assigned instead.

- gps_latitude is not numeric; how do you convert it from text to num?
- got it, thanks

Can you imaging yourself applyign the tidy data principles to your data and OpenRefine to clean is?

- I think I would as you can generate visualisations for tidying data faster than R Studio. The syntax is fairly simple too, wouldn't take much to learn. I'm not great at re-using scripts in R Studio.

yes, seems to be a big step up from doing it excel as well.

Any personal experiences where knowing some of today's material would have helped?

- Looking for outliers (categorical/numeric) using OpenRefine would have saved me some time in preivous projects.

when hitting outliers/ wrong data when running R script, to quickly edit the data.

I usually obtain numerical raw data from an instrument (via experimental measurements), and I really hope to gain a better understanding and good practises of cleaning data, analysing data, and finally visualisation it to produce meaningful results. andOpenRefine already seems so powerful for this first step.

Potentially, I believe they are useful for my project, but I need to spend more time understanding it. I like the idea of keeping and being able to replicate the manipulations on a similar dataset.

https://datacarpentry.org/openrefine-socialsci/07-resources/index.html

Feedback:

Keep:

- content is good, covers a lot of data issues
- the live document is an excellent way administer exercises and discuss solutions
- I thought the lesson went well and at a good pace.
- I’m a Linux user and (obviously) dependent on FOSS; but I really liked the community aspect of the carpentry group and would like to contribute to it (going forward) if I can.
- Really helpful, fine for beginners, which is great.

- very useful information!
- Good help from Robert!
- Thank you.

To improve:

- pacing - oftentimes I was trying to catch up having submitted an answer to an earlier question
- can you highlight your cursor so we can track your mouse better? would help us keep on track with you
- add a discussion section to the live document do that we don't have to switch between Zoom and pad.carpentries.org
- The issue with virtual lessons, however, is that if I missed a step it can be difficult to go back to a specific point/step. (Ps - I appreciate how diffiuclt it is for you to manage/deliver a virtual session).
- It appears there is a lot of documentation/guidance/resource material available, however, I think accessing or locating it is a bit cumbersome – mainly because there appears to be multiple locations.
- Virtul lessons are very difficult. Probabaly, slowing down the pace, or splitting people into level groups could be helpful. Also, some introductory videos/materials could be useful for beginners to be watched before the workshop.
- Use one data set for us to work with. Not only does it save space on our own systems, but more importantly saves us one tab when switching around to keep up with you.
- Use a platform that allows captioning (Teams does). I missed a great deal of what you said, plus if it's a permanent stream, it would allow all of us to go back and catch up on directions that we missed switching between Zoom, internet, excel, etc or from tech issues.

Day 2: Introduction to R

SETUP: https://datacarpentry.org/r-socialsci/setup.html

R materials: https://datacarpentry.org/r-socialsci/
Dataset description: https://datacarpentry.org/socialsci-workshop/data/
We are going to use SAFI_clean.csv
https://ndownloader.figshare.com/files/11492171

Some additional material can be found under the URL below
(in the 'slides' folder, download the .pdf files to get access to embedded URLs):
https://github.com/robertn01/Data_Carpentry_SocialSciences_HeriotWatt

Sign In / Check In (from **line 296**): Please add your name, do you have more than one screen available (tablets included), also feel free to add a movie/series you'd recommend

Xiang Shi - I have two screens
Yujia Han, two screens, les bureau des legendes
Mohammed Almoghayer, two screens
Berke Ricketti, 2 screens, Bojack Horseman

Stephen Burnside, 2 screens
Peter McKenna; 2 screens
Lee Picken; 2 screens
Kerr Samson; 2 screens
sofie voerman; 1 screen
Elena Shevchenko, 1 screen :(
Alexey Narykov, 1 screen
Burhan Turgut, 2 screens
Muhammad Junaid Arshad, 1 screen
Joseph Gibbons, I'm using a dual-monitor setup, True Detective
Cheng Siew Goh - 2 screens

Fill in the Pre-workshop survey (if you havent done that yet):
https://www.surveymonkey.com/r/dcpreworkshopassessment?workshop_id=2020-07-28-heriotwatt-dc-online

Please rename youself in the meeting if it is consisting of numbers:
in Zoom Rooms click on Participants icon at the bottom, in the Participants window enter your name in the "New Screen Name" field. You can als check "Remember my name for future meetings"

1. Xiang Shi
2. Yujia Han
3. Mohammed Almoghayer
4. Stephen Burnside
5. Jessica Chen-Burger
6. Alexey Narykov
7. Elena Shevchenko
8. Keith Davidson
9. Peter McKenna
10. Berke Ricketti
11. Burhan Turgut
12. sofie voerman
13. kym
14. ranjeet
15. Lee Picken
16. Goh

Uselful URLs

https://cran.r-project.org

https://www.tidyverse.org

Cheatsheets

https://rstudio.com/resources/cheatsheets/

R Intro: commands (execute Ctrl + Enter on Windows; Cmd + Enter on Mac)

comment -- a line preceded with the hash sign; it is not run as a command by R

help(ls) # shows help in the RStudio's help pane on thecommand ls

R.version # to check R version; for more info; could use *sessionInfo()*

install.packages("tidyverse") # to install the *tidyverse* package

search() # optional: call this function to print out all the 'active' packages loaded into RStudio

library(tidyverse) # after installed a package you need to load it into R!

dir.create("data") # example R syntax to create a directory (can use RStudio's GUI too to create a new dir.)
dir.create("data_output")
dir.create("fig_output")

download.file("https://ndownloader.figshare.com/files/11492171","data/SAFI_clean.csv",mode="wb")
alternative: use SAFI_clean.csv from yesterday and copy it manually into data folder

getwd() # check what is your current working directory path

File > **New File** > **R Script** to create the script file; **File** > or Ctrl+S to save (Cmd + S on Mac)

Sections: Shift + Ctrl/ Cmd + R # and you will have the chance to name section

<- # assignment operator (shortkey: Windows: Alt + '-' ; Mac: Option + '-')

arithmetic operations, using R as calculator
3+5
12/7

x <- 5
7 -> y # (arrow head points towards named variable/ object)
(y <- 7) # combine value assignment and print to console operations

area_hectares <- 1.0
area_hectares # to check the value, displayed in the console window
(area_hectares <- 1.0)
2.47 * area_hectares
area_acres <- 2.47 * area_hectares
area_hectares <- 50.0 # re-assignment, now the old value is overwritten by 50.0

NB: you can check your variables in the Environment pane

#question: ## please post questions into the chat

#I have entered the following and got the results below: (all variables are shown as row 1, i.e. [1]):
#> x = 2
#> y = 3
#> z = 4
#> x
#[1] 2
#> y
#[1] 3 <--- shouldn't this be [2] ?
> z
#[1] 4 <--- shouldn't this be [3] ?
#>
#answer: the printed values are as expected 2,3 and 4 , respectively; disregard the [1] in the output #for now, this is the way the value is printed, it is followed by the value; we should use the #assignment operator **<-** see below
x <- 2 # now you can refer to value 2 by the using x
y <- 3 # y refers to 3
z <- 4 # z to 4 (internally R creates an 'object' to store the value in the memory allocated for it)

# **Exercise**: body mass index calculation
height <- 172
weight <- 81.7
(bmi <- weight / (height * height))

Functions
functions -- call using the following syntax: *function_name***(***arguments***)** where function name is the # name e.g. sqrt and arguments are a comma separated list, enclosed in parentheses

sqrt(4) # apply a function to a value
a <- 4
sqrt(a) # can pass a variable as input
round(3.14159) # round numeric values to the closest integer. (one could specify desired decimal places by using an optional parameter)
round(3.14159, 2) # see comments above - this syntax is equivalent round(3.14159, digits = 2)

getting help
args(round) # using function name as argument to args, to print expected
help(round) # see infor about round function the Help tab
?round
??round
# some packages have tutorial-like vignettes with usage examples
vignette(readr)

vectors and data types
help(c) # c for combine

hh_members <- c(3,7,10,6)

```
hh_members

respondent_wall_type <- c("muddaub", "burntbricks", "sunbricks")
# see how it appears in the Envisonment pane
respondent_wall_type

Later we will see after loading the SAFI_clean.csv into RStudio..
e.g. script:
library(tidyverse)              # loading tidyverse package /its functions into RStudio
survey <- read_csv('data/SAFI_clean.csv')
colnames(survey)                        # print out variable or column names in the 'survey' table called a data.frame or tibble object in RStudio - you can use class(survey) to check object class or type
respondent_wall_type <- survey$respondent_wall_type

# (!) vectors are homogenous -- they have elements of the same type (e.g. numeric for hh_members)

length(hh_members)
length(respondent_wall_type)

class(hh_members)
class(respondent_wall_type)

str(hh_members)
str(respondent_wall_type)

possessions <- c("radio", "bicycle", "television")          # can use single or double quotations
possessions <- c(possessions, "mobile_phone")           # append an element
possessions <- c("car", possessions)                    # prepend an element

# logical or Boolean

1 == 1    #TRUE
1 == 0   #FALSE
class(TRUE)  #  logical
x <- 1
typeof(x)
class(x)

x <- 1L
typeof(x)
class(x)

# Complex numbers

x <- 1 + 4i
typeof(x)
class(x)

# Characters
```

```
b <- "Mario"
typeof(b)
class(b)

# Mixed data will be converted to the 'highest' according to the data type hierarchy

n <- c(1, 2, 3, "a")
typeof(n)
n <- c(TRUE, 1, 2, 3, F). # TRUE and T are equivalent representation of logical values (analogue case for False)
class(n)
```

Question: If logical TRUE and FALSE are represented internally as 1 and 0, then how may we get the original representation back as logical symbols?
Answer: from programmer's point of view all variables are 'objects', TRUE and FALSE technically dont have to be represented as a number; we can explicitly coerce a value as **as.logical(1)** , which will return **TRUE**; in general FALSE is often equvalent to 0, while TRUE is something non-zero; e.g. as.logical(99) will also yield TRUE.
Recommendation: do not rely on automatic coercion in your code.

```
# Automatic coercion

#(roughly) Logical/ boolean --> Integer --> Double/ float --> Character --> List

# Subsetting vectors

ls() # lists objects stored in RStudio's Environment
respondent_wall_type
respondent_wall_type[2] # in R indexing starts from 1
respondent_wall_type[c(2,4)]
respondent_wall_type[2:5]      # syntax ~
named_object[start_index_value_included:stop_index_value_excluded]

# Conditional subsetting (vectors)
ls()
hh_members
hh_members[c(T, F, T, F)]              # equivalent to hh_members[c(TRUE, FALSE, TRUE, FALSE)]
hh_members > 5
hh_members[hh_members > 5]

# 'AND' ( & ), 'OR ( | )' - logical statements              used to chain together selection conditions

# operators: ==, >, <, >=, <=, !
```

(https://www.tutorialgateway.org/logical-operators-in-r/)

```
hh_members[hh_members < 3 | hh_members > 5]
hh_members[hh_members >= 7 & hh_members == 3]
possessions[possessions == 'car' | posssessions == 'bycicle']


# AND
# TRUE & TRUE          yields TRUE
# TRUE & FALSE                FALSE
# FALSE & TRUE                FALSE
# FALSE & FALSE                FALSE
#

# OR

# TRUE & TRUE          yields TRUE
# TRUE & FALSE                TRUE
# FALSE & TRUE                TRUE
# FALSE & FALSE                FALSE
#

# NOT
# !             # negation operator
!FALSE         # yields TRUE

# %in%

possessions %in% c('car', 'bycicle')
possessions[possessions %in% c('car', 'bycicle')]

# missing data NA
rooms <- c(2, 1, 2, NA, 4)
mean(rooms)   # (!) will not work because of NA value
max(rooms)
is_na()   # can be used to check whether a value is NA
mean(rooms, na.rm = TRUE)   # first removes NAs
max(rooms, na.rm = TRUE)
# also check
# na.omit
# complete.case
```

Question: why Ihave gotten the follwoing results:
>res = c("mud", "burn", "t3", "t4", "t5")
>res[c(TRUE, FALSE)]
[1] “mud” “t3” “t5”

Answer:if the c() vector isn't the same length as the object/ vector you are going to apply this boolean filter...
the filter (T, F) will be repeated until reach the end of the 'res' object

mud -- TRUE ---> print out
burn -- FALSE --> won't be printed out on the console
...etc.
ONLY elements under 'TRUE' index will be kept and returned/ printed - here every second value

Exercise:

```
rooms <- c(1, 2, 1, 1, NA, 3, 1, 3, 2, 1, 1, 8, 3, 1, NA, 1)

# 1. given rooms, create a vector with NAs removed
crooms <- rooms[!is.na(rooms)]

# 2. use median() to calculate the median of the rooms vectors
median(rooms, na.rm = TRUE)
#or
median(crooms)

# 3. use R to figure out how many households in the set use more than 2 rooms
crooms[ crooms > 2 ]
length( crooms[ crooms > 2 ] )

# DataFrames and Tibbles

#NB: you can install tidyverse using: install.packages("tidyverse")

library(tidyverse)              # load a previously installed package into RStudio IDE to get access to its full functionality (all the functions, methods, routines, predefined constants/ other objects, etc)

interviews <- read_csv('data/SAFI_clean.csv', na = 'NULL')
View(interviews)        # NB: capital 'V' !          it is an RStudio-specific functionality

?read_csv
class(interviews)
typeof(interviews)

# Inspecting DataFrames

dim(interviews)                 # returns with the dimensions (number of rows and columns)
nrow(interviews)                # returns with the number of rows or records or observations stored
ncol(interviews)                # returns with the number of columns or variables or parameters
head(interviews)                # returns with the top 6 rows by default - one can customise:
head(interviews, 10)
tail(interviews)                 # returns with the last 6 rows by default
names(interviews)               # returns with the column names, equivalent to colnames(interviews)
names(interviews)[2]            # second column's name

str(interviews)
summary(interviews)
```

```
glimpse(interviews)     # alternative  to str()



# Index and subset DataFrames

# [row_number, col_number]

interviews[1,1]
interviews[1,6]

interviews[[1]]

seq(1,10)
seq(1,10, 2)         # generate a sequence between 1 and 10, with step size 2

interviews[1:3, 7]
interviews[3,]              # or equivalently  interviews[3,:]
interviews[1:3,]

head_interviews <- interviews[1:6,]             # NB: here subsetting interval start and stop elements are
included

interviews[,-1]
interviews[,-c(7:8)]

names(interviews)
interviews['years_liv']
interviews[['years_liv']]
interviews$rooms



# Factors - for categorical variables, can be used as facets when plotting

# red - 1
# green - 2
# blue - 3

favcol <- factor(c('red', 'blue', 'green', 'green', 'blue', 'red' ))        # assigning alphabetical order as 'levels'
(a: smallest - z:highest)
favcol

plot(favcol)
# your Plots pane needs to be large enough to view the figure (may need to decrease font size)

# dplyr ------ for data manipulation/ transformation
# part of tidyverse
```

```
# now we can iteratively build data analysis pipelines

names(interviews)
select(interviews, village, no_members, years_liv)
filter(interviews, village == 'God')

select(filter(interviews, village == 'God'), village, no_members, years_liv)

# the pipe operator %>%      (Shortkeys: Win/ Linux: Shift + Ctrl + M; Mac: Shift + Cmd + M)
# from magrittr package
# (!) NB: %>% can't be on the beginning of the line (you will get an error message otherwise)

# dplyr verbs:
# filter : subset dataset by rows
# select : subset dataset by columns
# mutate : introduce a new column based on transforming other data (can refer to same dataset e.g. #
for extracting day, hour, month fro ma date
# group_by : group data by one or multiple variables e.g. by village
# summarise : (reduction operation) get  some stats e.g number of entries etc.
# there are more ...
# aggregate ...
# count ...
# check out dplyr documentation for more details

interviews %>% filter(village == 'God') %>%
                    select(village, no_members, years_liv)

# can do assignment:

interviews %>% filter(village == 'God') %>%
                    select(village, no_members, years_liv) -> interviews_God

 interviews_God <- interviews %>%
                       filter(village == 'God') %>%
                       select(village, no_members, years_liv)

interviews %>% mutate(people_per_room = no_membrs/ rooms)

interviews %>% group_by(village) %>%
                    summarise(mean_no_members = mean(no_membrs))

interviews %>% group_by(village) %>%
                    summarise(mean_no_members = mean(no_membrs)) # %>%
                    # ungroup()

interviews %>% group_by(village, memb_assoc) %>%
                     filter(!is.na(memb_assoc)) %>%
                     summarize(mean_no_members = mean(no_membrs))  %>%    # note summariZe vs
summariSe
```

arrange(mean_no_mebrs) # by default ASC order, descending order: arrange(desc(mean_no_mebrs))

interviews %>% count(village)

materials: https://datacarpentry.org/r-socialsci/ (we almsot got through lesson 3, and we will continue from the onward, and we will get to plotting)

Feedback:

Keep:

- focus on tidyverse functions
- Well explained in great details, I really enjoyed it!
- Thank you
- Well covered on multiple aspects
- Really thorough, lots to take in.

To improve:

- good practice in R: copy and paste regularly rather than typing
- A little bit hard to follow at the very end, but that is understandable due to limited timing.
- Some things were too obvious, could have used that time for more complex examples

Added a copy of what I typed out here:

- https://github.com/marioa/public-misc/blob/master/Carpentries/CarpentryScript-29Jul20.R

The problem I had plotting the factors on the afternoon did not happen once I reduced the size of the font. I think
I had another problem with group_by but this seemed work ok.

Day 3: Data Analysis and Visualisation in R

Check-in: please add your name and one thing that was most frustrating so far and one thing that you learned that was useful
dow
1.Alexey Narykov, juggling the windows
2.Kerr Samson, remembering R commands from last week...!
3.Yujia Han. Too many commands to remember, not sure if I shall remember all of them. Benefited much from the setting up instructions (i.e. creating seperate folders in the wd), which was not learnt from other workshops.

4. Sofie Voerman. shortcut for %>% and writing them was useful. slightly less useful is the slow speed sometimes to make sure everyone can catch up.
5.Burhan Turgut, too many commands,
6.Elena Shevchenko, could not follow most of it
7.Kym Craig, frustrating sometimes is switching between everything (package, Zoom, this sheet), useful - open refine
8. Ranjeet, handling multiple windows, R has much more capabilities than I had thought!
9.Xiang Shi, good knowledge abt R
10. Stephen Burnside, remembering all the commands
11. Berke Ricketti, number of comamnds to remember in R, data file organisation
12.Muhammad Junaid Asrhad, good to get basics of R
13. Cheng Siew Goh, not fully understand the commands and their purposes, good thing is to get some basic knowledge of R
14. Mohammed Almoghayer, remembering all the commends and a getting little bit confused with the different commands between R and Python.
15.Lee Picken - like most others, the hardest thing is remembering all of the commands.
16. Joseph Gibbons
17.
18.
19.
20.
21.
22.
23.
24.

```
library(tidyverse)

# (optional) if you don't have the data folder in your working directory
dir.create("data")

# load data (if you had a project from last week)
interviews <- read_csv("data/SAFI_clean.csv", na = "NULL")

# download the data
download.file("https://ndownloader.figshare.com/files/11492171",
              "data/SAFI_clean_dl.csv", mode = "wb")

# factors recap
respondent_floor_type <- factor(c('earth', 'cement', 'cement', 'earth'))

str(respondent_floor_type)

plot(respondent_floor_type)

nlevels(respondent_floor_type)

# reorder levels
```

```
respondent_floor_type <- factor(respondent_floor_type, levels = c("earth",
"cement"))

# inspect
respondent_floor_type

# change a level
levels(respondent_floor_type)[2] <- "brick"

# note changed levels
levels(respondent_floor_type)

# factrored can be ordered
respondent_floor_type_ordered = factor(respondent_floor_type, ordered=TRUE)

# can convert factors to cahrecters
as.character(respondent_floor_type)

# e.g.
year_fct <- factor(c(1990, 1983, 1977, 1998, 1990))

# if we want to get the values we can't do as.numeric(year_fct) because it will give us the indices
# we need to instead use the levels associated to years values to as indices to get the values
# before converting to numbers
as.numeric(levels(year_fct))[year_fct]

## working with dates

library(lubridate)

str(interviews)

dates <- interview$interview_date

interviews$day <- day(dates)
interviews$month <- month(dates)
interviews$year <- year(dates)

# iteratively building data analysis pipelines
# disregard observations with NA value for memb_asoc
# then group by village and memb_assoc (whether a household is a member of irrigation association)
# then summarise the mean and min numbers for household members for the groups
interviews %>%
    filter(!is.na(memb_assoc)) %>%
    group_by(village, memb_assoc) %>%
    summarize(mean_no_membrs = mean(no_membrs), min_membrs =
min(no_membrs)) %>%
    arrange(min_membrs)
```

```
interviews %>%
  count(no_meals)
```

dplyr excercise:

How many households in the survey have an average of two meals per day? Three meals per day? Are there any other numbers of meals represented?

Solution:

```
interviews %>%
  count(no_meals)
```

#Ex2:

Use group_by() and summarize() to find the mean, min, and max number of household members for each village. Optional: Also add the number of observations (hint: see ?n).

Solution:

```
interviews %>%
  group_by(village) %>%
  summarize(
    mean_no_membrs = mean(no_membrs),
    min_no_membrs = min(no_membrs),
    max_no_membrs = max(no_membrs),
    n = n()          # n() to be used inside summarise()
  )
```

Reshaping data with 'tidyr'

```
# The four rules defining a tidy dataset:

# 1. Each variable has its own column
# 2. Each observation has its own row
# 3. Each value must have its own cell
# 4. Each type of observational unit forms a table
```

Pivoting wider (AKA 'wide format')

```
# pivot_wider() takes three principal arguments:

# 1. the data
# 2. the names_from column variable whose values will become new column names.
# 3. the values_from column variable whose values will fill the new column variables.

# Further arguments include 'values_fill' which, if set, fills in missing values with the value provided.
```

Example script:

Pivoting longer (AKA 'long format')

pivot_longer() takes four principal arguments:

1. the data
2. cols are the names of the columns we use to fill the values variable (or to drop).
3. the names_to column variable we wish to create from column names.
4. the values_to column variable we wish to create and fill with values associated with the named columns.

Example script:

```
interviews_long <- interviews_wide %>%
   pivot_longer(cols = c(burntbricks, cement, muddaub, sunbricks),
                names_to = "respondent_wall_type",
                values_to = "wall_type_logical")
```

Feedback:

Keep:

- It is very helpful to have tutors tagging along the workshop to explain what went wrong in the error messages and offer additional advice, other than the instructor's teaching
- Thorough explanation on the details, short keys and varieties of a function
-
-

To improve:

- A bit confused on which conversation board to use (here or zoom?), could have make a fixed arragement perhaps?
-
-
-

Day 3 - part 2: **Intrpo to visualisation in R using ggplot2**
Tidy data principles:

Each variable has its own column
Each observation has its own row
Each value must have its own cell
Each type of observational unit forms a table

```
library(tidyverse)   # load a package, also loads ggplot2

# get the data
download.file("https://ndownloader.figshare.com/files/11492171",
              "data/SAFI_clean.csv", mode = "wb")


# load the data


interviews_plotting <- interviews %>%
    ## pivot wider by items_owned
    separate_rows(items_owned, sep=";") %>%
    mutate(items_owned_logical = TRUE) %>%
    pivot_wider(names_from = items_owned,
                values_from = items_owned_logical,
                values_fill = list(items_owned_logical = FALSE)) %>%
    rename(no_listed_items = `NA`) %>%
    ## pivot wider by months_lack_food
    separate_rows(months_lack_food, sep=";") %>%
    mutate(months_lack_food_logical = TRUE) %>%
    pivot_wider(names_from = months_lack_food,
                values_from = months_lack_food_logical,
                values_fill = list(months_lack_food_logical = FALSE)) %>%
    ## add some summary columns
    mutate(number_months_lack_food = rowSums(select(., Jan:May))) %>%
    mutate(number_items = rowSums(select(., bicycle:car)))
```

Concept: gradually add layers in a specific order

```
# DATA %>%
#   ggplot(aes(<MAPPINGS>)) +
#   <GEOM_FUNCTION>()     # geometric objects 'geom'
#   gradually building up the graphics

interviews_plotting %>%
  ggplot()

#scatter plot

interviews_plotting %>%
  ggplot(aes(x = no_membrs, y = number_items)) +
  geom_point()
```

```
# assign plot to variable - create a plot object
interviews_plot <- interviews_plotting %>%
  ggplot(aes(x = no_membrs, y = number_items))

interviews_plot +
  geom_point()

interviews_plot +
  geom_point(alpha = 0.5, color = "blue")

interviews_plot +
  geom_jitter(aes(color = village), alpha = 0.5)    # resolve overplotting
```

Exercise: Create a scatter plot of *rooms* by *village* with the *respondent_wall_type* showing in different colors. Is this a good way visualise?

```
interviews %>%
  ggplot(aes(x = village, y = rooms)) +
  geom_jitter(aes(color = respondent_wall_type))

interviews %>%
  ggplot(aes(x = village, y = rooms)) +
  geom_boxplot()   +   # + coord_flid()
  geom_jitter(alpha = 0.5, color = 'tomato')



interviews %>%
  ggplot(aes(x = village, y = rooms)) +
  geom_jitter(aes(color = respondent_wall_type))

# box/violin plot

interviews_plotting %>%
  ggplot(aes(x = respondent_wall_type, y = rooms)) +
  geom_boxplot() + coord_flip()



interviews_plotting %>%
  ggplot(aes(x = respondent_wall_type, y = rooms)) +
  geom_boxplot() +
  geom_jitter(alpha = 0.5, color = "tomato")

interviews_plotting %>%
  group_by(respondent_wall_type, rooms) %>%
  mutate(num = n()) %>%
  ggplot(aes(x = respondent_wall_type, y = rooms)) +
  geom_tile(aes(fill = num))
```

```
interviews_plotting %>%
  ggplot(aes(x = respondent_wall_type, y = rooms)) +
  geom_violin() +
  geom_jitter(alpha = 0.5, color = "tomato")


# bar chart
interviews_plotting %>%
  ggplot(aes(x = respondent_wall_type)) +
  geom_bar(fill = "red", colour = "black")

interviews_plotting %>%
  ggplot(aes(x = respondent_wall_type)) +
  geom_bar(aes(fill = village))

interviews_plotting %>%
  ggplot(aes(x = respondent_wall_type)) +
  geom_bar(aes(fill = village), position = 'dodge')

interviews_plotting %>%
  filter(respondent_wall_type != 'cement') %>%
  ggplot(aes(x = respondent_wall_type)) +
  geom_bar(aes(fill = village), position = 'dodge') +
  xlab("Wall type") +
  ylab("number of observations")


percent_wall_type <- interviews_plotting %>%
  filter(respondent_wall_type != "cement") %>%
  count(village, respondent_wall_type) %>%
  group_by(village) %>%
  mutate(percent = n / sum(n) * 100) %>%
  ungroup()

percent_wall_type %>%
  ggplot(aes(x = village, y = percent, fill = respondent_wall_type)) +
  geom_bar(stat = "identity", position = "dodge")

# with custom labels
percent_wall_type %>%
  ggplot(aes(x = village, y = percent, fill = respondent_wall_type)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Proportion of wall type by village",
       x = "Village",
       y = "Percent")
```

Exercise: Create a bar plot showing the proportion of respondents in each **village** who are or are not part of an irrigation association (**memb_assoc**). Include only respondents who answered that question in the calculations and plot. Which village had the lowest proportion of respondents in an irrigation association?

Solution:

```
# unique(interviews_plotting$memb_assoc)     # check unique values

percent_memb_assoc <- interviews_plotting %>%
  filter(!is.na(memb_assoc)) %>%
  count(village, memb_assoc) %>%
  group_by(village) %>%
  mutate(percent = n / sum(n) * 100) %>%
  ungroup()   # internally removes aggregation/ grouping

percent_memb_assoc %>%
  ggplot(aes(x = villag, y = percent, fill = memb_assoc)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Proportion of irrigation association members by village",
```

- x = "Village",
- y = "Percent") +

```
  facet_wrap(~village)                    # creates subplots withing a plot based on categorical variable


# faceting : looking at items

percent_items <- interviews_plotting %>%
  gather(items, items_owned_logical, bicycle:no_listed_items) %>%
  filter(items_owned_logical) %>%
  count(items, village) %>%
  ## add a column with the number of people in each village
  mutate(people_in_village = case_when(village == "Chirodzo" ~ 39,
                                       village == "God" ~ 43,
                                       village == "Ruaca" ~ 49)) %>%
  mutate(percent = round(digits=2, n / people_in_village * 100))

# Step 1 - prepare the data

percent_items <- interviews_plotting %>%
  gather(items, items_owned_logical, bicycle:no_listed_items) %>%
  filter(items_owned_logical) %>%
  count(items, village) %>%
  ## add a column with the number of people in each village
  mutate(people_in_village = case_when(village == "Chirodzo" ~ 39,
                                       village == "God" ~ 43,
                                       village == "Ruaca" ~ 49)) %>%
  mutate(percent = round(n / people_in_village * 100, digits = 2))

# Step 2 - do the plotting

percent_items %>%
```

```
  ggplot(aes(x = village, y = percent)) +
  geom_bar(stat = 'identity', position = 'dodge') +
  facet_wrap(~items) +
  theme_bw()                # there are many other 'theme'


# ggplot2 themes

#save plot object
myplot <- percent_items %>%
  ggplot(aes(x = village, y = percent)) +
  geom_bar(stat = "identity", position = "dodge") +
  facet_wrap(~items) +
  labs(title = "Percent of respondents in each village who owned each item",
     x = "Village",
     y = "Percent of Respondents") +
  theme_bw() +
  theme(text = element_text(size = 16))

dir.create("fig_create")


# export plot as png (alternativel use the Export button in the Plots pane)

ggsave("fig_output/myplot.png", myplot, width = 15, height = 10)
```

Feedback:

Keep:

- Worked examples and exercises
-
- The presenters knowledge of the language is apparent, and I enjoy when they add their own opinions and tips as they are going through the workshop steps (so long as it doesn't delay the progress of the workshop).
- prompt and helpful answer in chat
-

To improve:

- Overall pace of delivery - slow at start then rushing through material at end so difficult to keep up and take in all information
-
- In this workshop I found the continued interruption of the presenter by one of the helpers distracting, appeared to disrupt the flow of the lesson, and was a little frustrating.
- What do we need to install for tomorrow's lesson? Is installing DB.Browser for SQLite is sufficient?
-

Day 4: Introduction to databases and data management with SQL

Check-In: name + pronoun

1.Xiang Shi
2.Peter McKenna (he)
3.Yujia Han (she)
4.Stephen Burnside
5.Muhammad Junaid Arshad
6. Joseph Gibbons
7. Cheng Siew Goh
8. Kym Craig (she)
9.Alexey Narykov (he)
10.Lee Picken
11. Mohammed Almoghayer
12.sofie voerman
13.Elena Shevchenko
14.Kerr Samson (he)
15. Berke Ricketti (he)
16. Ranjeet
17.Burhan Turgut(he)
18.
19.
20.
21.
22.
23.
24.

DATA TYPES in SQLite

NULL representing missing data; can specify whether a column can have mising date when creating a table
INTEGER (signed up to 8 bytes)
REAL (8 bytes floating point approx number)
TEXT string
BLOB binary data (stored as is, 'raw')

Set background for missing values:
Edit -> preferences -> Data Brwoser => set NULL bgcolor to red

SELECT satement and 6 components: used to fitler by columns

```
SELECT colnames
FROM tablename
WHERE conditions
GROUP BY colnames
HAVING conditions
ORDER BY colnames
```

```
SELECT *
FROM Farms
LIMIT 10;
```

```
-- select specific column names
SELECT Id, B16_years_liv
FROM Farms
LIMIT 10
;
```

Exercise: get 5 first rows from **Farms** table with only columns Id, B16 to B20

Solution:

```
SELECT Id
    , B16_years_liv
    , B17_parents_liv
    , B18_sp_parents_liv
    , B19_grand_liv
    , B20_sp_grand_liv
FROM Farms
LIMIT 5
;
```

WHERE can be used to filter rows:

```
SELECT Id, B16_years_liv
FROM Farms
WHERE B16_years_liv > 25
;
```

other operators: <, <=, =, >=, <>, use AND and OR t ocombine conduitions

```
SELECT Id, B16_years_liv
FROM Farms
WHERE B16_years_liv < 18 OR B16_years_liv > 65
;
```

```
-- use IN t ochec kfor particular values
```

```
SELECT Id, B16_years_liv
FROM Farms
WHERE B16_years_liv IN (10,20,30,40,50,60)
;

-- use IS when testing for NULL   NB: Text 'NULL' is not the same as NULL (special value for missing value)
SELECT *
FROM Farms
WHERE F14_items_owned IS NOT NULL
;
```

Exercise:
Write a query using the Farms table which returns the columns Id, A09_village, A11_years_farm, B16_years_liv. We are only interested in rows where the A09_village value is either ‘God’ or ‘Ruaca’. Additionally we only want A11_years_farm values in the range 20 to 30 exclusive and B16_years_liv values strictly greater than 40. There are many ways of doing this, but try to use an inequality, an IN clause and a BETWEEN clause.

Solution:

```
SELECT Id, A09_village, A11_years_farm, B16_years_liv
FROM Farms
where (A09_village = "God" OR
      A09_village = "Ruaca" ) AND
     (A11_years_farm > 20 and
      A11_years_farm <30) AND
      (B16_years_liv > 40);
```

DOCS: https://www.sqlite.org/fullsql.html

```
-- sorting using ORDER BY default is ASC (ascending)
SELECT Id, A09_village, A11_years_farm, B16_years_liv
FROM Farms
WHERE A09_village = 'God'
ORDER BY A11_years_farm DESC
;

-- UPDATE  to change values
UPDATE Farms
SET F14_items_owned = '[]'
WHERE F14_items_owned is NULL
;

-- creating new columns
SELECT D02_total_plot, D02_total_plot * 2.4701 AS D02_total_plot_converted
FROM Plots
;
```

```
-- using a built-in function ROUND
SELECT D02_total_plot, ROUND(D02_total_plot * 2.4701, 2) AS D02_total_plot_converted
FROM Plots
;
```

Exercise:
Write an SQL query which returns the **Id**, **plot_Id**, **D01_curr_plot** and **D02_total_plot** columns from the **Plots** table with the addition of a calculated column representing the **plot area in acres** and **a column representing the units of the calculated column**.

Solution(s):
home work

other built in function: e.g. substr(a,b,c) and instr(a,b) and || for string concatenation

BACK at 3pm

```
SELECT A01_interview_date, A04_start, A05_end
FROM Farms
;
```

```
-- get date text to yyyy-mm-dd  format
SELECT A01_interview_date,
  date(
   substr(A01_interview_date, 7, 4) || '-' ||
   substr(A01_interview_date, 4, 2) || '-' ||
   substr(A01_interview_date, 1, 2)
  ) as converted_date
FROM Farms
ORDER BY converted_as_text
;
```

```
-- conditional column creation
SELECT Id, Country,
  CASE
    WHEN 'Moz' THEN 'Mozambique'
    WHEN 'Taz' THEN 'Tanzania'
  ELSE 'Unknown Country'
  END AS country_full
FROM Farms
;
```

Exercise:
select Id, A11_years_farm from Farms and bin in a new colum by years into categories 1-20,21-40,41-60, > 60.

Solution:

```
SELECT Id, A11_years_farm,
    CASE
    When A11_years_farm <21 Then '1-20'
    When (A11_years_farm>20 and A11_years_farm <41) then '21-40'
    When A11_years_farm between 41 and 60 then '41-60'
    When A11_years_farm > 60 Then '>60'
    ELSE 'Unknown'
    END AS category
FROM Farms;
```

```
SELECT Id, A11_years_farm,
  CASE
    WHEN A11_years_farm BETWEEN 1 AND 20 THEN '1-20'
    WHEN A11_years_farm BETWEEN 21 AND 40 THEN '21-40'
    WHEN A11_years_farm BETWEEN 41 AND 60 THEN '41-60'
  ELSE '> 60'
  END AS years_farm_bin
FROM Farms
;

-- AGGREGATION

SELECT
      min(A11_years_farm),
      max(A11_years_farm),
      avg(A11_years_farm)
FROM Farms
;

SELECT DISTINCT A09_village
FROM Farms
;

SELECT A09_village,
  COUNT(*) as num_obs
FROM Farms
GROUP BY A09_village
;
```

Exercise:
get min, max and avg values as well as a count of the number of records involved for the ‘A11_years_farm’ column for each village (A09_village) in the ‘Nhamatanda’ district (A07_district)

Solution suggestions:

```
SELECT A09_village,
```

```
    min(A11_years_farm) as min,
    max(A11_years_farm) as max,
    round(avg(A11_years_farm), 2) as average,
    count(A11_years_farm) as entries
FROM Farms
WHERE A07_district = 'Nhamatanda'
group by A09_village;

SELECT A09_village,
  min(A11_years_farm) as min,
  max(A11_years_farm) as max,
  ROUND(avg(A11_years_farm),2) as avg
FROM Farms
WHERE A07_district = 'Nhamatanda'
GROUP BY A09_village
;

SELECT A08_ward,
    min(A11_years_farm) as min_years,
    max(A11_years_farm) as max_years,
    count(*) as how_many_farms
FROM Farms
GROUP BY A08_ward
HAVING how_many_farms > 2;
```

Exercise: Using the Crops table write a query which will list all of the crops (D_curr_crop) which are grown in over 100 plots.

Solution suggestions:

```
SELECT D_curr_crop, COUNT(*) AS num_plots
FROM Crops
GROUP BY D_curr_crop
HAVING num_plots > 100
;

-- creating tables from other tables

CREATE TABLE Farms_location AS
SELECT Id,
    Country,
    A06_province,
    A07_district,
    A08_ward,
    A09_village
FROM Farms;

--
install.packages("RSQLite")
```

```
library("RSQLite")   # requrie recent R

dbfile = "data/SQL_SAFI.sqlite"
sqlite = dbDriver("SQLite")
mydb = dbConnect(sqlite, dbfile)
results = dbSendQuery(mydb, "SELECT * FROM Crops LIMIT 10")

data = fetch(results)

View(data)  #in RStudio
dbClearResult(results)
```

-- JOINS

Q: Which Farms with more than 12 people in the household grow Maize?

(Good R book https://r4ds.had.co.nz)

Joining multiple tables

Q: For Farms with more than 12 people in the household how much land is devoted to growing Maize?

```
SELECT a.Id as Farms_Id, a.B_no_membrs,
      b.Id, b.plot_id as plot_id, b.D02_total_plot,
      c.Id as Crops_Id, c.plot_Id as crops_plot_id, c.D_curr_crop
FROM Farms as a
JOIN Plots as b
JOIN Crops as c
ON a.Id = b.Id
   AND (b.Id = c.Id AND b.plot_id = c.plot_id)
   AND a.B_no_membrs > 12
   AND c.D_curr_crop = 'maize'
;
```

Exercise: Modify the query above so that only the ‘Id’, ‘D02_total_plot’ and the ‘D_curr_crop’ columns are shown and at the same time *summarise* the data so that there is only *one entry for each Farm*. i.e sum the ‘D02_total_plot’ column.

Solutio nsuggestions:

Post-workshop:
complete the post workshop survey: https://www.surveymonkey.com/r/dcpostworkshopassessment?workshop_id=2020-07-28-heriotwatt-dc-online

Feedback:
Thanks for organising the hands-on workshop. Appreciated. +1

Although I enjoyed learning about SQlite, I'm unsure why this may be used as opposed to OpenRefine. For example, is DQlite to be used to 'clean' data and sort it for analysis? Or is it a tool to be used for outputing data for reuse by someone else?
Thanks lot of examples covered with quite details.
Keep:

- the syntax and operations are worth keeping
- The excercises make this session exceptionally fun
- Informative and friendly sessions +1
- Again, well taught (as was the case for the other workshops) +1

To improve:

- The command line demostration is complete black screen and I was not able to see anything. Found an URL on this problem, but I was not able to use this solution (https://support.zoom.us/hc/en-us/articles/202082128-Black-Screen-During-Screen-Sharing) Thought to share here anyway.
- perhaps spend a little more time on joins?
- As a brand new beginner for SQL, I find it a little bit difficult to concentrate taking all information, especially the last 30 mins.
- The last session was quite fast and hard to catch up.
- Emphasing and addressing the motivaiton and strengths of using SQlite over other tools (e.eg, OpenRefine, etc)
- remind everyone to install software the ws prior, so no need to catch up and get lost/ wait for participants