Andreas Tang Hansen
AU641294

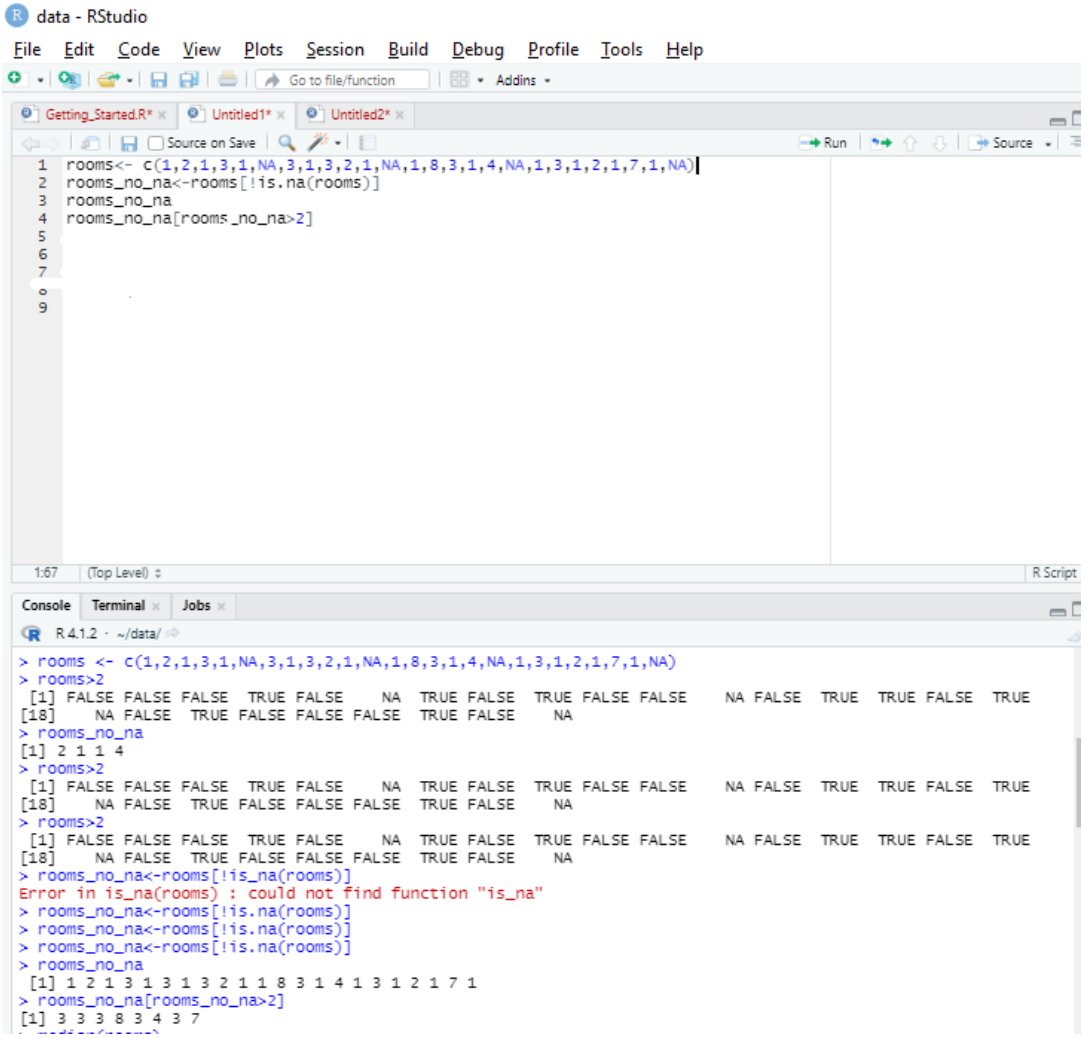
Adela Sobotkova
Digital Methods

ASSIGNMENT 3:

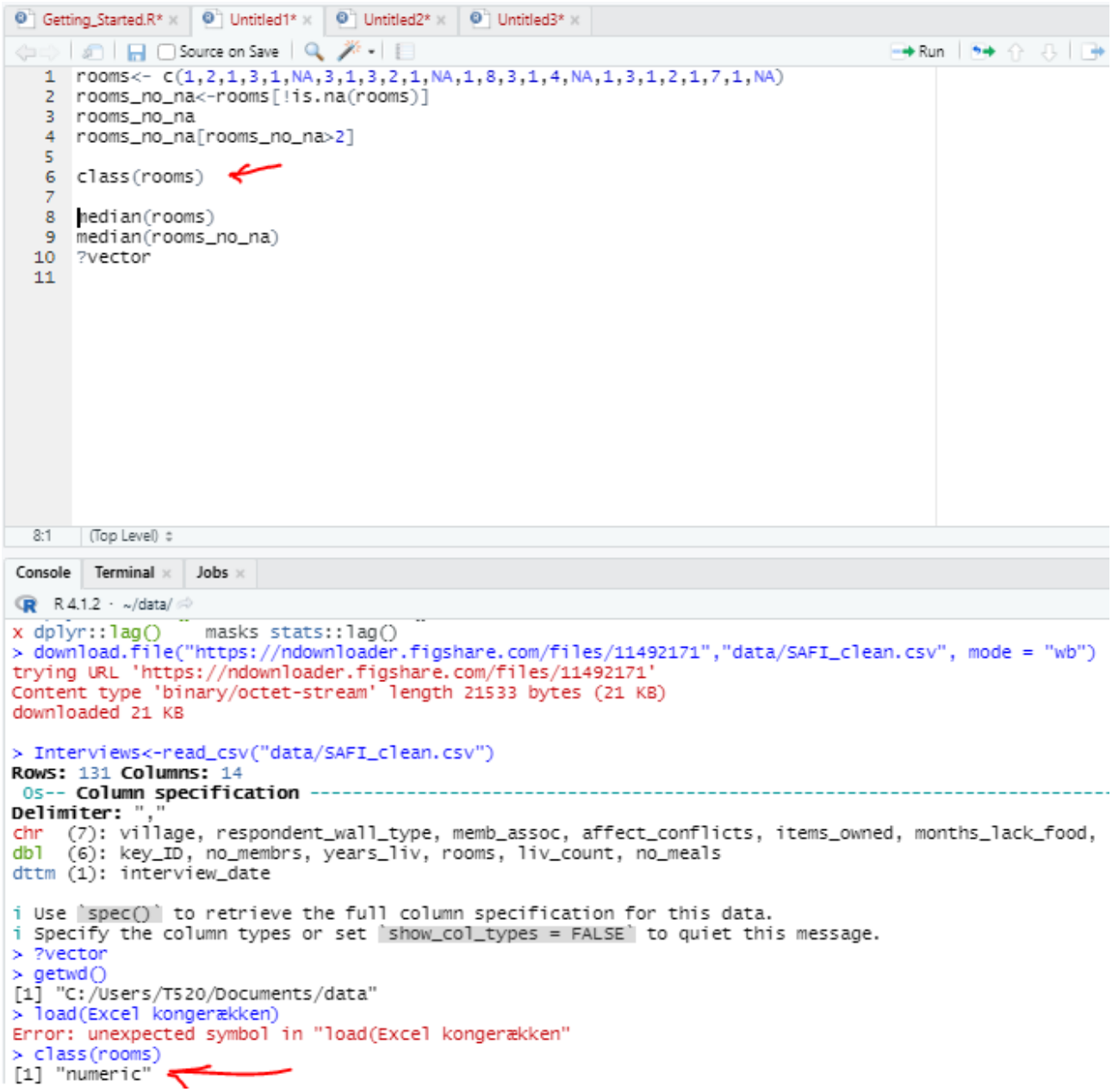
1.
To find the values in the "*rooms*" column of the *SAFI_Clean.CSV* document, I first typed the values of the "rooms" column:
rooms <- c(1, 2, 1, 3, 1, NA, 3, 1, 3, 2, 1, NA, 1, 8, 3, 1, 4, NA, 1, 3, 1, 2, 1, 7, 1, NA).
I then asked R, to show the ones, whose value is greater than 2, by writing the following code:
rooms>2.
It will then show me the following: "*FALSE FALSE FALSE TRUE FALSE NA TRUE FALSE TRUE FALSE FALSE NA FALSE TRUE TRUE FALSE TRUE NA FALSE TRUE FALSE FALSE FALSE TRUE FALSE NA*" This shows the individual numbers and if they are greater (TRUE) or lower (FALSE) than 2. "NA" is still showing.

By using the following code: **rooms_no_na<-rooms[!is.na(rooms)]** we tell R to filter the "NA's" away from the "rooms" column. It does so by creating a new list called "*rooms_no_na*"

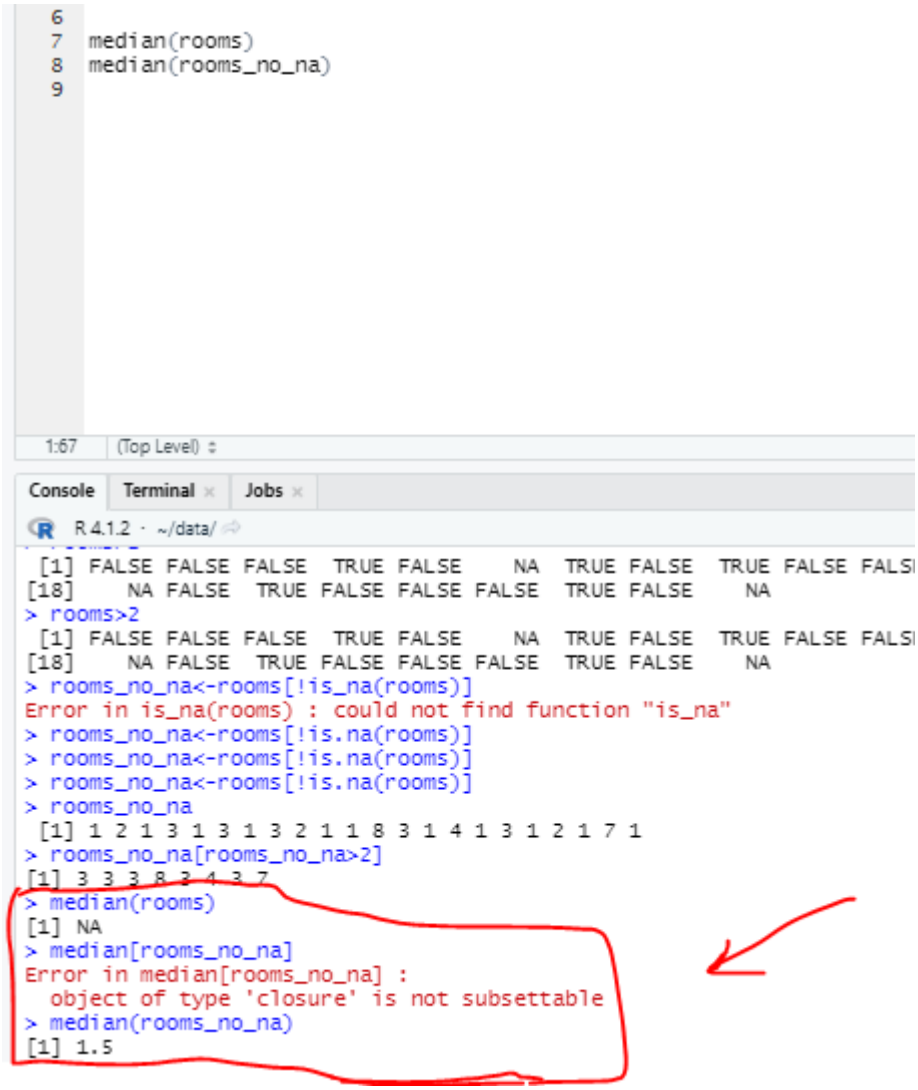
So now if we use the "*rooms_no_na*" it shows the "rooms" but without the "NA" answers:
1 2 1 3 1 3 1 3 2 1 1 8 3 1 4 1 3 1 2 1 7 1.
And then for it to exclusively show the numbers greater than 2, we can use the following code:
rooms_no_na[rooms_no_na>2].
And then it will show us the final result: *3 3 3 8 3 4 3 7*



2. The type of data in the ”rooms” vectors are numerical values
You can just the following code in R “**class(rooms)**”, it will thereby show you what kind of vectors you get in R:



3. By using the median function on the “rooms” column in R (**median(rooms)**), we get the median number. It is “NA”, but if we use the new list we created, the “*rooms_no_na*” we get “1.5” as the median.



4. https://github.com/Digital-Methods-HASS/Au641294_Hansen_Andreas

Andreas Tang Hansen
AU641294

2nd Weekly Peergrade assignment
Submitted 16-11-2021

Adela Sobotkova
Digital Methods

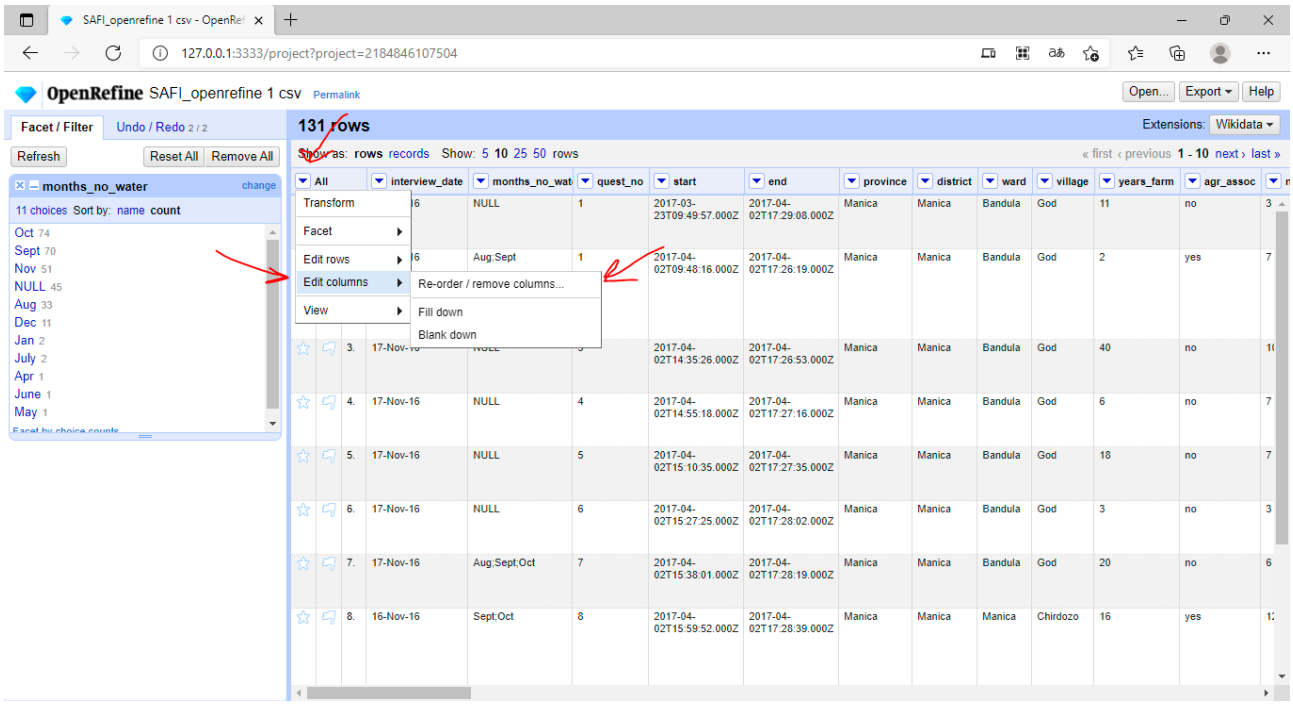
ASSIGNMENT 2:

1. Check the attached file
Source used: https://danmarkshistorien.dk/leksikon-og-kilder/vis/materiale/kongeraekken/

2. No, it does not. However, it is possible to export the file through the export option within Openrefine.

3.

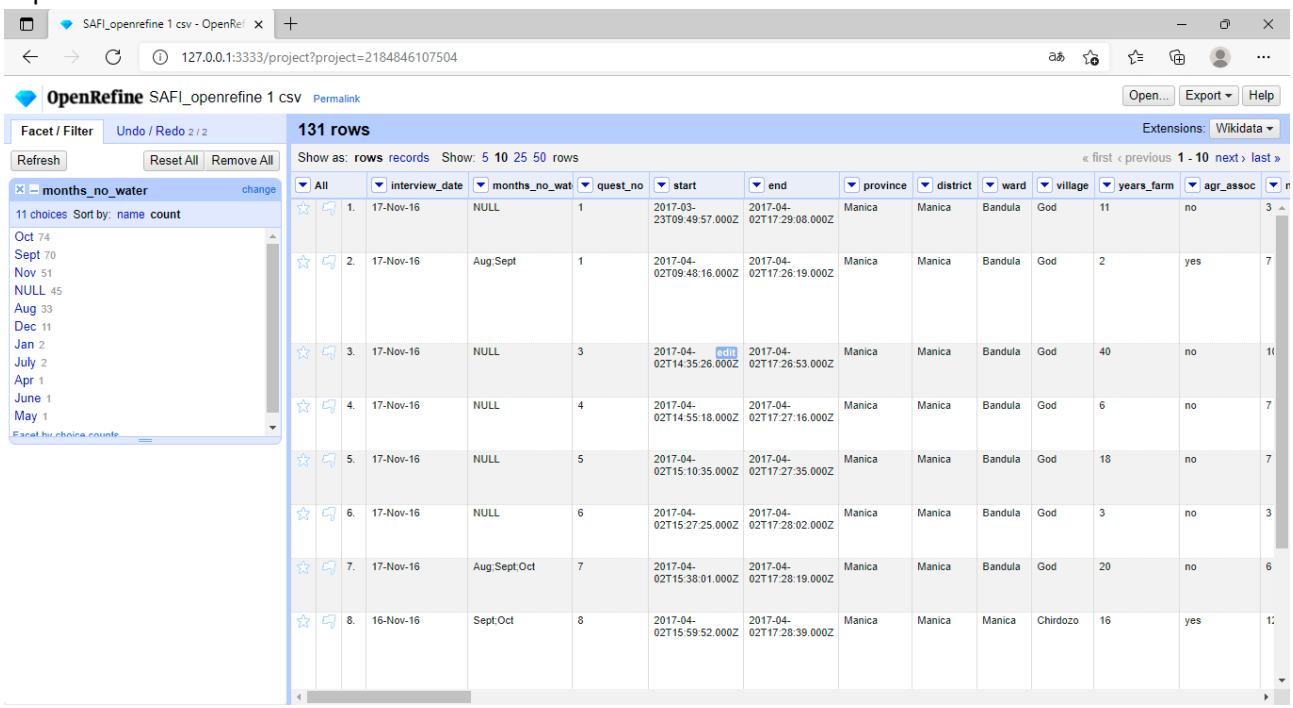
For simplicity sake I moved the column “Months_no_water” to the front by clicking on the *All* column and under the *Edit Columns* I then went into the section called *Re-order/remove columns*, from where you can rearrange the order.



The first I used the following command within *Transform*:
value.replace("[","").replace("]","").replace("'","").replace(" ","") by doing this I first tell the database to replace or filtrate various spaces between the months here amongst the quotes, both double and single, commas, bracets and spaces.

The Second command was**: value.split(";")** in the *custom text faucet*. By doing this I separate the months from each other and remove the “” between the months, allowing the database to locate and count the months. By doing this I can now determine, that the two driest months of the year were October and

September as shown underneath:



Peergrade Assignment 4 - Make Data Move

05/10/2020

Explore global development with R

Today, you will load a filtered gapminder dataset - with a subset of data on global development from 1952 - 2007 in increments of 5 years - to capture the period between the Second World War and the Global Financial Crisis.

Your task: Explore the data and visualise it in both static and animated ways, providing answers and solutions to 7 questions/tasks below.

Get the necessary packages

First, start with installing the relevant packages 'tidyverse', 'gganimate', and 'gapminder'.

```
## -- 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   2.1.0     v forcats 0.5.1

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

Look at the data and tackle the tasks

First, see which specific years are actually represented in the dataset and what variables are being recorded for each country. Note that when you run the cell below, Rmarkdown will give you two results - one for each line - that you can flip between.

```
data(gapminder)
str(gapminder)

## tibble [1,704 x 6] (S3: tbl_df/tbl/data.frame)
##  $ country  : Factor w/ 142 levels "Afghanistan",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ continent: Factor w/ 5 levels "Africa","Americas",..: 3 3 3 3 3 3 3 3 3 3
...
##  $ year     : int [1:1704] 1952 1957 1962 1967 1972 1977 1982 1987 1992 1997
...
```

```
##  $ lifeExp  : num [1:1704] 28.8 30.3 32 34 36.1 ...
##  $ pop      : int [1:1704] 8425333 9240934 10267083 11537966 13079460 14880372
12881816 13867957 16317921 22227415 ...
##  $ gdpPercap: num [1:1704] 779 821 853 836 740 ...
```

```
unique(gapminder$year)
```

```
##  [1] 1952 1957 1962 1967 1972 1977 1982 1987 1992 1997 2002 2007
```

```
head(gapminder)
```

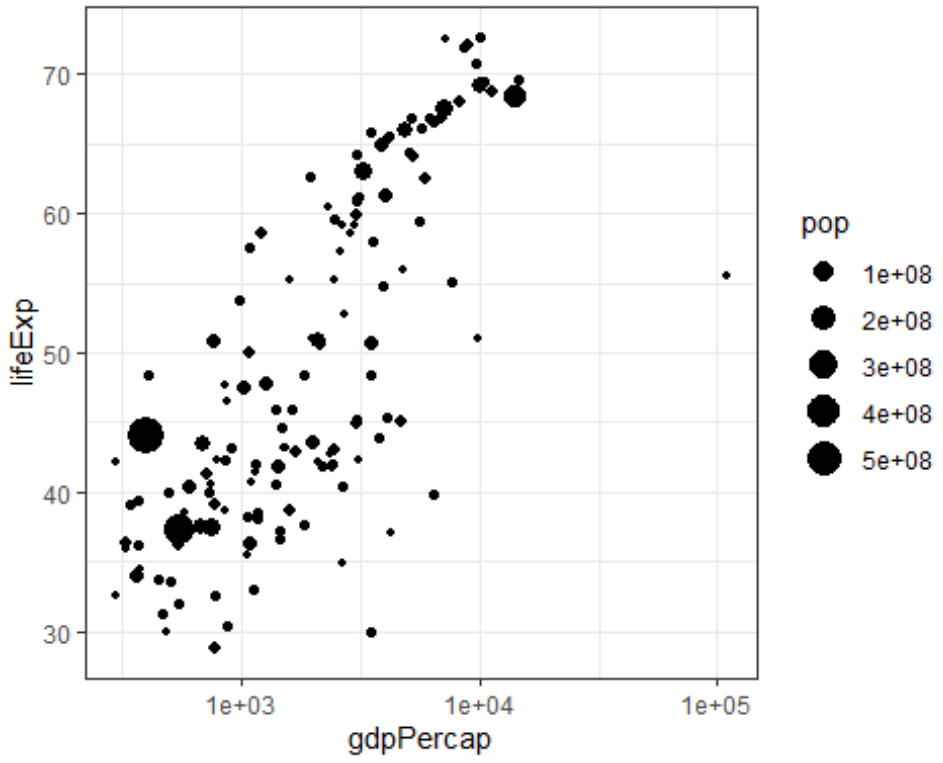
```
## # A tibble: 6 x 6
##   country     continent  year lifeExp      pop gdpPercap
##   <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
## 1 Afghanistan Asia       1952    28.8  8425333      779.
## 2 Afghanistan Asia       1957    30.3  9240934      821.
## 3 Afghanistan Asia       1962    32.0 10267083      853.
## 4 Afghanistan Asia       1967    34.0 11537966      836.
## 5 Afghanistan Asia       1972    36.1 13079460      740.
## 6 Afghanistan Asia       1977    38.4 14880372      786.
```

The dataset contains information on each country in the sampled year, its continent, life expectancy, population, and GDP per capita.

Let's plot all the countries in 1952.

```
theme_set(theme_bw())  # set theme to white background for better visibility

ggplot(subset(gapminder, year == 1952), aes(gdpPercap, lifeExp, size = pop)) +
  geom_point() +
  scale_x_log10()
```



...

We see an interesting spread with an outlier to the right. Answer the following questions, please:

1. *Why does it make sense to have a log10 scale on x axis?* # Answer= Because it is the default option for Rstudios/Rmarkdown

2. *Who is the outlier (the richest country in 1952 - far right on x axis)?* # Answer= Kuwait,see how we did below, we opened "gapminder" than piped it, so we could filter by year 1952. After that we asked R to arrange it by "gdpPercap" in descending order

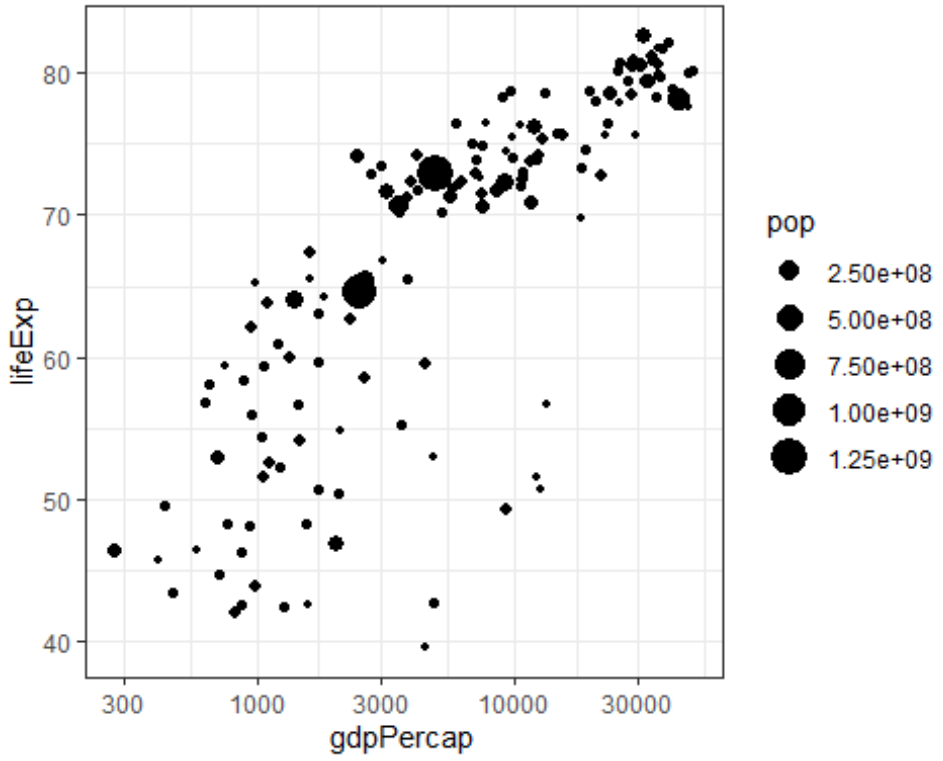
```
gapminder %>%
  filter(year == 1952) %>%
  arrange(desc(gdpPercap))
```

```
## # A tibble: 142 x 6
##    country        continent  year lifeExp       pop gdpPercap
##    <fct>          <fct>     <int>   <dbl>     <int>     <dbl>
##  1 Kuwait         Asia       1952    55.6    160000   108382.
##  2 Switzerland    Europe     1952    69.6   4815000    14734.
##  3 United States  Americas   1952    68.4 157553000    13990.
##  4 Canada         Americas   1952    68.8  14785584    11367.
##  5 New Zealand    Oceania    1952    69.4   1994794    10557.
##  6 Norway         Europe     1952    72.7   3327728    10095.
##  7 Australia      Oceania    1952    69.1   8691212    10040.
```

```
##  8 United Kingdom Europe     1952    69.2  50430000     9980.
##  9 Bahrain        Asia       1952    50.9    120447     9867.
## 10 Denmark        Europe     1952    70.8   4334000     9692.
## # ... with 132 more rows
```

Next, you can generate a similar plot for 2007 and compare the differences

```
ggplot(subset(gapminder, year == 2007), aes(gdpPercap, lifeExp, size = pop)) +
  geom_point() +
  scale_x_log10()
```



…

The black bubbles are a bit hard to read, the comparison would be easier with a bit more visual differentiation.

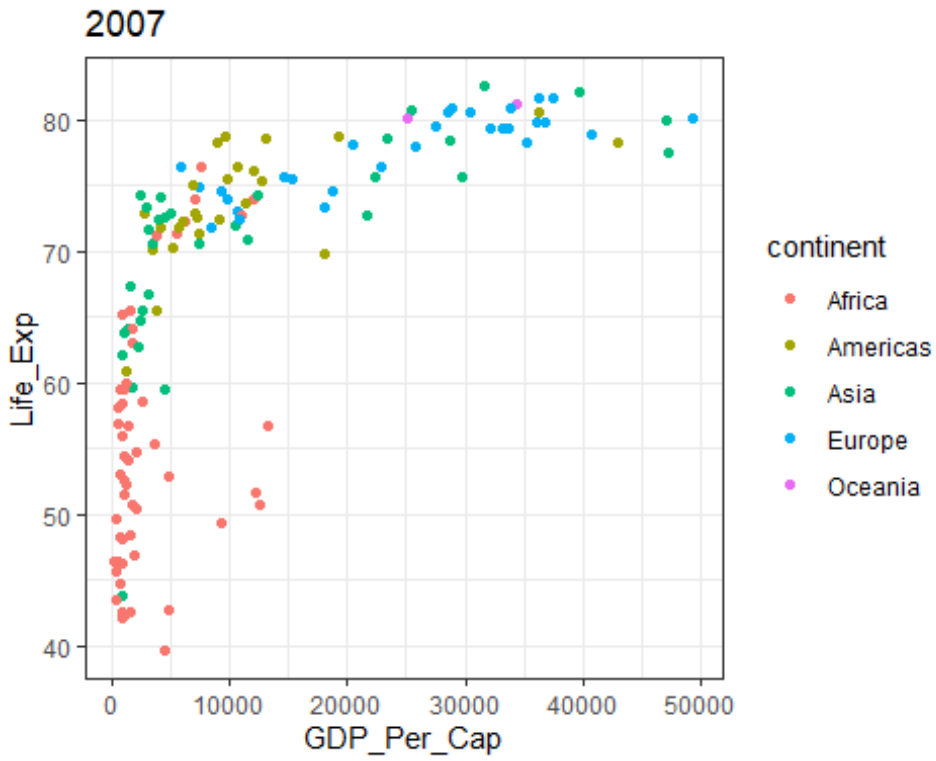
Tasks:

3. *Differentiate the **continents** by color, and fix the axis labels and units to be more legible (**Hint**: the 2.50e+08 is so called "scientific notation", which you might want to eliminate)* # Answer= To arrange it by color, we copied the code above from line 72 to 76, than added the following (labs(title,fill, x & y)) and edited what they meant. We also deleted pop= size. You can see it in line 96 to 105
4. *What are the five richest countries in the world in 2007?* # Answer= Norway, Kuwait, Singapore, United States and Ireland, We did the same as in Question 2.

```r
gapminder %>%
  filter(year == 2007) %>%
  arrange(desc(gdpPercap))
```

```
## # A tibble: 142 x 6
##    country          continent  year lifeExp       pop gdpPercap
##    <fct>            <fct>     <int>   <dbl>     <int>     <dbl>
##  1 Norway           Europe     2007    80.2   4627926    49357.
##  2 Kuwait           Asia       2007    77.6   2505559    47307.
##  3 Singapore        Asia       2007    80.0   4553009    47143.
##  4 United States    Americas   2007    78.2 301139947    42952.
##  5 Ireland          Europe     2007    78.9   4109086    40676.
##  6 Hong Kong, China Asia       2007    82.2   6980412    39725.
##  7 Switzerland      Europe     2007    81.7   7554661    37506.
##  8 Netherlands      Europe     2007    79.8  16570613    36798.
##  9 Canada           Americas   2007    80.7  33390141    36319.
## 10 Iceland          Europe     2007    81.8    301931    36181.
## # ... with 132 more rows
```

```r
ggplot(subset(gapminder, year == 2007), aes(gdpPercap, lifeExp, color=continent))
+
  geom_point() +
   labs(title = "2007",
         fill = "Continent",
         x = "GDP_Per_Cap",
         y = "Life_Exp")
```




```
  scale_x_log10()

## <ScaleContinuousPosition>
##  Range:
##  Limits:    0 --    1
```

Make it move!

The comparison would be easier if we had the two graphs together, animated. We have a lovely tool in R to do this: the `gganimate` package. Beware that there may be other packages your operating system needs in order to glue interim images into an animation or video. Read the messages when installing the package.

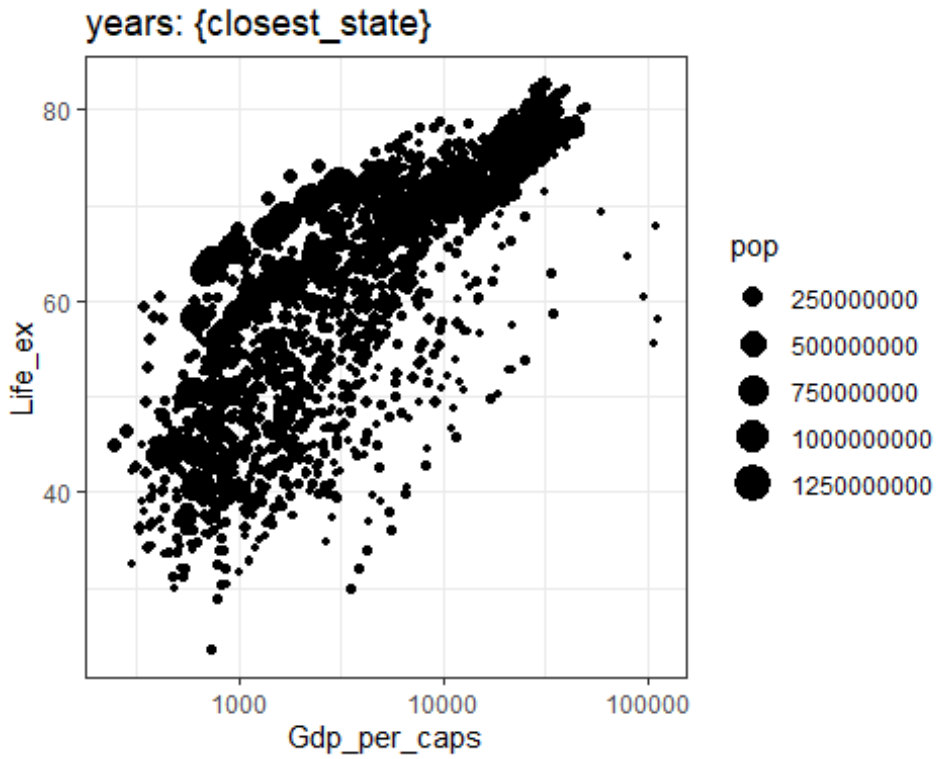
Also, there are *two* ways of animating the gapminder ggplot.

Option 1: Animate using transition_states()

The first step is to create the object-to-be-animated

```
options(scipen = 999)
#The commmand above removes scientific notations
anim <- ggplot(gapminder, aes(gdpPercap, lifeExp, size = pop)) +
  geom_point() +
  scale_x_log10()+ labs(title = 'years: {closest_state}') +
  labs(fill = "population",
```

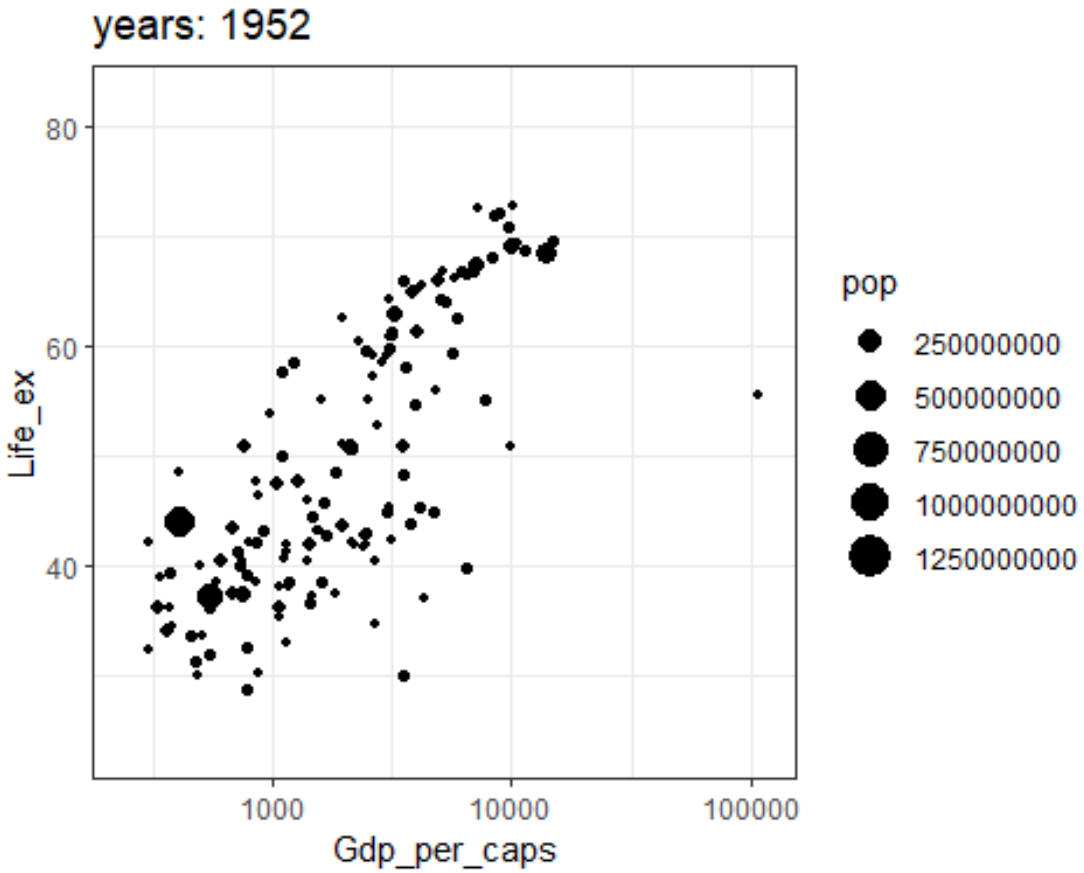
```
       x = "Gdp_per_caps",
       y = "Life_ex")
anim
```



...

This plot collates all the points across time. The next step is to split it into years and animate it. This may take some time, depending on the processing power of your computer (and other things you are asking it to do). Beware that the animation might appear in the bottom right 'Viewer' pane, not in this rmd preview. You need to `knit` the document to get the visual inside an html file.

```
anim + transition_states(year,
                         transition_length = 1,
                         state_length = 1)
```



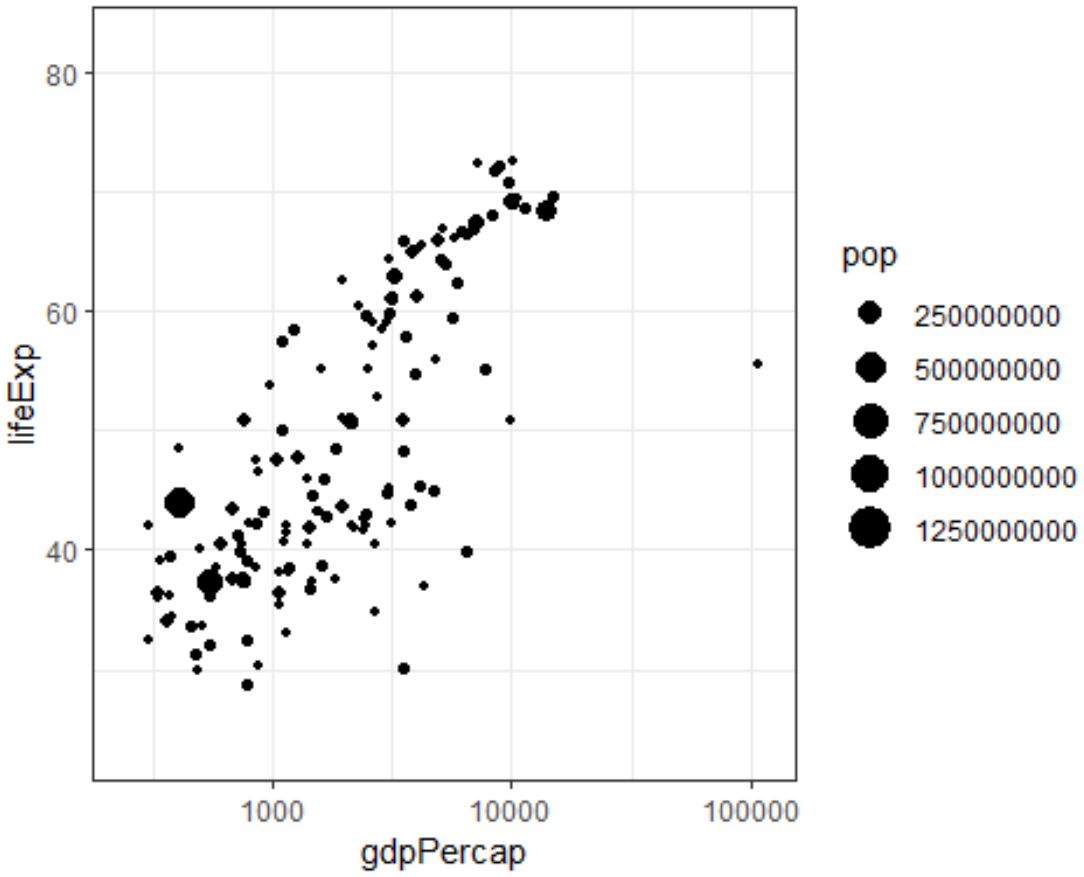
...

Notice how the animation moves jerkily, 'jumping' from one year to the next 12 times in total. This is a bit clunky, which is why it's good we have another option.

Option 2 Animate using transition_time()

This option smoothes the transition between different 'frames', because it interpolates and adds transitional years where there are gaps in the timeseries data.

```
anim2 <- ggplot(gapminder, aes(gdpPercap, lifeExp, size = pop)) +
  geom_point() +
  scale_x_log10() + # convert x to log scale
  transition_time(year)
anim2
```



The much smoother movement in Option 2 will be much more noticeable if you add a title to the chart, that will page through the years corresponding to each frame.

Now, choose one of the animation options and get it to work. You may need to troubleshoot your installation of `gganimate` and other packages

5. *Can you add a title to one or both of the animations above that will change in sync with the animation? (**Hint**: search labeling for `transition_states()` and `transition_time()` functions respectively)*

Answer= By using the code in the animation section of option 1, we simply added (lab:years {closest_state}) which means that the title will show years, and change the closest state/title after each animation. Check line 119 to 125

6. *Can you made the axes' labels and units more readable? Consider expanding the abreviated lables as well as the scientific notation in the legend and x axis to whole numbers.*

Answer= We did the same as in question 3, and in the 119 to 125 lines we inserted what we did in question 3 and edited the labs. The scipen=999 commands means that the scientific notations are removed and the pop will now be shown as actual numbers.

7. *Come up with a question you want to answer using the gapminder data and write it down. Then, create a data visualisation that answers the question and explain how your visualization answers the question. (Example: you wish to see what was mean life expectancy across the continents in the year you were born versus your parents' birth years). [Hint: if you wish to have more data than is in the filtered gapminder, you can load either the `gapminder_unfiltered` dataset and download more at https://www.gapminder.org/data/]*

Answer= I asked the question "What was the 10 richest countries in 1997"? The answer is: Norway, Kuwait, United States, Singapore, Switzerland, Netherlands, Denmark, Austria, Canada, Japan

```
gapminder %>%
  filter(year == 1997) %>%
  arrange(desc(gdpPercap))
```

```
## # A tibble: 142 x 6
##    country       continent  year lifeExp       pop gdpPercap
##    <fct>         <fct>     <int>   <dbl>     <int>     <dbl>
##  1 Norway        Europe     1997    78.3   4405672    41283.
##  2 Kuwait        Asia       1997    76.2   1765345    40301.
##  3 United States Americas   1997    76.8 272911760    35767.
##  4 Singapore     Asia       1997    77.2   3802309    33519.
##  5 Switzerland   Europe     1997    79.4   7193761    32135.
##  6 Netherlands   Europe     1997    78.0  15604464    30246.
##  7 Denmark       Europe     1997    76.1   5283663    29804.
##  8 Austria       Europe     1997    77.5   8069876    29096.
##  9 Canada        Americas   1997    78.6  30305843    28955.
## 10 Japan         Asia       1997    80.7 125956499    28817.
## # ... with 132 more rows
```

Credits: Andreas Tang Hansen, Anders Bergmann Rostermund, Martin Butzbach, Erik Luis Lanuza Oehlerich

1. **Changing the date format from "dd-mm-yyyy" to "yyyy-mm-dd"**

https://regex101.com/r/GzecCQ/1

2.

Changing "StopwordlistVoyant" into the "StopwordlistR" format, to be separated by " and , characters

https://regex101.com/r/MHJYem/1

Changing "StopwordlistR" into the "StopwordlistVoyant" format, by removing the " and , characters and starting a newline for each word.

https://regex101.com/r/aLUcYI/1

3. About data management in a spreadsheet:

Among the most important aspects of managing data in a spreadsheet is to be consistent. This includes using a consistent layout of files and data, consistent naming of identifiers and variables and consistent formats.

Another good idea, although not necessary is, when referring to dates, to use the year-month-date format. It is generally considered the most manageable format when working with dates.

As always with you work with data, it is highly advised to not only save frequently, but to also create a backup of the raw data or the spreadsheet itself, preferable both.

An important but frequent mistake that is made is, that many often leave an empty cell. This is a mistake that can often affect the entire data management process, because the computer has troubles understanding how to interpret the empty cell, or rather, the fact, that it cannot interpret the mistake.

Often times it will be beneficial to put just one word into the cell. An example could be to write "Danish soldier", but consider dividing them up and only writing one word pr. Cell. Possibly divide them into *occupation* and *nationality*, thereby; you have two cells with one word each. Naturally, it goes the same for applying units or measurements in the same cell, so for example, do not use: "20 Kg", instead divide them like in the previous example; *Unit* and *number* as an example.

Other great ideas include ideas such as; creating a data dictionary and saving a plain-text version of the data.