Dplyr tutorial: A demo using data from Great British Bake Off

Making the most of census microdata: An Introductory workshop

Ana Morales

This example will demonstrate the main functions of the package **dplyr**. We will be using data of the participants of the Great British Bake Off 2018. You will need to copy the R code and paste it on an R script. **R codes are coloured throughout the document.**

First we need to load dplyr into R

```
library(dplyr)
```

Let's make our own dataset

Here we are creating free variables (vectors in R). C**opy the following code and paste on to an R Studio Script:**

```
names<-c("Antony Amourdoux", "Briony Williams", "Dan Beasley-Harling","Imelda
McCarron", "Jon Jenkins", "Karen Wright","Kim-Joy","Luke Thompson","Manon Lag
rÃ¨ve", "Rahul Mandal", "Ruby Bhogal" ,"Terry Hartill")

sex<- c("male", "female","male", "female", "male", "female", "female","male",
"female", "male", "female", "male")

hometown<-c("London", "Bristol", "London", "County Tyrone", "Newport","Wakefi
eld", "Leeds", "Sheffield", "London", "Rotherham","London", "West Midlands")

occupation<- c("Banker", "Full-time parent", "Full-time parent", "Countryside
recreation officer", "Blood courier", "In-store sampling assistant", "Mental
health specialist","Civil servant/house and techno DJ", "Software project man
ager", "Research scientist", "Project manager", "Retired air steward")

age<- c(30, 33, 36, 33, 47, 60, 27, 30, 26, 30, 30, 56)
```

And now we put all the variables together in a dataframe

```
gbbo<-data.frame(names, age, sex, hometown, occupation)
```

Let’s start with the Demo

1. select()

Select the variables names and age

```
select(gbbo, names, age, sex)
```

```
##                   names age    sex
## 1      Antony Amourdoux  30   male
## 2       Briony Williams  33 female
## 3  Dan Beasley-Harling  36   male
## 4       Imelda McCarron  33 female
## 5           Jon Jenkins  47   male
## 6          Karen Wright  60 female
## 7               Kim-Joy  27 female
## 8         Luke Thompson  30   male
## 9        Manon LagrÃ¨ve  26 female
## 10         Rahul Mandal  30   male
## 11          Ruby Bhogal  30 female
## 12        Terry Hartill  56   male
```

You can store the selected variables into a new dataframe

```
gbbo_1<-select(gbbo, names, age, sex)
View(gbbo_1) # This allow you to see the dataset in a different t
ab
```

2. filter()

Now, filter (select) a subsample of gbbo participants younger than 30 and store them into a new dataframe

```
filter(gbbo_1, age<30)
```

```
##              names age    sex
## 1          Kim-Joy  27 female
## 2 Manon LagrÃ¨ve  26 female
```

```
gbbo_2<-filter(gbbo_1, age<30)
```

3. join()

This example, uses the join function to add more variables to the last dataset created.

```
left_join(gbbo_2, gbbo)

## Joining, by = c("names", "age", "sex")
```

```
##             names age    sex hometown                occupation
## 1         Kim-Joy  27 female    Leeds Mental health specialist
## 2 Manon LagrÃ¨ve  26 female   London Software project manager
```

dplyr automatically identifies the common variables and joins the dataset accordingly, but it is good practice to specify the variable(s) that indentifies the cases in both datasets; such as ID, names (if they are unique), etc. For example:

- Let's specify that our key variable is "names"

```
left_join(gbbo_2, gbbo, by = "names")
```

```
##             names age.x  sex.x age.y  sex.y hometown
## 1         Kim-Joy    27 female    27 female    Leeds
## 2 Manon LagrÃ¨ve    26 female    26 female   London
##                 occupation
## 1 Mental health specialist
## 2 Software project manager
```

- Let's use more that one key variable: names and age

```
left_join(gbbo_2, gbbo, by = c("names","age"))
```

```
##             names age  sex.x  sex.y hometown                occupation
## 1         Kim-Joy  27 female female    Leeds Mental health specialist
## 2 Manon LagrÃ¨ve  26 female female   London Software project manager
```

Now store this newly created dataset into a new one, under a different name.

Note: In this example we have been saving all the new datasets that we are creating. This is only done with the purpose of showing you the changes in the data after running the functions. But if you are working with big datasets, this is not a very good idea, since your R console will be populated with several datasets that are not being used.

```
gbbo_3<- left_join(gbbo_2, gbbo, by = c("names","age"))
```

4. rename()

Using rename function to change variable names. Here we are changing the variable "hometown" to "city"

```
rename(gbbo_3, "city"="hometown")
```

```
##             names age  sex.x  sex.y   city              occupation
## 1         Kim-Joy  27 female female  Leeds Mental health specialist
## 2 Manon LagrÃ¨ve  26 female female London Software project manager
```

5. summarise()

Here we will get some descriptive statistics using the summarise function from dplyr. This function is only for continuous variables.

```
summarise(gbbo, mean(age))
```

```
##   mean(age)
## 1      36.5
```

summarise is very handy since it also allows us to save the summarised variable, you just need to specify a name before the statistics asked

```
summarise(gbbo, mean_age=mean(age))
```

```
##   mean_age
## 1     36.5
```

you can ask for more than one statistic and store them all

```
summarise(gbbo, mean_age=mean(age),
                    st.dev_age= sd(age),
                    median_age=median(age))
```

```
##   mean_age st.dev_age median_age
## 1     36.5   11.42963       31.5
```

6. group_by()

This function works by grouping according to a variable

```
group_by(gbbo, sex)
```

```
## # A tibble: 12 x 5
## # Groups:   sex [2]
##    names                  age sex    hometown      occupation
##    <fct>                <dbl> <fct>  <fct>         <fct>
##  1 Antony Amourdoux        30 male   London        Banker
##  2 Briony Williams         33 female Bristol       Full-time parent
##  3 Dan Beasley-Harling     36 male   London        Full-time parent
##  4 Imelda McCarron         33 female County Tyrone Countryside recreation ~
##  5 Jon Jenkins             47 male   Newport       Blood courier
##  6 Karen Wright            60 female Wakefield     In-store sampling assis~
##  7 Kim-Joy                 27 female Leeds         Mental health specialist
##  8 Luke Thompson           30 male   Sheffield     Civil servant/house and~
##  9 Manon LagrÃ¨ve          26 female London        Software project manager
## 10 Rahul Mandal            30 male   Rotherham     Research scientist
## 11 Ruby Bhogal             30 female London        Project manager
## 12 Terry Hartill           56 male   West Midlands Retired air steward
```

As you can see, group_by does not seem to do anything. This is because it works in combination with other functions, for instance: summarise.

Let's save the group under a new dataset

```
bysex<-group_by(gbbo, sex)
```

And now, let's use this new dataset to get an indicator of the average age by sex of the participants.

```
summarise(bysex, age_mean=mean(age))
```

```
## # A tibble: 2 x 2
##   sex    age_mean
##   <fct>     <dbl>
## 1 female     34.8
## 2 male       38.2
```

We have created a new aggregated variable **age_mean** that takes the mean of the variable age according to sex.

Bonus (homework)

When you search for examples using **dplyr** on the web, you are very likely to encounter this symbol **%>%** called “pipe”. We are not covering this in this tutorial, but we will just give you and example of what it is and how to use it.

Pipes are meant to make the coding easy to write and read. It writes the code following a logical set of instructions. This is an example of the last code we used, but now rewritten with pipes

```
gbbo %>%                                  # Take the data
  group_by(sex) %>%                       # Now group it by sex
  summarise(age_mean=mean(age))           # Finally, we are creating our
# aggregated variable, all of this in one go!
```

```
## # A tibble: 2 x 2
##   sex    age_mean
##   <fct>     <dbl>
## 1 female     34.8
## 2 male       38.2
```

You can try to rewrite the previous codes using pipes!