AMAAS: Data Wrangling with dplyr

Jaap Saers (jpps2@cam.ac.uk) & Benjamin Utting (bju23@cam.ac.uk)

October 26, 2018

Analytical Methods in Anthropology and Archaeology Seminar (AMAAS)

Michaelmas 2018

The following packages are required for completing these exercises: *tidyverse*, *devtools*, *bioanth (available from GitHub)*

Introduction

This R Markdown document contains exercises to accompany a workshop for learning dplyr. Answers to these questions will be distributed online.

A First Look at our Dataset

Before we get started, we should do some preliminary data exploration.

```
#Descriptions of each variable:
?goldman

#Structure of our dataset:
str(goldman)

#Summary statistics of our dataset:
summary(goldman)
```

Now we can take a closer look at some of the variables.

```
#Where are the specimens housed?
summary(goldman$Inst)

#How many of each sex is represented?
summary(goldman$Sex)
#985 males and 543 females with 10 indeterminates (5 likely male, 5 likely female)
#Important to note that this sample is sex-biased!

#Where are these specimens from?
summary(goldman$Location)
```

Data Reorganization

This dataset contains specimens from all over the world. However, the column containing this information is organized poorly. Some specimens have three levels of location information (e.g. Aleutian Islands, Alaska, United States) whereas others only have one (e.g. Peru).

How do we address this issue?

```r
#Create a new dataset called "separated_location" with the "separate" function
separated_location <- goldman %>%
  separate(Location, into = c("Subregion", "Region", "Country"), sep = ",", fill = "left")
str(separated_location)

#This function returned "character" class data, so let's transform it back into
#factor data for consistency with the original dataset.
separated_location$Subregion <- as.factor(separated_location$Subregion)
separated_location$Region <- as.factor(separated_location$Region)
separated_location$Country <- as.factor(separated_location$Country)

#Explore data using "summary()" function
summary(separated_location$Subregion)
summary(separated_location$Region)
summary(separated_location$Country)
```

We also might want a slightly lower resolution of data. We have country names, but no column with continent names. Here, we can add a new column containing which continent each specimen is from.

```r
#Defining levels for each continent
EU <- c(" United Kingdom", "Austria", "Belgium", "France", "Germany", "Greenland",
        "Italy")
NAm <- c(" United States")
SAm <- c("Argentina", "Chile", "Ecuador", "Peru")
AFR <- c("Canary Islands", "Democratic Republic of the Congo", "Egypt", "Madagascar",
         "South Africa", "Sudan", "Tanzania")
ASIA <- c("Andaman Islands", "China", "Indonesia", "Japan", "Malaysia",
          "Philippine Islands", "Russia")
OCE <- c("Australia", "Papua New Guinea", "Solomon Islands", "Tasmania")
#Note the space before "United States" and "United Kingdom" - including this is
#necessary in order to properly complete this task.

#Using "mutate()" to create a new column and define conditions for each new variable:
separated_location <- separated_location %>%
    mutate(Continent = factor(case_when(Country %in% EU ~ "EU",
                                        Country %in% NAm ~ "NAm",
                                        Country %in% SAm ~ "SAm",
                                        Country %in% AFR ~ "AFR",
                                        Country %in% ASIA ~ "ASIA",
                                        Country %in% OCE ~ "OCE")))

#Take a look at what we have for each continent:
summary(separated_location$Continent)

#Finally, we can rearrange and define our new dataset.
separated_location <- separated_location %>%
  select(Inst, ID, Sex, Age, NOTE, Continent, Country, Region, Subregion, everything())
summary(separated_location)
```

Pipe Operators

Pipe operators (%>%) are much easier to read and write with compared to the base R style. To type them, press "ctrl + shift + m" on Windows, or "cmd + shift + m" on iOS.

Instead of nesting functions (reading from the inside to the outside), the idea of of piping is to read the functions from left to right.

Base R way of coding: show the top couple rows for separated_location with sex and age:

```
head(select(separated_location, Sex, Age))
```

With pipe operators: from (dataset) %>% (operation) %>% (what to show). this becomes very useful when compiling a sequence of operations.

```
separated_location %>%
  select(Sex, Age) %>%
  head
```

The following exercises should be completed with pipe operators.

We will use the separated_location dataset for all of the exercises.

For each exercise, name your new dataframe "newdata".

Exercise 1: select()

Create a new dataframe using select() containing country, sex, and age.

Exercise 2: select()

Create a new dataframe using select() containing all variables *except* NOTE and Inst.

Exercise 3: select()

Create a new dataframe using select() containing all columns between LHUM and OSCX.

Exercise 4: select()

Create a new dataframe using select() containing all columns with femoral head measurements (hint: all columns contain the string "FHD").

Exercise 5: filter()

Filter the dataset so that it only contains individuals from the subregion Alaska with both humeri present.

Exercise 6: filter()

Create a new dataset with only males that are between 50 and 70 kilograms (AVG.FHD for weight calculated from femoral head diameter).

Exercise 7: filter()

Filter out rows that are not from Egypt or the United states using the %in% operator.

Exercise 8: select(), arrange()

Arrange separated_location dataset by continent, then by country. show the first 10 rows using the head() function.

Exercise 9: select(), arrange()

Same as above, except include AVG.FHD and lastly arrange the rows in the AVG.FHD column in a descending order. For this, use the function desc().

Exercise 10: select(), mutate()

We want to estimate body mass from the average of right and left humeral head diameters using a regression equation. Create a new column of estimated body mass using the following formula: BM = -17.555 + 1.803* humeral head diameter. Remember, we want the new column to be the average of left and right humeral head diameter.

Exercise 11: mutate(), summarise()

The summarise() function will create summary statistics for a given column in the data frame such as finding the mean. For example, to compute the average tibia length, create a new column using mutate() that averages the right and left tibia maximum length. Then, use summarise() and apply the mean() function to the new column and call the summary value AVG.TML.

Hint: there are some NA values in the new column, as not all right or left tibiae are present. Therefore, when calling mean() it returns NA. To address this, use mean(, na.rm=TRUE) to discount NA values. An alternative way would be to use filter() to remove individuals without a tibia length before summarising.

Exercise 12: mutate(), summarise()

You can return many summary statistics in one table. Using the same mean tibia length created in exercise 11, provide the mean, SD, min, max, median, mode, and total n of average tibia lengths.

Exercise 13: group_by(), mutate(), summarise()

The group_by() verb is an important function in dplyr. It's related to concept of "split-apply-combine".

We want to split the data frame by some variable (e.g. Continent), apply a function to the individual data frames, and then combine the output.

Let's do that: split the separated_location data frame by continent, then ask for the same summary statistics as above. We expect a set of summary statistics for each continent

Exercise 14: group_by(), mutate(), summarise()

Among recent humans brachial and crural indices are positively correlated with mean annual temperature. High indices are found in groups from warmer climates, and lower indices are found in groups from colder climates. Let's see if this is the case in our dataset.

Group the dataset by continent and summarise mean brachial and crural index of the LEFT HAND SIDE.

You'll have to calculate the anthropometric indices first:

- Brachial index: (max radius length*100) / maximum humerus length
- Crural index: (max tibia length*100) / maximum femur length