Code

MATH 118: Notes B

Wrangling data with dplyr: filter, select, arrange

Importing Data

In this class, we are going to be working with a dataset relating to the languages spoken at home by Canadian Residents. Many Indigenous peoples exist in Canada with their own languages and cultures. Sadly, colonization has led to the loss of many of these languages. This data is a subset of data collected during the 2016 census.

What is a .csv file?

How do we import it into R?

Hide

```
#can_lang.csv needs to be saved in the same directory as this file.
#can_lang <- read.csv("can_lang.csv")
```

Alternatively, you can download it directly from the internet. Github user `ttimbers` hosts this file to share with the public.

Hide

```
can_lang <- read.csv("https://raw.githubusercontent.com/ttimbers/canlang/master/inst/extdata/can_lang.csv")
```

Let's take a look at this data for a minute to see what information has been recorded.

Installing and Using Packages

Sometimes everything we need (data, functions, etc) are not available in base R. In R, expert users will package up useful things like data and functions into packages that be download and used.

First, you need to download the package from the right hand menu –> You only need to do this once.

In each new .Rmd document, you need to call any packages you want to use but adding the code `library(packagename)` inside an R chunk.

For example, in this class we will use the `tidyverse` package a lot.

Hide

```
library(tidyverse)
```

`dplyr`

There are actually many commonly used packages wrapped up inside one `tidyverse` package.

Today we are specifically going to be talking about the package `dplyr` which is useful to manipulating data sets.

`filter`

We can use the `filter` function to extract ***rows*** from the data that have a particular characteristic.

For example, we may be interested in only looking at only the languages in this dataset that are Aboriginal languages.

Hide

```
#start with the can_lang dataset, the pipe "%" means apply the action on the following line to the previous lin
e. In this case, pick out only the rows were the category variable is "Aboriginal languages"
can_lang  %>%
  filter(category == "Aboriginal languages")
```

```
## # A tibble: 67 × 6
##    category             language                  mothe…¹ most_…² most_…³ lang_…⁴
##    <chr>                <chr>                       <int>   <int>   <int>   <int>
##  1 Aboriginal languages Aboriginal languages, n…      590     235      30     665
##  2 Aboriginal languages Algonquian languages, n…      45      10       0     120
##  3 Aboriginal languages Algonquin                    1260     370      40    2480
##  4 Aboriginal languages Athabaskan languages, n…      50      10       0      85
##  5 Aboriginal languages Atikamekw                    6150    5465    1100    6645
##  6 Aboriginal languages Babine (Wetsuwet'en)          110      20      10     210
##  7 Aboriginal languages Beaver                        190      50       0     340
##  8 Aboriginal languages Blackfoot                    2815    1110      85    5645
##  9 Aboriginal languages Carrier                      1025     250      15    2100
## 10 Aboriginal languages Cayuga                         45      10      10     125
## # … with 57 more rows, and abbreviated variable names ¹mother_tongue,
## #   ²most_at_home, ³most_at_work, ⁴lang_known
```

Hide

```
##note the aboriginal languages is text/categorical and so quotation marks are needed.
##R doesn't care about whether they are double quotation marks (") or single ('). They work the same.
# If we don't assign it to an object, then it just prints out for us to see!
```

Hide

```
#oftentimes, we want to take our subset and give it a new name. This takes our subset and assigns it to a new da
taset called `aboriginal_lang`.
aboriginal_lang <- can_lang  %>%
  filter(category == "Aboriginal languages")

#Notice if you assign it to an object that it doesn't print out the contents.
# You'll see the new object in your environment on the top right --->
# If you click on the word `aboriginal languages` (not the blue play button) it will open the object so you can
see what is saved inside.
```

It can also be used with numeric criteria.

Suppose we want a list of all the languages in Canada that are spoken by less than 100 people as their mother tongue.

Hide

```
rare_lang <- can_lang  %>%
  filter(mother_tongue < 100)
```

The logical operators are given below:

Operator	Description
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
==	Equal to
!=	Not equal to
!x	Not x
x | y	x OR y
x & y	x AND y

select

select is used to extract only certain ***columns***. For example, perhaps we only want to print out a list names of the aboriginal languages (language column).

Hide

```
aboriginal_lang %>%
  select(language)
```

```
## # A tibble: 67 × 1
##    language
##    <chr>
##  1 Aboriginal languages, n.o.s.
##  2 Algonquian languages, n.i.e.
##  3 Algonquin
##  4 Athabaskan languages, n.i.e.
##  5 Atikamekw
##  6 Babine (Wetsuwet'en)
##  7 Beaver
##  8 Blackfoot
##  9 Carrier
## 10 Cayuga
## # … with 57 more rows
```

We can combine criteria together as well in one command with multiple pipes:

Hide

```
can_lang %>%
  filter(category == "Aboriginal languages") %>%
  select(language)
```

```
## # A tibble: 67 × 1
##    language
##    <chr>
##  1 Aboriginal languages, n.o.s.
##  2 Algonquian languages, n.i.e.
##  3 Algonquin
##  4 Athabaskan languages, n.i.e.
##  5 Atikamekw
##  6 Babine (Wetsuwet'en)
##  7 Beaver
##  8 Blackfoot
##  9 Carrier
## 10 Cayuga
## # … with 57 more rows
```

arrange

The `arrange` function allows us to order the rows of the data frame by the values of a particular column.

For example, arrange all the aboriginal languages in canada by from most to least spoken as mother tongue.

Hide

```
aboriginal_lang %>%
  arrange(desc(mother_tongue))
```

```
## # A tibble: 67 × 6
##    category             language          mother_tongue most_a…¹ most_…² lang_…³
##    <chr>                <chr>                     <int>    <int>   <int>   <int>
##  1 Aboriginal languages Cree, n.o.s.              64050    37950    7800   86115
##  2 Aboriginal languages Inuktitut                 35210    29230    8795   40620
##  3 Aboriginal languages Ojibway                   17885     6175     765   28580
##  4 Aboriginal languages Oji-Cree                  12855     7905    1080   15605
##  5 Aboriginal languages Dene                      10700     7710     770   13060
##  6 Aboriginal languages Montagnais (Innu)         10235     8585    2055   11445
##  7 Aboriginal languages Mi'kmaq                    6690     3565     915    9025
##  8 Aboriginal languages Atikamekw                  6150     5465    1100    6645
##  9 Aboriginal languages Plains Cree                3065     1345      95    5905
## 10 Aboriginal languages Stoney                     3025     1950     240    3675
## # … with 57 more rows, and abbreviated variable names ¹most_at_home,
## #   ²most_at_work, ³lang_known
```

Hide

```
#use arrange(variable) to go from least to most
#use arrange(desc(variable)) to go from most to least, arrange(-variable) also works
```

slice

The slice function will allow us to pick only a subset of the rows based on their numeric order (1st through last).

For example, if I want a list of the 10 most commonly spoken aboriginal languages.

Hide

```
aboriginal_lang %>%
  arrange(desc(mother_tongue)) %>%
  slice(1:10) %>%
  select(language, mother_tongue) #optional
```

```
## # A tibble: 10 × 2
##    language          mother_tongue
##    <chr>                     <int>
##  1 Cree, n.o.s.              64050
##  2 Inuktitut                 35210
##  3 Ojibway                   17885
##  4 Oji-Cree                  12855
##  5 Dene                      10700
##  6 Montagnais (Innu)         10235
##  7 Mi'kmaq                    6690
##  8 Atikamekw                  6150
##  9 Plains Cree                3065
## 10 Stoney                     3025
```

Badges Earned

Badges in Progress

Brain Break

- Students at Allison Bernard Memorial High School in Eskasoni, Cape Breton recorded Paul McCartney's Blackbird in their native Mi'kmaq language. (https://www.youtube.com/watch?v=99-LoEkAA3w)
- The Jerry Cans are a band from Iqaluit, Nunavut who combine traditional Inuit throat singing with folk music and country rock. Their music is largely written in Inuktitut (the indigineous language of the Inuit) (https://www.youtube.com/watch?v=wW0gpo2deKg)