Intro to Dataframes in R

Justin Millar

September 6, 2017

Reading CSV datafiles into R

We often store our data in comma seperated value (CSV) files, which can be read into R using the `read.csv()` function:

```
# Download example .csv file
download.file("https://ndownloader.figshare.com/files/2292169",
              "data/portal_data_joined.csv")

# Save into variable
surveys <- read.csv('data/portal_data_joined.csv')
```

Note: this code requires having a `data/` folder in your project

Functions for characterizing dataframe

We can run the name of the variable to view the dataframe, but often there will be too much information to display in the console

Here are some useful functions for characterizing a dataframe:

```
head(surveys)     # Top of dataframe
tail(surveys)     # Bottom of dataframe
dim(surveys)      # Dimensions
ncol(surveys)     # Number of columns
nrow(surveys)     # Number of rows
names(surveys)    # Column names
rownames(surveys) # Row names
str(surveys)      # Structure, with class, length, and content
summary(surveys)  # Summary statistics for each columns
```

Challenge Exercise

What type of vectors are each of the columns in the `surveys` dataframe?

Indexing and subsetting dataframes

Dataframes are also subsetted or with square brackets, expect we must specify rows then columns`[row,column]`:

```
surveys[1, 1]   # first element in the first column of the data frame (as a vector)
surveys[1, 6]   # first element in the 6th column (as a vector)
surveys[, 1]    # first column in the data frame (as a vector)
surveys[1]      # first column in the data frame (as a data.frame)
surveys[1:3, 7] # first three elements in the 7th column (as a vector)
surveys[3, ]    # the 3rd element for all columns (as a data.frame)
head_surveys <- surveys[1:6, ] # equivalent to head(surveys)
```

Use the `-` sign to exclude certain sections:

```
surveys[,-1]          # The whole data frame, except the first column
surveys[-c(7:34786),] # Equivalent to head(surveys)
```

Subsetting columns by name

Columns can be selected by name using the these operators:

```
surveys["species_id"]       # Result is a data.frame
surveys[, "species_id"]     # Result is a vector
surveys[["species_id"]]     # Result is a vector
surveys$species_id          # Result is a vector
```

Challenge Exercise

How many were collected in 1990?

Factors

Factors are used for storing categorical data, which are separated into **levels**:

```
sex <- factor(c("male", "female", "female", "male"))
levels(sex)
nlevels(sex)
```

We can rename the levels in a factor, either individually or all at once:

```
levels(sex)[1] <- "F"       # Change the first element
levels(sex) <- c("F", "M")  # Change all factors
```

Finally, we may want to convert factors to `char` or `numeric`:

```
as.character(sex)
f <- factor(c(1990, 1983, 1977, 1998, 1990))
as.numeric(levels(f))[f] # We want to use the levels in this case
```

Challenge Exercise

Create a new dataframe, `subset_survey`, that only contains records these `species_id`: RM, OL, and PP.

How many of each species are in each plot type?

Plots in base R

One of the main reasons to use R is creating graphics

The basic function for generating graphics is `plot()`

```
plot(x = surveys$weight, y = surveys$hindfoot_length)
plot(surveys$species_id)
```

Customizing plots

Plots can be customized by adding arguments to the function:

```
plot(x = surveys$weight, y = surveys$hindfoot_length,
     xlab = 'Weight (g)', ylab = 'Hindfoot length (mm)',
     main = "Weight vs. Footlength", col = 'blue')
```

Challenge Exercise

With the `subset_survey` dataframe, use the `plot()` function to display the number of each `sex`. Be sure all levels are correctly labelled.

Create a similar plot for the number of specimens caught in each year.

Plot Types

The `type =` argument in the `plot()` can be used to create different types of plots

```r
x <- seq(1,10,1)
y <- 2^x

plot(x, y) # Default is type = 'p'
plot(x, y, type = 'l')
plot(x, y, type = 'b')
plot(x, y, type = 'h')
plot(x, y, type = 'o')
```

Other types of plots

There are other functions for creating popular graphics

```
# Histograms
hist(surveys$weight)

# Boxplots
boxplot(surveys$weight ~ surveys$species_id)
```

Note the syntax for creating a boxplot, this can be read as plot `weight` BY `species_id`

Exercise Challenge

Come up with your own visualization for some aspect of this data. What does your graphic show?