Lesson #3 – **'Starting with data …'** https://datacarpentry.org/R-ecology-lesson/02-starting-with-data.html

Learning Objectives

- Load external data from a .csv file into a data frame
- Describe what a data frame is
- Summarize the contents of a data frame
- Use indexing to subset specific portions of data frames

PART 1 (in this lesson..)

- Describe what a factor is
- Convert between strings and factors
- Reorder and rename factors
- Change how character strings are handled in a data frame
- Format dates

Lesson #3 – **'Starting with data …'** Preliminary notes (some good practices)

Create a header for your R script to record some of the essential details about the project – sample/ template below:

```
##############################################################################
# Project owner(s): X Y
# Title/ occupation and affiliation: PhD student @ The University of Edinburgh
# Project theme/ brief: Data Carpentry script for R for Ecology Curriculum
# Project title: Lessons #3: 'Starting with data'/ PART 1
# Date/ time: 16-17 June 2020
# R script version:  v1.1
# Web: https://datacarpentry.org/R-ecology-lesson/02-starting-with-data.html
# Metadata/ data description path: R_project_location/documents
##############################################################################
```

Is always a good practice to create a *data dictionary*/ *data description* or more broadly, a *metadata file* by e.g. using a text editor or word processor and place the document into your '*document*' subfolder within your *working directory*. To create documentation about your work to further increase/ ensure transparency and reproducibility, below is a good reference point to start.

https://data.research.cornell.edu/content/readme

RESEARCH DATA MANAGEMENT SERVICE GROUP

Comprehensive Data Management Planning & Services

Guide to writing "readme" style metadata

Usually a good practice to clean your working environment (RStudio's *memory*)…

```
# It is usually a good practice to clean the working Environment in RStudio - below optional
actions:

# rm(list = ls()) # alternatively Ctrl+Shift+F10
# rm(list = ls(all.names = TRUE)) # will clear all objects includes hidden objects.
# gc() #free up memory and report the memory usage.
# 'Ctrl + L' # clearing the Console

# Also, checking if the working directory was set correctly by using the getwd() command.
```

Re-loading packages if necessary..

```
# Assume all the essential R packages were pre-installed earlier.
# Now we may only need to (re-)load them into RStudio (just in case)

library(gridExtra) # ggplot
library(hexbin) # ggplot
library(dbplyr) # R and databases
library(RSQLite) # R and databases
library(tidyverse) # lesson 3 and onward
# all of these are apart of tidyverse, but sometimes learners are unable
# to install tidyverse, and need to install them individually, so here they are
library(lubridate)
library(readr)
library(ggplot2)
library(dplyr)
library(magrittr)
library(tidyr)
```

Lesson #3 – **'Starting with data …'** - R's *data.frame* object (*the concept..*)

- Data frames are the *de facto* data structure for most tabular data, and what we use for statistics and plotting.

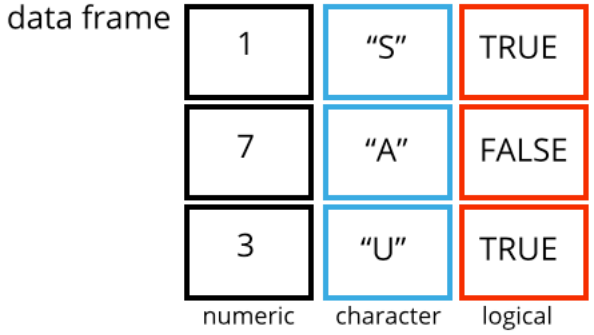
- A data frame can be created by hand, but most commonly they are generated by the functions **read.csv()** or **read.table()**; in other words, when *importing* spreadsheets from your hard drive (or the web).

- A data frame is the *representation of data in the format of a table* (or *2-dimensional array*) where the columns are vectors that all have the same length.

- Because *columns are vectors*, *each column must contain a single type of data* (e.g., characters, integers, factors or categorical values).

R's *tibble* is a modern, tidy edition of a base-R's data.frame object

https://cran.r-project.org/web/packages/tibble/vignettes/tibble.html



Lesson #3 – **'Starting with data …'** Load external data from a .csv file into a *data.frame*

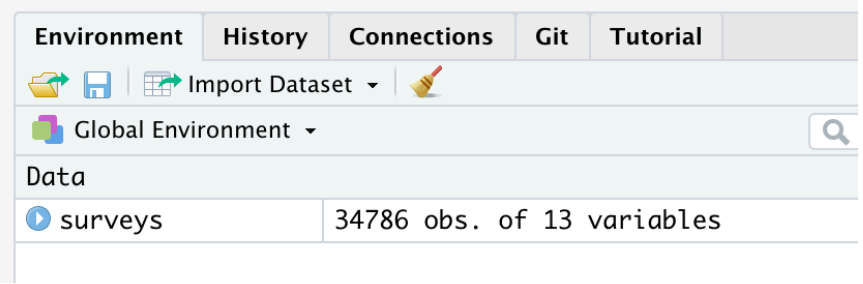
1. Download raw data file (*portal_data_joined.csv*)

```
download.file(url="https://ndownloader.figshare.com/files/2292169",
              destfile = "data_raw/portal_data_joined.csv")
```

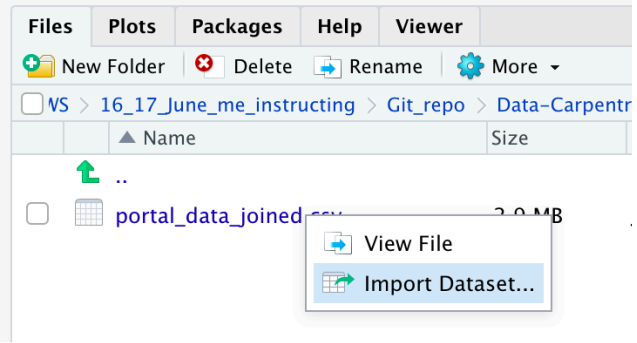
2. *Load .csv file* into an Rstudio's Environment as a *data.frame* object called '*surveys*'

```
surveys <- read.csv("data_raw/portal_data_joined.csv")
```

```
surveys <- read_csv('raw_data/portal_data_joined.csv')
```



Alternatively →



Column	Description
record_id	Unique id for the observation
month	month of observation
day	day of observation
year	year of observation
plot_id	ID of a particular plot
species_id	2-letter code
sex	sex of animal ("M", "F")
hindfoot_length	length of the hindfoot in mm
weight	weight of the animal in grams
genus	genus of animal
species	species of animal
taxon	e.g. Rodent, Reptile, Bird, Rabbit
plot_type	type of plot

```
> surveys <- read_csv('raw_data/portal_data_joined.csv')
Parsed with column specification:
cols(
  record_id = col_double(),
  month = col_double(),
  day = col_double(),
  year = col_double(),
  plot_id = col_double(),
  species_id = col_character(),
  sex = col_character(),
  hindfoot_length = col_double(),
  weight = col_double(),
  genus = col_character(),
  species = col_character(),
  taxa = col_character(),
  plot_type = col_character()
)
>
```

Lesson #3 – **'Starting with data …'** Load external data from a .csv file into a *data.frame (cont'd)*

3. Other ways of checking if the raw data loaded into RStudio's *Environment* are by typing the either of the following commands into either the *Console* or the *Source* (*script*) pane.

```
> surveys
# A tibble: 34,786 x 13
   record_id month   day  year plot_id species_id sex   hindfoot_length weight genus species taxa
       <dbl> <dbl> <dbl> <dbl>   <dbl> <chr>      <chr>           <dbl>  <dbl> <chr> <chr>   <chr>
 1         1     7    16  1977       2 NL         M                  32     NA Neot… albigu… Rode…
 2        72     8    19  1977       2 NL         M                  31     NA Neot… albigu… Rode…
 3       224     9    13  1977       2 NL         NA                 NA     NA Neot… albigu… Rode…
 4       266    10    16  1977       2 NL         NA                 NA     NA Neot… albigu… Rode…
 5       349    11    12  1977       2 NL         NA                 NA     NA Neot… albigu… Rode…
 6       363    11    12  1977       2 NL         NA                 NA     NA Neot… albigu… Rode…
 7       435    12    10  1977       2 NL         NA                 NA     NA Neot… albigu… Rode…
 8       506     1     8  1978       2 NL         NA                 NA     NA Neot… albigu… Rode…
 9       588     2    18  1978       2 NL         M                  NA    218 Neot… albigu… Rode…
```

```
> View(surveys)
```

surveys

Filter

	record_id	month	day	year	plot_id	species_id	sex	hindfoot_length	weight	genus	spe
1	1	7	16	1977	2	NL	M	32	NA	Neotoma	
2	72	8	19	1977	2	NL	M	31	NA	Neotoma	
3	224	9	13	1977	2	NL	NA	NA	NA	Neotoma	
4	266	10	16	1977	2	NL	NA	NA	NA	Neotoma	
5	349	11	12	1977	2	NL	NA	NA	NA	Neotoma	
6	363	11	12	1977	2	NL	NA	NA	NA	Neotoma	

Lesson #3 – **'Starting with data …'** Load data into a *data.frame (cont'd)*

```
> head(surveys)
# A tibble: 6 x 13
  record_id month   day  year plot_id species_id sex   hindfoot_length weight genus species taxa
      <dbl> <dbl> <dbl> <dbl>   <dbl> <chr>      <chr>           <dbl>  <dbl> <chr> <chr>   <chr>
1         1     7    16  1977       2 NL         M                  32     NA Neot… albigu… Rode…
2        72     8    19  1977       2 NL         M                  31     NA Neot… albigu… Rode…
3       224     9    13  1977       2 NL         NA                 NA     NA Neot… albigu… Rode…
4       266    10    16  1977       2 NL         NA                 NA     NA Neot… albigu… Rode…
5       349    11    12  1977       2 NL         NA                 NA     NA Neot… albigu… Rode…
6       363    11    12  1977       2 NL         NA                 NA     NA Neot… albigu… Rode…
# … with 1 more variable: plot_type <chr>
>
```

```
> tail(surveys)
# A tibble: 6 x 13
  record_id month   day  year plot_id species_id sex   hindfoot_length weight genus species taxa
      <dbl> <dbl> <dbl> <dbl>   <dbl> <chr>      <chr>           <dbl>  <dbl> <chr> <chr>   <chr>
1     26787     9    27  1997       7 PL         F                  21     16 Pero… leucop… Rode…
2     26966    10    25  1997       7 PL         M                  20     16 Pero… leucop… Rode…
3     27185    11    22  1997       7 PL         F                  21     22 Pero… leucop… Rode…
4     27792     5     2  1998       7 PL         F                  20      8 Pero… leucop… Rode…
5     28806    11    21  1998       7 PX         NA                 NA     NA Chae… sp.     Rode…
6     30986     7     1  2000       7 PX         NA                 NA     NA Chae… sp.     Rode…
# … with 1 more variable: plot_type <chr>
>
```

We can start inspecting the **str**ucture of our *data frame* with the function **str().**

```
> str(surveys)
tibble [34,786 × 13] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ record_id      : num [1:34786] 1 72 224 266 349 363 435 506 588 661 ...
 $ month          : num [1:34786] 7 8 9 10 11 11 12 1 2 3 ...
 $ day            : num [1:34786] 16 19 13 16 12 12 10 8 18 11 ...
 $ year           : num [1:34786] 1977 1977 1977 1977 1977 ...
 $ plot_id        : num [1:34786] 2 2 2 2 2 2 2 2 2 2 ...
 $ species_id     : chr [1:34786] "NL" "NL" "NL" "NL" ...
 $ sex            : chr [1:34786] "M" "M" NA NA ...
 $ hindfoot_length: num [1:34786] 32 31 NA NA NA NA NA NA NA NA ...
 $ weight         : num [1:34786] NA NA NA NA NA NA NA NA 218 NA ...
 $ genus          : chr [1:34786] "Neotoma" "Neotoma" "Neotoma" "Neotoma" ...
 $ species        : chr [1:34786] "albigula" "albigula" "albigula" "albigula" ...
 $ taxa           : chr [1:34786] "Rodent" "Rodent" "Rodent" "Rodent" ...
 $ plot_type      : chr [1:34786] "Control" "Control" "Control" "Control" ...
```

Inspecting *data.frame* objects (cont'd)

Below is a non-exhaustive list of functions to get a sense of the *content/ structure* of the data/ its data.frame (or tibble) representation. Let's try them out!

- Size:
  - `dim(surveys)` - returns a vector with the number of rows in the first element, and the number of columns as the second element (the **dim**ensions of the object)
  - `nrow(surveys)` - returns the number of rows
  - `ncol(surveys)` - returns the number of columns
- Content:
  - `head(surveys)` - shows the first 6 rows
  - `tail(surveys)` - shows the last 6 rows
- Names:
  - `names(surveys)` - returns the column names (synonym of `colnames()` for `data.frame` objects)
  - `rownames(surveys)` - returns the row names
- Summary:
  - `str(surveys)` - structure of the object and information about the class, length and content of each column
  - `summary(surveys)` - summary statistics for each column

NB: most of these functions are "generic", they can be used on other types of objects too!

Inspecting *data.frame* objects (cont'd)

```
> dim(surveys)
[1] 34786    13
> nrow(surveys)
[1] 34786
> ncol(surveys)
[1] 13
> names(surveys)
 [1] "record_id"       "month"           "day"             "year"            "plot_id"
 [6] "species_id"      "sex"             "hindfoot_length" "weight"          "genus"
[11] "species"         "taxa"            "plot_type"
> rownames(surveys)
   [1] "1"    "2"    "3"    "4"    "5"    "6"    "7"    "8"    "9"    "10"   "11"   "12"   "13"
  [14] "14"   "15"   "16"   "17"   "18"   "19"   "20"   "21"   "22"   "23"   "24"   "25"   "26"
  [27] "27"   "28"   "29"   "30"   "31"   "32"   "33"   "34"   "35"   "36"   "37"   "38"   "39"
  [40] "40"   "41"   "42"   "43"   "44"   "45"   "46"   "47"   "48"   "49"   "50"   "51"   "52"
  [53] "53"   "54"   "55"   "56"   "57"   "58"   "59"   "60"   "61"   "62"   "63"   "64"   "65"
  [66] "66"   "67"   "68"   "69"   "70"   "71"   "72"   "73"   "74"   "75"   "76"   "77"   "78"
  [79] "79"   "80"   "81"   "82"   "83"   "84"   "85"   "86"   "87"   "88"   "89"   "90"   "91"
```

```
> summary(surveys)
   record_id         month             day            year         plot_id       species_id
 Min.   :    1   Min.   : 1.000   Min.   : 1.0   Min.   :1977   Min.   : 1.00   Length:34786
 1st Qu.: 8964   1st Qu.: 4.000   1st Qu.: 9.0   1st Qu.:1984   1st Qu.: 5.00   Class :character
 Median :17762   Median : 6.000   Median :16.0   Median :1990   Median :11.00   Mode  :character
 Mean   :17804   Mean   : 6.474   Mean   :16.1   Mean   :1990   Mean   :11.34
 3rd Qu.:26655   3rd Qu.:10.000   3rd Qu.:23.0   3rd Qu.:1997   3rd Qu.:17.00
 Max.   :35548   Max.   :12.000   Max.   :31.0   Max.   :2002   Max.   :24.00

     sex            hindfoot_length     weight          genus              species
 Length:34786       Min.   : 2.00   Min.   :  4.00   Length:34786       Length:34786
 Class :character   1st Qu.:21.00   1st Qu.: 20.00   Class :character   Class :character
 Mode  :character   Median :32.00   Median : 37.00   Mode  :character   Mode  :character
                    Mean   :29.29   Mean   : 42.67
                    3rd Qu.:36.00   3rd Qu.: 48.00
                    Max.   :70.00   Max.   :280.00
                    NA's   :3348    NA's   :2503
     taxa            plot_type
 Length:34786       Length:34786
 Class :character   Class :character
 Mode  :character   Mode  :character
```

Challenge

Based on the output of `str(surveys)`, can you answer the following questions?

- What is the class of the object `surveys`?
- How many rows and how many columns are in this object?
- How many species have been recorded during these surveys?

Solution:

```
## * class: data frame
## * how many rows: 34786,  how many columns: 13
## * how many species: 48
```

Timer (mins): 02:00

Our survey data frame has rows and columns (it has 2 dimensions), if we want to extract some specific data from it, we need to specify the "*coordinates*" we want from it.

As per convention row numbers come first, followed by column numbers.

```
# first element in the first column of the data frame (as a vector)
surveys[1, 1]

# first element in the 6th column (as a vector)
surveys[1, 6]

# first column of the data frame (as a vector)
surveys[, 1]

# first column of the data frame (as a data.frame)
surveys[1]

# first three elements in the 7th column (as a vector)
surveys[1:3, 7]

# the 3rd row of the data frame (as a data.frame)
surveys[3, ]

# equivalent to head_surveys <- head(surveys)
head_surveys <- surveys[1:6, ]
```

NB: there are different ways of specifying these coordinates which lead to results with different classes.

Lesson #3 – **'Starting with data …'** Indexing and subsetting *data.frames (cont'd)*

```
> # first element in the first column of the data frame (as a vector)
> surveys[1, 1]
# A tibble: 1 x 1
  record_id
      <dbl>
1         1
> # first element in the 6th column (as a vector)
> surveys[1, 6]
# A tibble: 1 x 1
  species_id
  <chr>
1 NL
> # first column of the data frame (as a vector)
> surveys[, 1]
# A tibble: 34,786 x 1
   record_id
       <dbl>
 1         1
 2        72
 3       224
 4       266
 5       349
 6       363
 7       435
 8       506
 9       588
10       661
# … with 34,776 more rows
```

```
> # first column of the data frame (as a data.frame)
> surveys[1]
# A tibble: 34,786 x 1
   record_id
       <dbl>
 1         1
 2        72
 3       224
 4       266
 5       349
 6       363
 7       435
 8       506
 9       588
10       661
# … with 34,776 more rows
> # first three elements in the 7th column (as a vector)
> surveys[1:3, 7]
# A tibble: 3 x 1
  sex
  <chr>
1 M
2 M
3 NA
```

Lesson #3 – **'Starting with data …'** Indexing and subsetting *data.frames (cont'd)*

```
> # the 3rd row of the data frame (as a data.frame)
> surveys[3, ]
# A tibble: 1 x 13
  record_id month   day  year plot_id species_id sex   hindfoot_length weight genus species taxa
      <dbl> <dbl> <dbl> <dbl>   <dbl> <chr>      <chr>           <dbl>  <dbl> <chr> <chr>   <chr>
1       224     9    13  1977       2 NL         NA                 NA     NA Neot… albigu… Rode…
# … with 1 more variable: plot_type <chr>
> # equivalent to head_surveys <- head(surveys)
> head_surveys <- surveys[1:6, ]
```

DC_Ecology_Lesson_3_starting_wit… | head_surveys

Filter

	record_id	month	day	year	plot_id	species_id	sex	hindfoot_length	weight	genus	species	taxa	plot_type
1	1	7	16	1977	2	NL	M	32	NA	Neotoma	albigula	Rodent	Control
2	72	8	19	1977	2	NL	M	31	NA	Neotoma	albigula	Rodent	Control
3	224	9	13	1977	2	NL	NA	NA	NA	Neotoma	albigula	Rodent	Control
4	266	10	16	1977	2	NL	NA	NA	NA	Neotoma	albigula	Rodent	Control
5	349	11	12	1977	2	NL	NA	NA	NA	Neotoma	albigula	Rodent	Control
6	363	11	12	1977	2	NL	NA	NA	NA	Neotoma	albigula	Rodent	Control

Lesson #3 – **'Starting with data …'** Indexing and subsetting *data.frames (cont'd)*

: is a special function that creates numeric vectors of integers in *ascending* or *decreasing* order, test **1:10** and **10:1** for instance.

DC_Ecology_Lesson_3_starting_wit… * | head_surveys_asc

Filter

	record_id	month	day	year	plot_id	species_id	sex	hindfoot_length	weight	genus	species	taxa	plot_type
1	1	7	16	1977	2	NL	M	32	NA	Neotoma	albigula	Rodent	Control
2	72	8	19	1977	2	NL	M	31	NA	Neotoma	albigula	Rodent	Control
3	224	9	13	1977	2	NL	NA	NA	NA	Neotoma	albigula	Rodent	Control
4	266	10	16	1977	2	NL	NA	NA	NA	Neotoma	albigula	Rodent	Control
5	349	11	12	1977	2	NL	NA	NA	NA	Neotoma	albigula	Rodent	Control
6	363	11	12	1977	2	NL	NA	NA	NA	Neotoma	albigula	Rodent	Control
7	435	12	10	1977	2	NL	NA	NA	NA	Neotoma	albigula	Rodent	Control
8	506	1	8	1978	2	NL	NA	NA	NA	Neotoma	albigula	Rodent	Control
9	588	2	18	1978	2	NL	M	NA	218	Neotoma	albigula	Rodent	Control
10	661	3	11	1978	2	NL	NA	NA	NA	Neotoma	albigula	Rodent	Control

Showing 1 to 10 of 10 entries, 13 total columns

DC_Ecology_Lesson_3_starting_wit… * | head_surveys_asc | head_surveys_desc

Filter

	record_id	month	day	year	plot_id	species_id	sex	hindfoot_length	weight	genus	species	taxa	plot_type
1	661	3	11	1978	2	NL	NA	NA	NA	Neotoma	albigula	Rodent	Control
2	588	2	18	1978	2	NL	M	NA	218	Neotoma	albigula	Rodent	Control
3	506	1	8	1978	2	NL	NA	NA	NA	Neotoma	albigula	Rodent	Control
4	435	12	10	1977	2	NL	NA	NA	NA	Neotoma	albigula	Rodent	Control
5	363	11	12	1977	2	NL	NA	NA	NA	Neotoma	albigula	Rodent	Control
6	349	11	12	1977	2	NL	NA	NA	NA	Neotoma	albigula	Rodent	Control
7	266	10	16	1977	2	NL	NA	NA	NA	Neotoma	albigula	Rodent	Control
8	224	9	13	1977	2	NL	NA	NA	NA	Neotoma	albigula	Rodent	Control
9	72	8	19	1977	2	NL	M	31	NA	Neotoma	albigula	Rodent	Control
10	1	7	16	1977	2	NL	M	32	NA	Neotoma	albigula	Rodent	Control

Showing 1 to 10 of 10 entries, 13 total columns

One can also exclude certain indices of a data frame using the "**-**" sign, e.g.:

```
surveys[, -1]          # The whole data frame, except the first column

surveys[-c(7:34786), ] # Equivalent to head(surveys)
```

```
> surveys[-c(7:34786), ] # Equivalent to head(surveys)
# A tibble: 6 x 13
  record_id month   day  year plot_id species_id sex   hindfoot_length weight genus   species  taxa   plot_type
      <dbl> <dbl> <dbl> <dbl>   <dbl> <chr>      <chr>           <dbl>  <dbl> <chr>   <chr>    <chr>  <chr>
1         1     7    16  1977       2 NL         M                  32     NA Neotoma albigula Rodent Control
2        72     8    19  1977       2 NL         M                  31     NA Neotoma albigula Rodent Control
3       224     9    13  1977       2 NL         NA                 NA     NA Neotoma albigula Rodent Control
4       266    10    16  1977       2 NL         NA                 NA     NA Neotoma albigula Rodent Control
5       349    11    12  1977       2 NL         NA                 NA     NA Neotoma albigula Rodent Control
6       363    11    12  1977       2 NL         NA                 NA     NA Neotoma albigula Rodent Control
>
```

```
> surveys[, -1]          # The whole data frame, except the first column
# A tibble: 34,786 x 12
   month   day  year plot_id species_id sex   hindfoot_length weight genus   species  taxa   plot_type
   <dbl> <dbl> <dbl>   <dbl> <chr>      <chr>           <dbl>  <dbl> <chr>   <chr>    <chr>  <chr>
 1     7    16  1977       2 NL         M                  32     NA Neotoma albigula Rodent Control
 2     8    19  1977       2 NL         M                  31     NA Neotoma albigula Rodent Control
 3     9    13  1977       2 NL         NA                 NA     NA Neotoma albigula Rodent Control
 4    10    16  1977       2 NL         NA                 NA     NA Neotoma albigula Rodent Control
 5    11    12  1977       2 NL         NA                 NA     NA Neotoma albigula Rodent Control
 6    11    12  1977       2 NL         NA                 NA     NA Neotoma albigula Rodent Control
 7    12    10  1977       2 NL         NA                 NA     NA Neotoma albigula Rodent Control
 8     1     8  1978       2 NL         NA                 NA     NA Neotoma albigula Rodent Control
 9     2    18  1978       2 NL         M                  NA    218 Neotoma albigula Rodent Control
10     3    11  1978       2 NL         NA                 NA     NA Neotoma albigula Rodent Control
# … with 34,776 more rows
```

Subsetting by names: data frames can be subset by calling indices (as shown previously), but also by calling their column names directly:

```
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
```

```
> surveys["species_id"]       # Result is a data.frame
# A tibble: 34,786 x 1
   species_id
   <chr>
 1 NL
 2 NL
 3 NL
 4 NL
 5 NL
 6 NL
 7 NL
 8 NL
 9 NL
10 NL
# … with 34,776 more rows
> surveys[, "species_id"]     # Result is a vector
# A tibble: 34,786 x 1
   species_id
   <chr>
 1 NL
 2 NL
 3 NL
 4 NL
 5 NL
 6 NL
 7 NL
 8 NL
 9 NL
10 NL
# … with 34,776 more rows
```

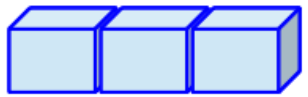
Lesson #3 – **'Starting with data …'** Indexing and subsetting *data.frames (cont'd)*

```
> surveys[["species_id"]]     # Result is a vector
  [1] "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL"
 [25] "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL"
 [49] "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL"
 [73] "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL"
 [97] "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL"
[121] "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL"
[145] "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL"
```

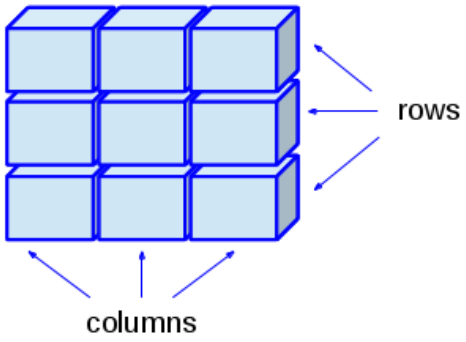
```
> surveys$species_id          # Result is a vector
  [1] "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL"
 [25] "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL"
 [49] "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL"
 [73] "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL"
 [97] "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL"
[121] "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL"
[145] "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL"
[169] "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL" "NL"
```

Lesson #3 – **'Starting with data …'** **Recap…**

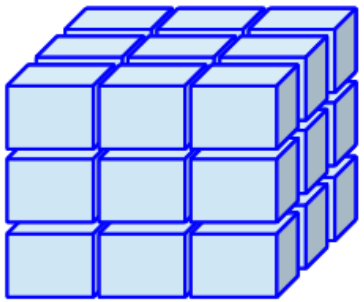
Vector



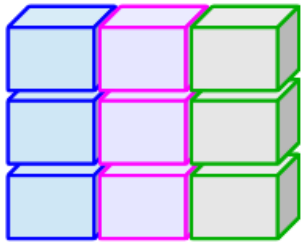
Matrix



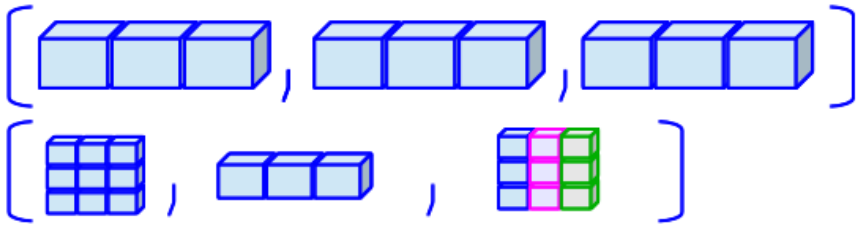
Array



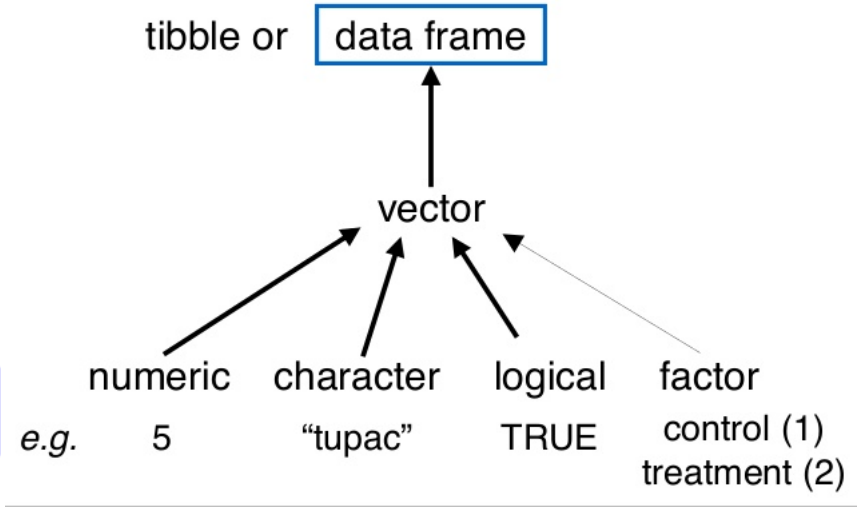
Data Frame
(Table)



Lists



R data structure flowchart



Next session: Factors

Challenge

1. Create a `data.frame` (`surveys_200`) containing only the data in row 200 of the `surveys` dataset.
2. Notice how `nrow()` gave you the number of rows in a `data.frame`?
   - Use that number to pull out just that last row in the data frame.
   - Compare that with what you see as the last row using `tail()` to make sure it's meeting expectations.
   - Pull out that last row using `nrow()` instead of the row number.
   - Create a new data frame (`surveys_last`) from that last row.
3. Use `nrow()` to extract the row that is in the middle of the data frame. Store the content of this row in an object named `surveys_middle`.
4. Combine `nrow()` with the `-` notation above to reproduce the behavior of `head(surveys)`, keeping just the first through 6th rows of the surveys dataset.

Timer (mins): 05:00

Solution:

```
## 1.
surveys_200 <- surveys[200, ]
## 2.
# Saving `n_rows` to improve readability and reduce duplication
n_rows <- nrow(surveys)
surveys_last <- surveys[n_rows, ]
## 3.
surveys_middle <- surveys[n_rows / 2, ]
## 4.
surveys_head <- surveys[-(7:n_rows), ]
```

Any questions for Lesson #3 – ‘Starting with data’?

Summary

What is a *data.frame*?
How can I read a complete csv file into R?
How can I get basic summary information about my dataset?
How can I change the way R treats strings in my dataset?

Your [*specific*] feedback is highly appreciated and is essential for our further improvement! Please leave your comments on Etherpad!