Module 1: Introduction to Data

Lecture 2: Data basics

Recap

- In our last lecture we were introduce to some definitions relating to data types.
- Let's test our knowledge of these terms in an example
- Make sure you have your iClickers ready to participate!

Observations, variables, and data matrices

- Last class we denoted n observations using the notation $x_1, x_2, \ldots, x_n$.
- For example, we may have surveyed n students to ask them their age to obtain $x_1 = 19$, $x_2 = 23$, $\ldots$, $x_{n-1} = 20$, $x_n = 23$
- In this case, each student was an *observational unit* for which a single *variable* was collected (age).
- A *variable* is simply an attribute or characteristic of the observational unit.

Observations, variables, and data matrices

- More generally, x_i may be a vector of a collection of multiple variables.
- For example, in addition to their age, we could have also asked each student to provide their gender and how many house do they sleep on average.
- In that case each *observation* would be a vector of three variables.
- Data of this type is often subscripted using two numbers (the first to denote the observation number, the second to denote the variable number)

Observations, variables, and data matrices

Data are often stored in a *data matrix* wherein observations make up the rows, and variables make up the columns

variable 1 ↓

Obs.	var 1	var 2	⋯	var p
1	x_{11}	x_{12}	⋯	x_{1p}
2	x_{21}	x_{22}	⋯	x_{2p}
⋮	⋮	⋮	⋮	⋮
i	x_{i1}	x_{i2}	⋯	x_{ip}
⋮	⋮	⋮	⋮	⋮
n	x_{n1}	x_{n2}	⋯	x_{np}

← *i^{th} observation*

Classroom survey

A survey was conducted on students in an introductory statistics course. Below are a few of the questions on the survey, and the corresponding variables the data from the responses were stored in:

- `gender`: What is your gender?
- `sleep`: How many hours do you sleep at night, on average?
- `intro_extra`: Do you consider yourself introverted or extraverted?
- `countries`: How many countries have you visited?
- `dread`: On a scale of 1-5, how much do you dread being here?

Data matrix

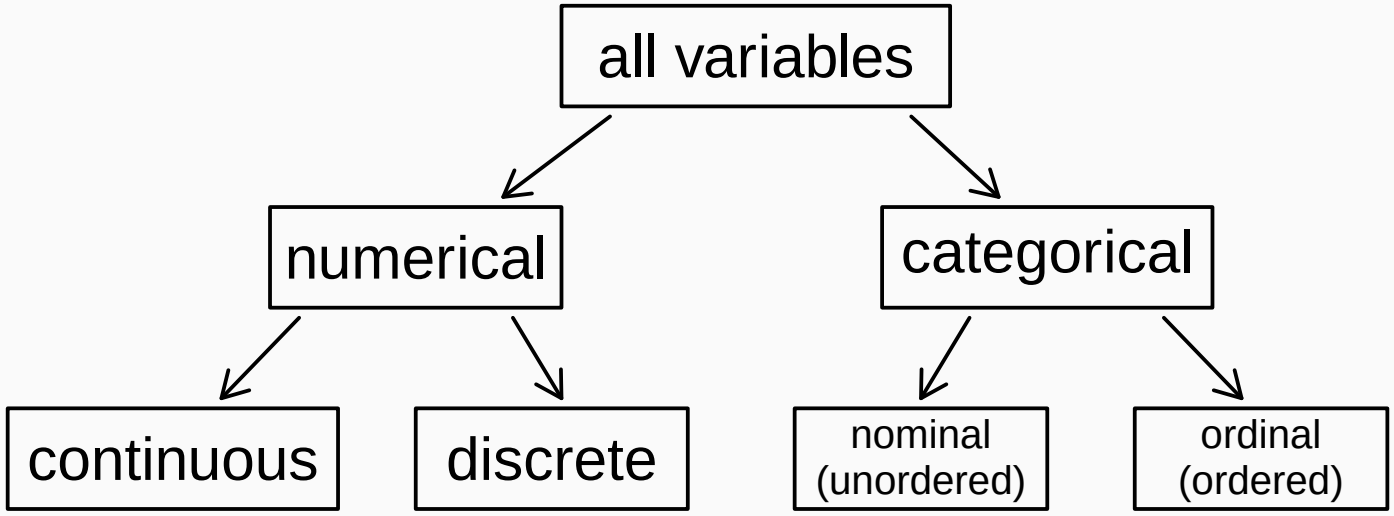
Data collected on students in a statistics class on a variety of variables:

variable ↓

Stu.	`gender`	`intro_extra`	$\cdots$	`dread`
1	male	extravert	$\cdots$	3
2	female	extravert	$\cdots$	2
3	female	introvert	$\cdots$	4
4	female	extravert	$\cdots$	2
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
86	male	extravert	$\cdots$	3

← *observation*

Types of variables



A special cas for the nominal (unordered categorical) is when there are only two categories. In that case, the variables is **binary**.

Types of variables (cont.)

	gender	sleep	intro_extra	countries	dread
1	male	5	extravert	3	3
2	female	7	extravert	7	2
3	female	5.5	introvert	1	4
4	male	7	extravert	2	2
5	other	3	introvert	1	3
6	female	3	extravert	9	4

A. numerical, continuous
B. numerical, discrete
C. categorical, nominal
D. categorical, binary
E. categorical, ordinal

- `gender`:
- `sleep`:
- `intro_extra`:
- `countries`:
- `dread`:

iClicker question

What type of variable is a telephone area code?

(a) numerical, continuous

(b) numerical, discrete

(c) categorical, nominal

(d) categorical, binary

(e) categorical, ordinal

Data frames

- In R, data matrices are stored in objects called *data frames*
- Like a data matrix, the rows can be treated as multivariate observational units, and the columns as variables (also referred to as *features*.
- These features can be different types: eg numeric or categorical
- There are a number of data sets already installed with R; see `data()`
- To load the data frame called `Orange`, for example, type:

```
data(Orange)
```

Data frames

- A description of this data set can be seen by typing:

```
?Orange
```

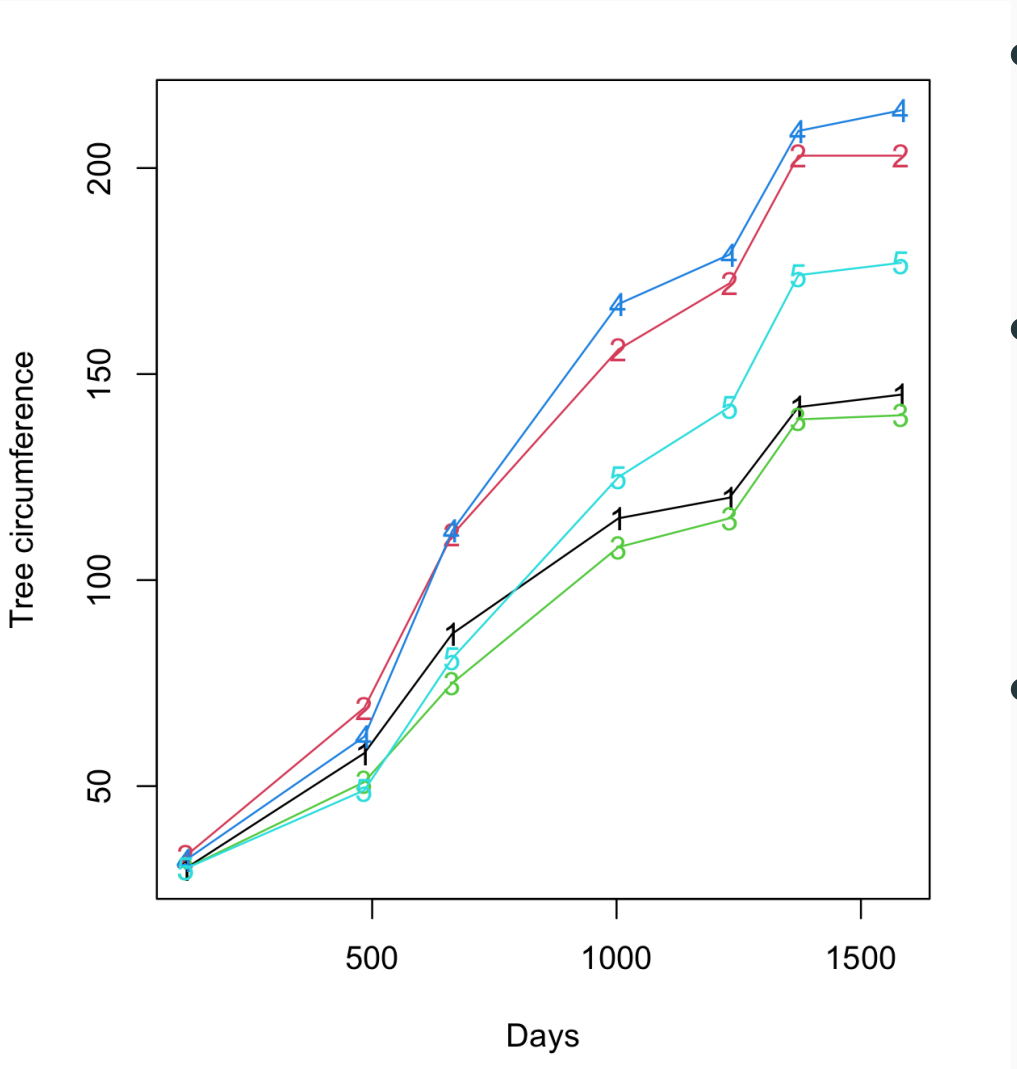
- The `Orange` data frame has 35 rows and 3 columns of records of the growth of orange trees.
  1. `Tree` indicates the tree on which the measurement is made. There are five trees which are ordered according to increasing maximum diameter.
  2. `age` a numeric vector giving the age of the tree (days since 1968/12/31)
  3. `circumference` a numeric vector of trunk circumferences (mm).
- Depending on how large you data set is, you can view it by calling it by name and pressing ENTER:

```
Orange # press ENTER
```

Data frames

- To see the first 6 observations in this data set type:

```
> head(Orange) # head(Orange, n = 6)
# n = 6 is the default, but you can change it
  Tree  age circumference
1    1  118            30
2    1  484            58
3    1  664            87
4    1 1004           115
5    1 1231           120
6    1 1372           142
> head(Orange, 2)
  Tree age circumference
1    1 118            30
2    1 484            58
```



- The figure on the left plots the circumference of the five orange trees over time
- Points are labeled by the tree number and data collected from the same tree are connected by lines
- We can see that tree 3 has the smallest maximum circumference and Tree 4 has the largest maximum circumference

Data frames

- To determine the sample size *n*, that is, to count how many rows or observations it contains use `nrow`. To count the number of columns (i.e. variables) use `ncol`:

```
> nrow(Orange)
[1] 35
> ncol(Orange)
[1] 3
```

- Alternatively, you can retrieve the *dimension* of the data frame which returns the number of rows and columns respectively.

```
> dim(Orange)
[1] 35  3
```

Data frames

- To extract any variable vector from the data set use:

```
> Orange$age
 [1]  118  484  664 1004 1231 1372 1582  118
 [9]  484  664 1004 1231 1372 1582  118  484
[17]  664 1004 1231 1372 1582  118  484  664
[25] 1004 1231 1372 1582  118  484  664 1004
[33] 1231 1372 1582
```

```
> Orange$Tree
 [1] 1 1 1 1 1 1 1 2 2 2 2 2 2 2 3 3 3 3 3 3
[21] 3 4 4 4 4 4 4 4 5 5 5 5 5 5 5
Levels: 3 < 1 < 5 < 2 < 4
```

What are some similarities and differences you see in these two variables?

factors

- Categorical variables in R are stored into a *factor*.
- The allowable categories for the categorical variable vector are defined by the *levels*
- If the categorical variable is ordinal, that is, the levels have some natural ordering or rank, this will be indicated by the comparison operator ($<$)
- For the `Tree` variable, we know that this is an ordinal categorical variable ordered by the maximum circumference of the trees.

factors

Let's consider another built-in data set called `PlantGrowth`. This data frame consists of 30 cases on 2 variables:

`group` The levels of `group` are `ctrl` (control group), `trt1` (treatment 1), and `trt2` (treatment 2).

`weight` measured dried weight of plants

```
> head(PlantGrowth,2) # prints the first 2 rows
  weight group
1   4.17  ctrl
2   5.58  ctrl
> tail(PlantGrowth, 2) # prints the last 2 rows
   weight group
29   5.80  trt2
30   5.26  trt2
```

factors

Below is an example of a nominal (unordered) categorical variable:

```
> PlantGrowth$group
 [1] ctrl ctrl ctrl ctrl ctrl ctrl ctrl ctrl
 [9] ctrl ctrl trt1 trt1 trt1 trt1 trt1 trt1
[17] trt1 trt1 trt1 trt1 trt2 trt2 trt2 trt2
[25] trt2 trt2 trt2 trt2 trt2 trt2
Levels: ctrl trt1 trt2
```

Notice how the levels are separated by spaces, i.e. the less than sign (<) is no longer being used to denote that the categories are ordered.

factor

You can see the data types of these vectors using the function `class()`.

```
> class(Orange$Tree)
[1] "ordered" "factor"
> class(PlantGrowth$group)
[1] "factor"
> class(Orange$age)
[1] "numeric"
> class(PlantGrowth$weight)
[1] "numeric"
```

Note: continuous variables are the default in R and often discrete variables will be stored as such (the discrete data type in R is `integer`).

Loading data in R

- The above examples involved using data sets that have already been installed to R
- In a practical setting, we will most likely be required to load our own data into R.
- The easiest way to do this is by saving your data in csv (comma separated values) format and using the `read.csv()` function; see `?read.csv` for more details
- Assuming your data is stored in your *working directory* (more on this in a second), you can load in R using:

```
x <- read.csv("name_of_file.csv")
```

Loading data into R

```
x <- read.csv("name_of_file.csv")
```

- In this example, `x` is simply the name we give our data frame (in the previous examples they stored the data under the object name `Orange` and `PlantGrowth`)
- Provided we keep to R's rules for naming, we can name this data frame object whatever we want.
- Lets consider the data we used during our clicker question.
- I have stored this to a file called `datamatrix.csv`

Example 1

Load the `datamatrix.csv` into R and save it to the data frame object `dat`.

Loading data into R

There are two ways I would suggest you to do this:

1. Change your working directory first, read in your data second
2. Read in your data while specifying the file path

I would recommend that you use approach number 1. over approach 2. Note that failing to specify the file path or changing to the appropriate working directory will result in an error:

```
> dat <- read.csv("datamatrix.csv")
Error in file(file, "rt") : cannot open the connection
In addition: Warning message:
In file(file, "rt") :
  cannot open file 'datamatrix.csv': No such file or direc
```

Changing your working directory

- If you are using RStudio, you can change your working director using the menu bar directions here.
- Whether or not your are using base R or Rstudio, you can change the working directory using the `setwd()` function.
- For example, if my file is saved in a folder called *stat230* in my *Documents* folder, I would use (on a Mac):

```
> setwd("/Users/ivrbik/Documents/stat230")
```

Once we've change our working directory to the folder in which the csv file is stored, we can run the code without an error:

```
> dat <- read.csv("datamatrix.csv")
```

Loading data into R

Alternatively you can specify the path directly in the file name:

```
> dat <- read.csv("/Users/ivrbik/Documents/stat230/datamatrix.csv")
```

- A common syntax error is to forget the quotations around the file name.
- Reading in data is often a very large source of annoyance with new comers to R
- If you get an error, DON'T PANIC. Labs are here to help you troubleshoot and errors WILL happen (frequently).

- Consider the following output:

```
> class(dat$gender)
[1] "character"
> class(dat$sleep)
[1] "numeric"
> class(dat$intro_extra)
[1] "character"
> class(dat$countries)
[1] "integer"
> class(dat$dread)
[1] "integer"
```

- While R will do it's best to try and guess the appropriate class for each variable, we will occasionally need to specify the data type explicitly ...

Data types in R

- A *character* object in R is used to represent string values
- In our case, we want R to treat `gender` and `intro_extra`

```
> factor(dat$intro_extra)
[1] extravert extravert introvert extravert
[5] introvert extravert
Levels: extravert introvert
```

- Don't forget to reassign this change to the data frame object:

```
factor(dat$intro_extra) <- factor(dat$intro_extra)
dat$intro_extra <- factor(dat$intro_extra)
```

Data types in R

- Extra care needs to be taken when working with the `dread` variable.
- For one, this categorical variable needs to be ordered
- Secondly, some of the possible categories are not present in the date (i.e no one answered `1` or `5`).
- To remedy this, we tell R what the possible values this variable can take on in the `levels` argument of this function and set the `ordered` argument to `TRUE`.

```
dat$dread <- factor(dat$dread,
                    levels=c(1,2,3,4,5),
                    ordered=TRUE)
```

Data types in R

Now we can see that the `dread` variable is stored as it should be:

```
> dat$dread
[1] 3 2 4 2 3 4
Levels: 1 < 2 < 3 < 4 < 5
```

A summary of the R classes corresponding to the data types we have discussed are summarized below: