

# R basics for Stata users

## Field Coordinator Training - R Track

Luiza Andrade, Leonardo Viotti & Rob Marty

June 2018



- 1 Introduction
- 2 Getting started
- 3 RStudio interface
- 4 Object-oriented language
- 5 R objects
- 6 Basic types of data
- 7 Advanced types of data
- 8 Help, Google and Stackoverflow

# Outline

- 1 Introduction
- 2 Getting started
- 3 RStudio interface
- 4 Object-oriented language
- 5 R objects
- 6 Basic types of data
- 7 Advanced types of data
- 8 Help, Google and Stackoverflow
- 9 Useful resources
- 10 Appendix

These training sessions will offer a quick introduction to R, its features and draw a comparison with Stata.

R is a powerful and flexible tool with a big and active community of users and developers that constantly posts in blogs and forums.

## Why choose R over Stata?

- It is way cooler.
- It is less specialized:
  - More flexibility when programming.
  - Many other functionalities.
- Much broader network of users in general:
  - Using google is a lot easier (you'll never want to see Statalist again in your life).
  - Development of new features and bug fixes happens faster.
- It makes prettier graphs.

What are the possible disadvantages of R?

- Steeper learning curve (at least in the beginning).
- Stata is more specialized:
  - Certain common tasks are simpler in Stata, especially when you're doing them for the first time in R.
- Stata has wider adoption among micro-econometricians.
  - Network externalities in your work environment.
  - Development of new specialized techniques and tools could happen faster (e.g. *ietoolkit*).

Here are some other advantages:

- R is a free and open source software!
- It allows you to have several data sets open simultaneously.
- It can run complex Geographic Information System (GIS) analyses.
- You can use it for web scrapping.
- You can run machine learning algorithms with it.
- You can create complex Markdown documents. This presentation, for example, is entirely done in RStudio.
- You can create dashboards and online applications with the Shiny package.

This first session will present the basic concepts you will need to use R.

The next sessions will include:

- **Writing replicable code** - Programming basics and best practices.
- **Data processing** - Data processing workflow and main functions.
- **Descriptive analysis and data visualization** - Exploratory and publication tables and graphs.
- **Basic spatial analysis** - GIS basics in R.

For the most recent versions of these trainings, visit the R-training GitHub repo at <https://github.com/luizaandrade/R-Training>



# Outline

- 1 Introduction
- 2 Getting started
- 3 RStudio interface
- 4 Object-oriented language
- 5 R objects
- 6 Basic types of data
- 7 Advanced types of data
- 8 Help, Google and Stackoverflow
- 9 Useful resources
- 10 Appendix

# Getting started

This training requires that you have R installed in your computer:

## Installation

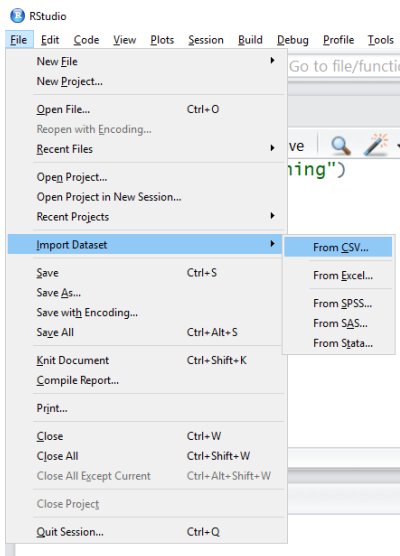
- Please visit (<https://cran.r-project.org>) and select a Comprehensive R Archive Network (CRAN) mirror close to you.
- If you're in the US, you can directly visit the mirror at Berkley university at (<https://cran.cnr.berkeley.edu>).
- Although, not necessary, we also strongly suggest installing R studio. You can get it in (<https://www.rstudio.com/>), but you need to install R first.

## Exercise 1: Import data

Let's start by loading the data set we'll be using:

- 1 In RStudio, go to File > Import Dataset > From Text (Base) and open the `lwh_clean.csv` file. Depending on your Rstudio version, it might be From CSV
- 2 The file should be in NISR R Training/Data/Final
- 3 Change the name to 'lwh' on the import window

# Getting started



# Getting started

Import Text Data

File/Url:  
~/GitHub/R-Training/DataWork/DataSets/Final/lwh\_clean.csv Browse...

Data Preview:

hh_code (integer) ▾	year (integer) ▾	treatment_hh (integer) ▾	site_code (character) ▾	gender_hhh (character) ▾	age_hhh (integer) ▾	num_dependents (integer) ▾	read_and_write (integer) ▾	w_income_total (double) ▾	w...
1001	2014	0	Rwamangana 2	Male	NA	NA	NA	0	

Previewing first 50 entries.

Import Options:

Name: lwh First Row as Names Delimiter: Comma ▾ Escape: None ▾  
Skip: 0 Trim Spaces Quotes: Default ▾ Comment: Default ▾  
Open Data Viewer Locale: Configure... NA: Default ▾

Code Preview:

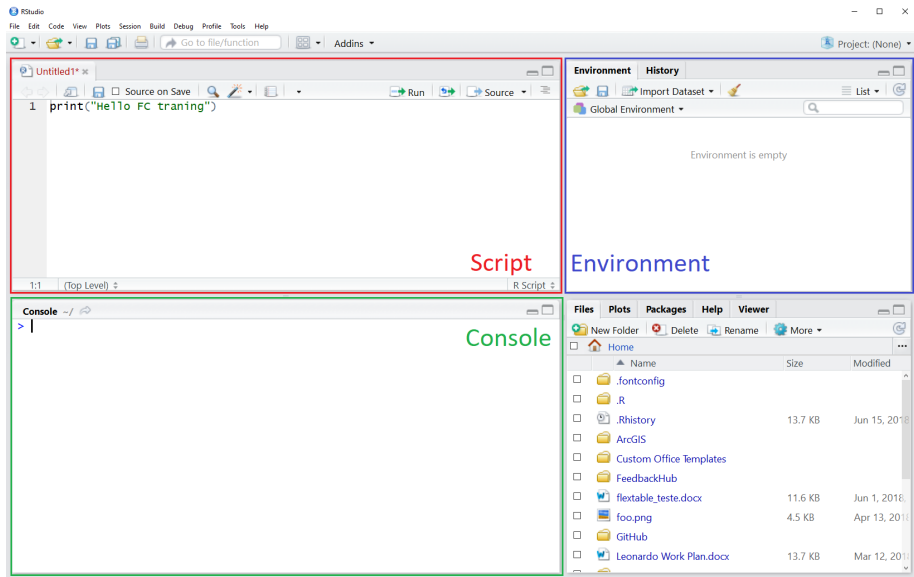
```
library(readr)
lwh_clean <- read_csv("~/GitHub/R-Training/DataWork/DataSets/Final/lwh_clean.csv")
View(lwh_clean)
```

Import Cancel

# Outline

- 1 Introduction
- 2 Getting started
- 3 RStudio interface**
- 4 Object-oriented language
- 5 R objects
- 6 Basic types of data
- 7 Advanced types of data
- 8 Help, Google and Stackoverflow
- 9 Useful resources
- 10 Appendix

# RStudio interface



# RStudio interface

The screenshot displays the RStudio IDE interface. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. Below the menu is a toolbar with icons for file operations and a search bar labeled 'Go to file/function'. The main editor window shows a script titled 'MASTER.R' with the following content:

```
1 #  
2 #  
3 #  
4 # Introduction to R for Stata users  
5 # MASTER SCRIPT  
6 #  
7 #-----  
8  
9 # PURPOSE: Set-up configurations and run scripts that are part of DIME  
10 # Training  
11  
12 # NOTES:  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
1000
```

The console window shows the following R code:

```
> col_integer(),  
treatment_hh = col_integer(),  
site_code = col_character(),  
gender_hhh = col_character(),  
age_hhh = col_integer(),  
num_dependents = col_integer(),  
read_and_write = col_integer(),  
w_income_total = col_double(),  
w_gross_yield_a = col_double(),  
w_gross_yield_b = col_double(),  
w_area_plots_a = col_double(),  
w_area_plots_b = col_double(),  
w_area_plots_c = col_double(),  
expend_food_yearly = col_double(),  
expend_food_lastweek = col_integer(),  
wdds_score = col_integer()  
> View(1wh)  
>
```

The Environment pane on the right shows the 'Data' tab with a table listing the loaded data:

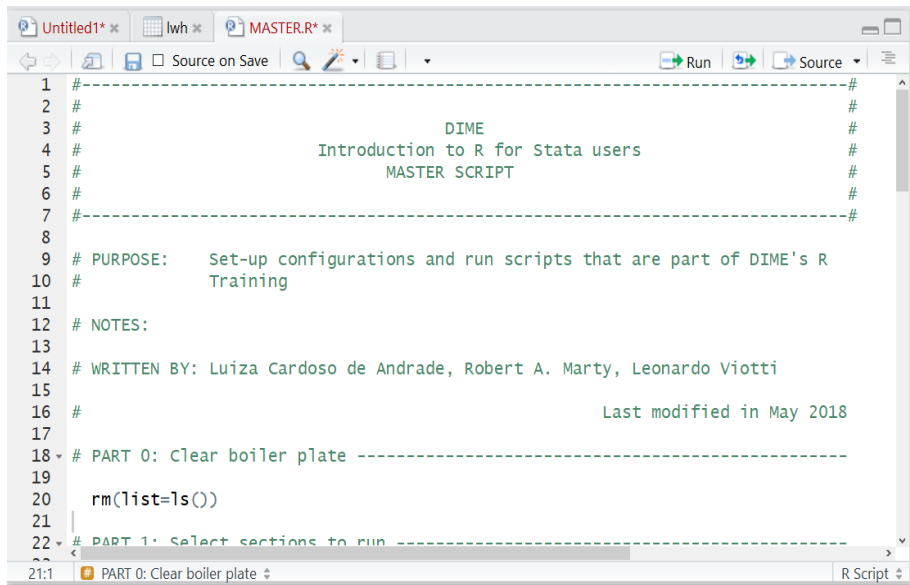
Object	Class	Attributes
1wh	data.frame	1504 obs. of 17 variables

The Files pane at the bottom shows the file explorer with a list of files and folders:

Name	Size	Modified
.fontconfig		
.R		
.Rhistory	13.7 KB	Jun 15, 2018
ArcGIS		
Custom Office Templates		
FeedbackHub		
flextable_teste.docx	11.6 KB	Jun 1, 2018
foo.png	4.5 KB	Apr 13, 2018
GitHub		
Leonardo Work Plan.docx	13.7 KB	Mar 12, 2018



# RStudio interface

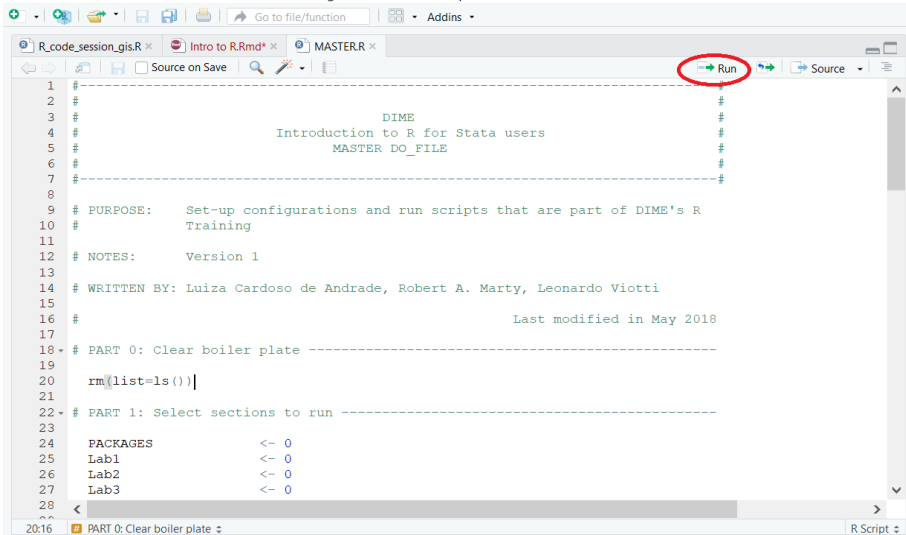


```
1 #-----#
2 #                                           #
3 #                               DIME          #
4 #       Introduction to R for Stata users    #
5 #               MASTER SCRIPT              #
6 #                                           #
7 #-----#
8
9 # PURPOSE:   Set-up configurations and run scripts that are part of DIME's R
10 #           Training
11
12 # NOTES:
13
14 # WRITTEN BY: Luiza Cardoso de Andrade, Robert A. Marty, Leonardo Viotti
15
16 #                                           Last modified in May 2018
17
18 # PART 0: Clear boiler plate -----
19
20 rm(list=ls())
21
22 # PART 1: Select sections to run -----
23
24 # PART 0: Clear boiler plate
```

# RStudio interface

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help



The screenshot shows the RStudio interface with a script editor open. The script contains R code for setting up configurations and running scripts. The code is as follows:

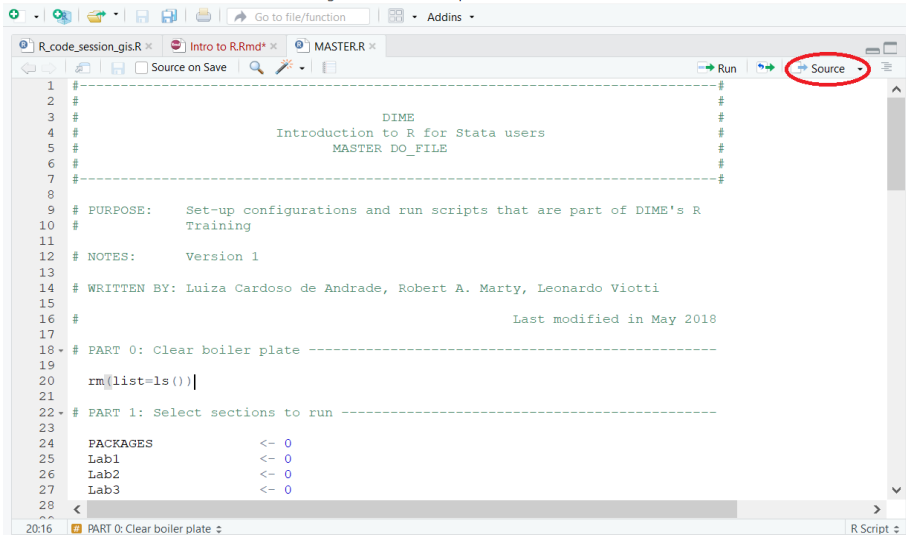
```
1 # -----  
2 #  
3 #  
4 #           DIME  
5 # Introduction to R for Stata users  
6 #           MASTER DO_FILE  
7 # -----  
8  
9 # PURPOSE:   Set-up configurations and run scripts that are part of DIME's R  
10 #           Training  
11  
12 # NOTES:     Version 1  
13  
14 # WRITTEN BY: Luiza Cardoso de Andrade, Robert A. Marty, Leonardo Viotti  
15  
16 #  
17 #           Last modified in May 2018  
18  
19 # PART 0: Clear boiler plate -----  
20 rm(list=ls())  
21  
22 # PART 1: Select sections to run -----  
23  
24 PACKAGES      <- 0  
25 Lab1          <- 0  
26 Lab2          <- 0  
27 Lab3          <- 0  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100
```

The Run button (a green play icon) is circled in red in the top right corner of the script editor. The status bar at the bottom shows the current line is 20:16 and the current section is PART 0: Clear boiler plate.

# RStudio interface

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help



The screenshot shows the RStudio interface with a script editor open. The toolbar at the top includes buttons for Run and Source. The 'Source' button is circled in red. The script editor contains the following R code:

```
1 # -----#
2 #                                           #
3 #                                           #
4 #               DIME                       #
5 #   Introduction to R for Stata users       #
6 #               MASTER DO_FILE            #
7 # -----#
8
9 # PURPOSE:   Set-up configurations and run scripts that are part of DIME's R
10 #           Training
11
12 # NOTES:     Version 1
13
14 # WRITTEN BY: Luiza Cardoso de Andrade, Robert A. Marty, Leonardo Viotti
15
16 #                                           Last modified in May 2018
17
18 # PART 0: Clear boiler plate -----
19
20 rm(list=ls())
21
22 # PART 1: Select sections to run -----
23
24 PACKAGES      <- 0
25 Lab1          <- 0
26 Lab2          <- 0
27 Lab3          <- 0
28
```

The status bar at the bottom shows the current line is 20:16 and the section is PART 0: Clear boiler plate.

# Outline

- 1 Introduction
- 2 Getting started
- 3 RStudio interface
- 4 Object-oriented language**
- 5 R objects
- 6 Basic types of data
- 7 Advanced types of data
- 8 Help, Google and Stackoverflow
- 9 Useful resources
- 10 Appendix

- In Stata, you can open ONE dataset, and perform operations that can change this dataset.
- You can also have other objects, such as matrices, macros and tempfiles, but they are secondary, and most functions only use the main dataset.
- If you wish to do any non-permanent changes to your data, you'll need to preserve the original data to keep it intact.
- R works in a completely different way: you can have as many datasets (objects) as you wish (or your computer's memory allows) and operations will only have lasting effects if you store them.

- Everything that exists in R's memory – variables, datasets, functions – is an object.
- An object is a chunk of data stored in the memory that has a name by which you call it (exactly like macros in Stata).
- If you create an object, it is going to be stored in memory until you delete it or quit R.
- Whenever you run anything you intend to use in the future, you need to store it as an object.

# Object-oriented language

To better understand the idea, we're going to use the data from the Rwanda LWH project. First, let's take a look at the data.

Type the following code to explore the data:

```
# We can use the function View() to browse the whole data  
View(lwh)  
  
# Alternatively we can print the first 5 obs. with head()  
head(lwh)
```

Now, let's try some simple manipulations. First, assume we're only interested in data of the year 2018.

## Exercise 2: Subset the data

- 1 Subset the `lwh` data set, keeping only observations where variable `year` equals 2018.

```
# To do that we'll use the subset() function  
subset(lwh, year == 2018)
```

- 2 Then, look again at the first 5 observations

```
# Use the head() function again  
head(lwh)
```



# Object-oriented language

```
head(lwh)
```

```
##   panel_id hh_code   wave year treatment_hh treatment_site   site_code
## 1      NA     NA   <NA>   NA          <NA>          <NA>          <NA>
## 2  100103   1001   FUP3 2014      Control      Control Rwamangana 2
## 3  100104   1001   FUP4 2016      Control      Control Rwamangana 2
## 4      NA     NA   <NA>   NA          <NA>          <NA>          <NA>
## 5  100202   1002 FUP1&2 2013      Control      Control Rwamangana 2
## 6  100203   1002   FUP3 2014      Control      Control Rwamangana 2
##   gender_hhh age_hhh num_dependents read_and_write w_gross_yield_a
## 1      <NA>     NA              NA              NA              NA
## 2      Male     NA              NA              NA              0
## 3      Male    28              NA              NA             13050
## 4      <NA>     NA              NA              NA              NA
## 5     Female    51              3              0             28000
## 6     Female    NA              NA              NA              0
##   w_gross_yield_b expend_food_yearly expend_food_lastweek
## 1              NA              NA              NA
## 2      114583.34      156532.5000      3000
## 3           0.00      250452.0000      4800
## 4              NA              NA              NA
## 5      156218.31       260.8875       5
## 6      61969.74      163576.4700      3135
```

# Object-oriented language

We can see that nothing happened to the original data. This happens because we didn't store the edit we made anywhere.

To store an object, we use the assignment operator (`<-`):

```
# Assign a value of the Answer to the Ultimate Question of  
# Life, the Universe, and Everything  
x <- 42
```

From now on, `x` is associated with the stored value (until you replace it delete it or close R).

## Exercise 3: Create an object

Create a new data set, called `lwh2018`, that is a subset of the `lwh` data set containing only data from the year 2018.

*# Using the same function but now assigning it to an object*

```
lwh2018 <- subset(lwh, year == 2018)
```

*# Display the 5 first obs. of the new data*

```
head(lwh2018)
```

*# Notice that we still have the original data set intact*

```
head(lwh)
```

# Object-oriented language

```
head(lwh2018)
```

```
##      panel_id hh_code      wave year treatment_hh treatment_site  site_code
## 11    100305    1003 Endline 2018      Control      Control Rwamangana 2
## 14    100405    1004 Endline 2018      Control      Control Rwamangana 2
## 19    100605    1006 Endline 2018      Control      Control Rwamangana 2
## 24    100705    1007 Endline 2018      Control      Control Rwamangana 2
## 31    101005    1010 Endline 2018      Control      Control Rwamangana 2
## 33    101105    1011 Endline 2018      Control      Control Rwamangana 2
##      gender_hhh age_hhh num_dependents read_and_write w_gross_yield_a
## 11      Female     53              NA              NA      200238.09
## 14      Female     93              NA              NA      471014.47
## 19       Male     27              NA              NA       51785.71
## 24      Female     58              NA              NA      325555.53
## 31      Female     35              NA              NA      129152.91
## 33      Female     58              NA              NA      129999.99
```

## Two an important concepts to take note:

- ❶ In R, if you want to change your data, you need to store it in an object. It is possible to simply replace the original data, but often, it's more practical to create a new dataset.
- ❷ Print (display) is built into R. If you execute an action and don't store it anywhere, R will simply print the results of that action but won't save anything in the memory.

# Outline

- 1 Introduction
- 2 Getting started
- 3 RStudio interface
- 4 Object-oriented language
- 5 R objects**
- 6 Basic types of data
- 7 Advanced types of data
- 8 Help, Google and Stackoverflow
- 9 Useful resources
- 10 Appendix

Objects can have different structures, that is, different ways to store data. Objects are the building blocks of R programming. You can create and manipulate them to explore your data and construct analytical outputs.

There are several types of objects. Here are the ones we will cover:

- **Vectors:** an unidimensional object that stores a sequence of values
- **Matrices:** a combination of vectors of the same length and same type.
- **Data frames:** a combination of different vectors of different types of the same length (the same as your data set in Stata)
- **Lists:** a multidimensional object that can store several objects of different dimension

# R objects

## Vectors

A vector is an unidimensional object composed by one or more scalars of the same type.

Use the following code to create vectors in two different ways

```
# Creating a vector with the c() function
```

```
v1 <- c(1,1,2,3,5)
```

```
# Alternative way to create an evenly spaced vector
```

```
v2 <- 1:5
```

You can use brackets for indexing

```
# Print the 4th element of the vector
```

```
v2[4]
```

```
## [1] 4
```



# R objects

## Vectors

To R, each of the columns of `lwh` is a vector.

Calling a vector from a `data.frame` column

We use the `$` to call vector (variables) by their names in a `data.frame`

Type the following code:

```
# Create a vector with the values of `age_hhh` variable  
age_vec <- lwh$age_hhh  
  
# See the 13th element of the column  
lwh$age_hhh[67]  
  
## [1] 61
```

# R objects

## Matrices

A matrix is a bidimensional object composed by one or more vectors of the same type.

Type the following code to test two different ways of creating matrices

```
# Matrix created by joining two vectors:
```

```
m1 <- cbind(v1,v1)
```

```
# Matrix using the
```

```
m2 <- matrix(c(1,1,2,3,5,8), ncol = 2)
```

# R objects

## Matrices

Since a matrix has two dimensions, you can use indexing on both:

Matrix indexing: typing `matrix[i,j]` will give you the element in the *i*th line and *j*th column

Now use the following code to check the elements of the matrices we created by indexing

```
# the element in the i-th row and j-th column of that matrix  
m2[1,2]
```

```
## [1] 3
```

# R objects

## Matrices

You can also use it to select only the lines or columns

```
# ith row of that matrix
```

```
m1[1,]
```

```
## v1 v1
```

```
## 1 1
```

```
# jth column of that matrix (as a vector)
```

```
m1[,2]
```

```
## [1] 1 1 2 3 5
```

# R objects

## Data Frames

The `lwh` and `lwh2018` objects are both data frames. You can also construct a new data.frame from scratch by combining vectors.

Now, type the following code to create a new data frame

```
# Dataframe created by binding vectors
```

```
df1 <- data.frame(v1,v2)
```

```
df1
```

```
##      v1 v2
```

```
## 1    1  1
```

```
## 2    1  2
```

```
## 3    2  3
```

```
## 4    3  4
```

```
## 5    5  5
```

You can also use indexing with data frames.

### Numeric indexing

*# The first column of lwh*

```
lwh[,1]
```

*# The 45th line of lwh*

```
lwh[45,]
```

*# Or the 45th element of the first line*

```
lwh[45,1]
```

Alternatively, you can use the column names for indexing, which is the same as using the \$ sign.

### Names indexing

*# Or the 42th element of the hh\_code column*

```
lwh[42, "hh_code"]
```

```
## [1] 1014
```

# R objects

## Data Frames

Data frames also contain metadata stored in attributes. All objects can have attributes, but data frame objects already come with a few.

### Attributes

```
# Check all attributes of df1
```

```
attributes(df1)
```

```
## $names
```

```
## [1] "v1" "v2"
```

```
##
```

```
## $row.names
```

```
## [1] 1 2 3 4 5
```

```
##
```

```
## $class
```

```
## [1] "data.frame"
```

```
# there are also specialized functions to check and manipulate specific attributes
```

```
names(df1)
```

```
## [1] "v1" "v2"
```



# R objects

## Lists

Lists are more complex objects that can contain many objects of different classes and dimensions.

Lists are fancy and can have a lot of functionalities. They are the output of many functions and are used to construct complex objects.

It would be beyond the scope of this introduction to go deep into them, but here's a quick example

Combine several objects of different types in a list

```
# Use the list() function  
lst <- list(v1, df1, 45)
```

Print the list yourself to see how it looks like.

# R objects

## Lists

```
# Check the contents of lst
```

```
print(lst)
```

```
## [[1]]
```

```
## [1] 1 1 2 3 5
```

```
##
```

```
## [[2]]
```

```
##      v1 v2
```

```
## 1    1  1
```

```
## 2    1  2
```

```
## 3    2  3
```

```
## 4    3  4
```

```
## 5    5  5
```

```
##
```

```
## [[3]]
```

```
## [1] 45
```

# Outline

- 1 Introduction
- 2 Getting started
- 3 RStudio interface
- 4 Object-oriented language
- 5 R objects
- 6 Basic types of data**
- 7 Advanced types of data
- 8 Help, Google and Stackoverflow
- 9 Useful resources
- 10 Appendix

# Basic types of data

R has different kinds of data that can be recorded inside objects. They are very similar to what you have in Stata, and the main types are string, integer, numeric, factor and boolean.

Let's start with the simpler ones:

## Strings

A sequence of characters and are usually represented between double quotes. They can contain single letters, words, phrases or even some longer text.

## Integer and numeric

As in Stata, these are two different ways to store numbers. They are different because they use memory differently. As default, R stores numbers in the numeric format (double).

# Basic types of data

## Strings

Now we'll use string data to practice some basic object manipulations in R.

### Exercise 4: Create a vector of strings

Create a string vector containing the names of commonly used statistical software:

```
# Creating string vector  
str_vec <- c("R",  
             "Python",  
             "SAS",  
             "Microsoft Excel",  
             "Stata")
```

Now print them to check them out.

### Exercise 5: Concatenate strings

- 1 Create a scalar (a vector of one element) containing the phrase “is better than” and call it `str_scalar`.
- 2 Use the function `paste()` with 3 arguments separated by commas:
  - The first argument as the 1st element of `str_vec`.
  - The second argument as the `str_scalar`.
  - The third argument as the 5th element of `str_vec`.
- 3 If you're not sure where to start, type:

```
help(paste)
```

# Basic types of data

## Strings

```
### Using the paste function to combine strings
```

```
# Scalar
```

```
str_scalar <- "is better than"
```

```
# Using the paste() function
```

```
paste(str_vec[1], str_scalar, str_vec[5])
```

```
## [1] "R is better than Stata"
```

# Outline

- 1 Introduction
- 2 Getting started
- 3 RStudio interface
- 4 Object-oriented language
- 5 R objects
- 6 Basic types of data
- 7 Advanced types of data**
- 8 Help, Google and Stackoverflow
- 9 Useful resources
- 10 Appendix



# Advanced types of data

R also has other more complex ways of storing data. These are the most used:

## Factors

Factors are numeric categorical values with text label, equivalent to labelled variables in Stata. Turning strings into factors makes it easier to run different analyses on them and also uses less space in the memory, which is why data frames do that by default.

## Booleans

Booleans are logical binary variables, accepting either TRUE or FALSE as values. They are automatically generated when performing logical operations

# Advanced types of data

## Factors

You can see that in `lwh` the `wave`, `treatment_hh`, `treatment_site`, `site_code` and `gender_hhh` are factor variables. You can see in your environment panel the type of all your variables, and for factors the number of levels.

# Advanced types of data

## Factors

Create a factor vector using the following code

```
# Basic factor vector
```

```
num_vec <- c(1,2,2,3,1,2,3,3,1,2,3,3,1)
```

```
fac_vec <- factor(num_vec)
```

```
# A bit fancier factor vector
```

```
fac_vec <- factor(num_vec, labels=c("A", "B", "C"))
```

```
# Change labels
```

```
levels(fac_vec) = c('One', 'Two', 'Three')
```

# Advanced types of data

## Factors

We'll learn how to deal with factors in detail on the next session, since they are very important for us. For now, here are two important things to keep in mind when using them:

### Warning:

Unlike Stata, in R

- 1 You use the labels to refer to factors
- 2 You cannot choose the underlying values

# Advanced types of data

## Booleans

Boolean data is not normally used directly in data frames, but rather to express the results of a logical condition.

### Exercise 6:

Create boolean vector with the condition of annual income below average:

```
# Create vector
bool_vec <- (lwh$income_total_win <
             mean(lwh$income_total_win))

# See the 5 first elements of the vector
head(bool_vec)

## [1] TRUE TRUE TRUE TRUE TRUE TRUE
```

# Advanced types of data

## Booleans

Let's use the boolean vector created to add a dummy variable in the `lwh` data set for the same condition.

### Exercise 6:

- 1 Create a column in `lwh` containing zeros and call it `income_low`. You can do this by typing:

```
lwh$income_low <- 0
```

- 2 Use `bool_vec` to index the lines of the `income_low` column and replace all observations that meet the condition with the value 1.

```
lwh$income_low[bool_vec] <- 1
```

# Advanced types of data

## Booleans

```
# Create column with zeros
```

```
lwh$income_low <- 0
```

```
# Replace with 1 those obs that meet the condition
```

```
lwh$income_low[bool_vec] <- 1
```

```
# See the first 5. obs.
```

```
head(lwh$income_low)
```

```
## [1] 1 1 1 1 1 1
```

# Outline

- 1 Introduction
- 2 Getting started
- 3 RStudio interface
- 4 Object-oriented language
- 5 R objects
- 6 Basic types of data
- 7 Advanced types of data
- 8 Help, Google and Stackoverflow**
- 9 Useful resources
- 10 Appendix



Help in R works very much like in Stata: the help files usually start with a brief description of the function, explain its syntax and arguments and list a few examples. There are two ways to access help files:

## Exercise 7: Use help

```
# The help() function
```

```
help(summary)
```

```
# and its abbreviation
```

```
?summary
```

- The biggest difference, however, is that R has a much wider user community and it has a lot more online resources.
- For instance, in 2014, Stata had 11 dedicated blogs written by users, while R had 550.<sup>1</sup>
- The most powerful problem-solving tool in R, however, is Google. Searching the issue, you have or the error message displayed usually yields tons of results and will probably lead you to a Stack Overflow page where someone asked the same question and several people gave different answers.

---

<sup>1</sup>Check <http://r4stats.com/articles/popularity/> for more.

# Outline

- 1 Introduction
- 2 Getting started
- 3 RStudio interface
- 4 Object-oriented language
- 5 R objects
- 6 Basic types of data
- 7 Advanced types of data
- 8 Help, Google and Stackoverflow
- 9 Useful resources**
- 10 Appendix

# Useful resources

## Blogs and online courses:

- Surviving graduate econometrics with R:  
<https://thetarzan.wordpress.com/2011/05/24/surviving-graduate-econometrics-with-r-the-basics-1-of-8/>
- An Introduction to R at <https://cran.r-project.org/>
- R programming in Coursera:  
<https://www.coursera.org/learn/r-programming>
- Try R in Code School: <http://tryr.codeschool.com/>
- R programming for dummies:  
<http://www.dummies.com/programming/r/>
- R bloggers: <https://www.r-bloggers.com/>
- R statistics blog: <https://www.r-statistics.com/>
- The R graph gallery: <https://www.r-graph-gallery.com/>

## Books:

- R for Stata Users - Robert A. Muenchen and Joseph Hilbe
- R Graphics Cookbook - Winston Chang
- R for Data Science - Hadley Wickha and Garrett Grolemond

Thank you!

# Outline

- 1 Introduction
- 2 Getting started
- 3 RStudio interface
- 4 Object-oriented language
- 5 R objects
- 6 Basic types of data
- 7 Advanced types of data
- 8 Help, Google and Stackoverflow
- 9 Useful resources
- 10 Appendix**

R's syntax is a bit heavier than Stata's:

- Parentheses to separate function names from its arguments.
- Commas to separate arguments.
- For comments we use the # sign.
- You can have line breaks inside function statements.
- In R, functions can be treated much like any other object Therefore, they can be passed as arguments to other functions.

Similarly to Stata:

- Square brackets are used for indexing.
- Curly braces are used for loops and if statements.
- Largely ignores white spaces.



# RStudio interface

## Script

Where you write your code. Just like a do file.

## Console

Where your results and messages will be displayed. But you can also type commands directly into the console, as in Stata.

## Environment

What's in R's memory.

## The 4th pane

Can display different things, including plots you create, packages loaded and help files.

# Numbers and integers

Two scalars, one with a round number the other with a fractional part

```
# a numeric scalar with an integer number
```

```
int <- 13
```

```
num <- 12.99
```

# Numbers and integers

Now we can see the objects classes with the `class()` function and test it with the `is.integer()` and `is.numeric()` functions.

*# you can see the number's format using the class function:*

```
class(int)
```

```
## [1] "numeric"
```

```
class(num)
```

```
## [1] "numeric"
```

*# you can test the class with the is. method*

```
is.integer(int)
```

```
## [1] FALSE
```

# Numbers and integers

## Numbers and integers

We can, however, coerce objects into different classes. We just need to be careful because the result might not be what we're expecting.

Use the *as.integer()* and *round()* functions on the *num* object to see the difference:

```
as.integer(num)
```

```
## [1] 12
```

```
# and
```

```
round(num)
```

```
## [1] 13
```