



Questions

Tags

Users

Badges

Unanswered

Ask Question

Read multiple CSV files into separate data frames

DID YOU KNOW?

GO DADDY DNS ANSWERS 10 BILLION queries daily & is available over IPV6

[Learn More](#)

Serious about tech? So are we. [tech.godaddy.com](#)

Suppose we have files `file1.csv`, `file2.csv`, ... , and `file100.csv` in directory `C:\R\Data` and we want to read them all into separate data frames (e.g. `file1`, `file2`, ... , and `file100`).

The reason for this is that, despite having similar names they have different file structures, so it is not that useful to have them in a list.

I could use `lapply` but that returns a single list containing 100 data frames. Instead I want these data frames in the Global Environment.

How do I read multiple files directly into the global environment? Or, alternatively, How do I unpack the contents of a list of data frames into it?

[r](#) [csv](#) [file](#) [lapply](#)

[link](#) | [improve this question](#)

edited **Mar 16 '11 at 13:35**

[Joshua Ulrich](#)

31.2k ●3 ●34 ●67

asked **Mar 16 '11 at 0:55**

[Fred](#)

423 ●1 ●10

100% accept rate

1 @Roman Luštrik Please see comment to @hadley below. Note I did not ask "What is the best way to read X number of files into R?". My question is more specific for a reason. I guess I should not have said I wanted to read 100 files (simply trying to be general) but 8 different files with similar names. But there are too many people here off on their high horse. – [Fred](#) Mar 16 '11 at 13:22 #

[feedback](#)

5 Answers

[active](#) [oldest](#) [votes](#)

- Quick draft, untested:
1. Use `list.files()` aka `dir()` to dynamically generate your list of files.
 2. This returns a vector, just run along the vector in a `for` loop.
 3. Read the i-th file, then use `assign()` to place the content into a new variable `file_i`

That should do the trick for you.

[link](#) | [improve this answer](#)

answered **Mar 16 '11 at 1:00**

[Dirk Eddelbuettel](#)

68k ●6 ●74 ●145

@Dirk Eddelbuettel Thanks, that works. Indeed that is what I tried to do originally but using `i <- read.csv(...)` inside the loop instead of `assign(i,read.csv(...))` . Why doesn't the former work? – [Fred](#) Mar 16 '11 at 1:19

1 Local scope versus global environment. You could try `i <- read.csv(...)` as well. – [Dirk Eddelbuettel](#) Mar 16 '11 at 1:21

@Dirk Eddelbuettel Many thanks, final question: Had I used `lapply` and dumped everything inside a list,

Hello World!

This is a collaboratively edited question and answer site for **professional and enthusiast programmers**. It's 100% free, no registration required.

[about »](#) [faq »](#)

tagged

[file](#) x 14249

[r](#) x 14063

[csv](#) x 4931

[lapply](#) x 63

asked 1 year ago

viewed 2,352 times

active 1 year ago

DID YOU KNOW?

GO DADDY MANAGES 25 PETABYTES of networked data storage

Serious about tech? So are we. [tech.godaddy.com](#)

[Learn More](#)

CAREERS 2.0

[Senior Developer](#)
[SpyderLynk](#)
Denver, CO

[C++ Developer](#)
[Research In Motion](#)
Andover, MA

[Wayfair.com -- Fulfillment Engineer](#)
(Hebron, Kentucky)
[Wayfair](#)
Hebron, KY

Linked

[reading csv files in a for loop and assigning dataframe names](#)

Related

how would I "unpack it"? I ask because `lapply` is much faster and I dislike loops. – [Fred](#) Mar 16 '11 at 1:25

3 Prove that `lapply` is faster in reading N files. Moreover, if you dislike loops the burden is on you to read up on the `*apply` family. And again, these days they are *not* generally faster. – [Dirk Eddelbuettel](#) Mar 16 '11 at 1:31

6 Yowser, assign and <- in the same answer! Has someone hijacked Dirk's account? – [mdsumner](#) Mar 16 '11 at 3:09

show 2 more comments

feedback



Don't. Keep them as a list. It's the way to go.

6

link | improve this answer

answered Mar 16 '11 at 3:13



[hadley](#)

17.5k ● 1 ● 27 ● 65

2 - because anything you are likely to want to do with 100 data frames will be easier to do if they are in a list than if they are 100 data frames with names file1 to file100. – [Spacedman](#) Mar 16 '11 at 8:16

2 @hadley @Spacedman I am actually not reading 100 files but 8. And although they have similar names they are very different in structure, so ill suited for working with `*apply` family of functions. There is a reason I asked the question I asked. – [Fred](#) Mar 16 '11 at 13:13

1 We can only answer the question that you posed! If your stackoverflow question isn't the same as your real question, you can't expect to get the best answers. – [hadley](#) Mar 16 '11 at 13:39

1 @hadley If you want to make a point the way I would have done it is (1) Answer the question like Dirk did: "This is how you do x" and (2) Mention that it may not be a good idea to do so. Instead some people just impose the party line unawares that the reason some we come to this forum is precisely to ask the not so obvious. – [Fred](#) Mar 16 '11 at 14:29

2 @Fred - if you asked me how to commit suicide, I would walk you over to the counselling center and make sure you got help. It's unethical to do anything else. I will continue to give answers that I think people need, not the answers people want. If you don't like it downvote me and move on with your life. – [hadley](#) Mar 16 '11 at 16:43

show 2 more comments

feedback



Use `assign` with a character variable containing the desired name of your data frame.

2

```
for(i in 1:100)
{
  oname = paste("file", i, sep="")
  assign(oname, read.csv(paste(oname, ".txt", sep="")))
}
```

link | improve this answer

answered Mar 16 '11 at 1:02



[Hong Ooi](#)

3,825 ● 5 ● 16

feedback

[How to create a column containing a string of stars to indicate levels of a factor in a data frame in R](#)

[How to Import a CSV file containing multiple sections into R?](#)

[setting levels inside lapply loop in r](#)

[How to return a data.frame with a given name from a function?](#)

[R: reference iteration number in call to sfLapply\(1:N, function\(x\)\)](#)

[Use mapply to fit list of lm models to list of data frames \[R\]](#)

[Using lapply with changing arguments](#)

[Using lapply\(\) for data structured in lists of lists from simulation study](#)

[Extract columns from list of coeftest objects](#)

[combination of expand.grid and mapply?](#)

[multiple plots from data frames in a list after a conditional test](#)

[R apply function with multiple parameters](#)

[lapply is part of what package in R?](#)

[Using get inside lapply, inside a function](#)

[Apply function in R](#)

[Using multicore in R to analyse GWAS data](#)

[Measure the max value of all previous values in a data frame](#)

[Can't access items after an lapply](#)

[How to format data in \(a\) CSV file\(s\) so that it can easily be imported in R?](#)

[Writing multiple values in different columns in a csv file in R](#)

[Access lapply index names inside FUN](#)

[Applying strsplit function for multiple files](#)

[Repeat elements for unequal size objects in an R list](#)

[Use filename to average data by month](#)

[Convert data frame of redundant frequencies](#)



1



A simple way to access the elements of a list from the global environment is to `attach` the list. Note that this actually creates a new environment on the search path and copies the elements of your list into it, so you may want to remove the original list after attaching to prevent having two potentially different copies floating around.

[link](#) | [improve this answer](#)

answered **Mar 16 '11 at 14:31**



Aaron

7,269 ●9 ●20

[feedback](#)



1



Thank you all for replying.

For completeness here is my final answer for loading any number of (tab) delimited files, in this case with 6 columns of data each where column 1 is characters, 2 is factor, and remainder numeric:

```
##Read files named xyz1111.csv, xyz2222.csv, etc.
filenames <- list.files(path="../Data/original_data",
  pattern="xyz+.*csv")

##Create list of data frame names without the ".csv" part
names <- substr(filenames,1,7))

###Load all files
for(i in names){
  filepath <- file.path("../Data/original_data/",paste(i, ".csv", sep=""))
  assign(i, read.delim(filepath,
    colClasses=c("character", "factor", rep("numeric", 4)),
    sep = "\t"))
}
```

[link](#) | [improve this answer](#)

edited **Mar 16 '11 at 20:13**

[community wiki](#)

[4 revs](#)

[Fred](#)

- 1 A couple of things: 1) you don't need to use `lapply` to generate the data frame names, because `substr` is already vectorised; just use `substr(filenames, 1, 7)`. And 2) if your data is not actually comma delimited, you shouldn't use `read.csv`. The point of that function is to read csv files, not general delimited data. If your data is tab delimited, consider `read.delim` (and you don't need the `header=T` part either). – [Hong Ooi](#) Mar 16 '11 at 2:01

@Hong Ooi Many thanks! Corrected. The original files are tab delimited `.txt` with some weird encoding. If I import those I get garbage columns named `X` at end of dataframe. So I opened `.txt` Open Office Calc, saved as `.csv` and now they import fine. Somehow Calc did not replace tab separation when saving as csv file. – [Fred](#) Mar 16 '11 at 2:17

- 1 A couple of very minor points: (1) using [single forward] slashes as path separators is platform-independent (and seems neater to me, but that's a matter of taste); (2) `file.path()` could be substituted for your outer `paste()` [again not a big deal but slightly more semantic] – [Ben Bolker](#) Mar 16 '11 at 2:22

@Ben Bolker Thanks! Corrected. I am new to Stack Overflow. Learning a lot! – [Fred](#) Mar 16 '11 at 2:40

[feedback](#)

Your Answer



log in

or

Name

Email

Home Page

By posting your answer, you agree to the [privacy policy](#) and [terms of service](#).

Not the answer you're looking for? Browse other questions tagged [r](#) [csv](#) [file](#) [apply](#) or [ask your own question](#).

 question feed