

save() and load()

Fundamental reproducibility functions

Charlotte Frei, PhD, PE

January 29, 2020



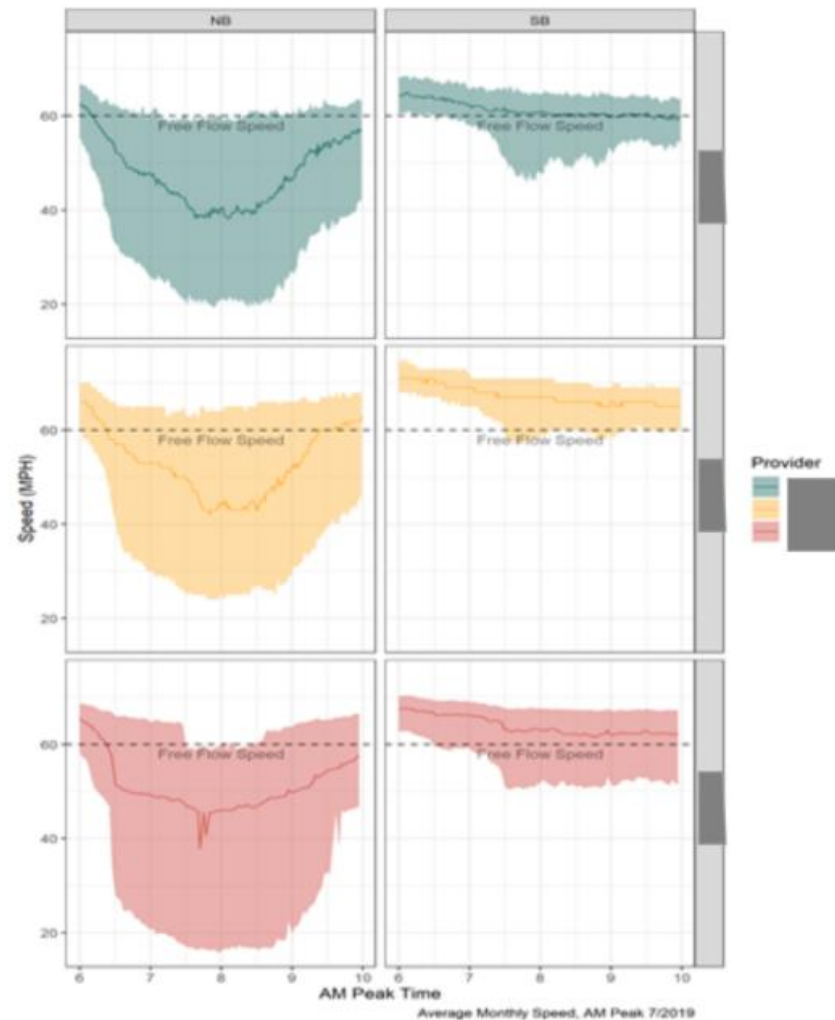
**CDM
Smith**

Outline

- Why save() and load()?
- How-ish
- That's neat?

Why?

- Modularity
- Reproducibility
- Future revision, improvements
- Portability



save()

Arguments

- ... the names of the objects to be saved (as symbols or character strings).
- list** A character vector containing the names of objects to be saved.
- file** a (writable binary-mode) [connection](#) or the name of the file where the data will be saved (when [tilde expansion](#) is done). Must be a file name for `save.image` or `version = 1`.
- ascii** if `TRUE`, an ASCII representation of the data is written. The default value of `ascii` is `FALSE` which leads to a binary file being written. If `NA` and `version >= 2`, a different ASCII representation is used which writes double/complex numbers as binary fractions.
- version** the workspace format version to use. `NULL` specifies the current default format (3). Version 1 was the default from R 0.99.0 to R 1.3.1 and version 2 from R 1.4.0 to 3.5.0. Version 3 is supported from R 3.5.0.
- envir** environment to search for objects to be saved.
- compress** logical or character string specifying whether saving to a named file is to use compression. `TRUE` corresponds to `gzip` compression, and character strings `"gzip"`, `"bzip2"` or `"xz"` specify the type of compression. Ignored when `file` is a connection and for workspace format version 1.
- compression_level** integer: the level of compression to be used. Defaults to `6` for `gzip` compression and to `9` for `bzip2` or `xz` compression.
- eval.promises** logical: should objects which are promises be forced before saving?
- precheck** logical: should the existence of the objects be checked before starting to save (and in particular before opening the file/connection)? Does not apply to version 1 saves.
- safe** logical. If `TRUE`, a temporary file is used for creating the saved workspace. The temporary file is renamed to `file` if the save succeeds. This preserves an existing workspace `file` if the save fails, but at the cost of using extra disk space during the save.

save() simple

```
save(objects, file = "myFile.Rdata")
```

- Use for: objects that you want to reference/reshape in other files, or send to colleagues for analysis

save() example

- Saving color palette vector, plot objects, data.frames, reference info:

↓ descriptive file name with date so you can delete files when you realize you screwed up

```
fileName = paste(substr(Sys.time(), 1, 10), "_studyZone_DataTrans8_", lookup$month[m], lookup$ye  
fileLoc = c("C:/users/freica/Documents/3rdPartyData/data/monthlyData/")  
save(misseg, weekpsummary, monthpsummary, mst2, file = paste(fileLoc, fileName, sep = ""))
```

↗ List of objects you want to take with you

↑ the location and name

↖ Local or shared location for drafts/final versions for team

- To read into another .Rmd or modular code file

load()

Reload Saved Datasets

Reload datasets written with the function `save` .

Keywords [file](#)

Usage

```
load(file, envir = parent.frame(), verbose = FALSE)
```

Arguments

- file** a (readable binary-mode) [connection](#) or a character string giving the name of the file to load (when [tilde expansion](#) is done).
- envir** the environment where the data should be loaded.
- verbose** should item names be printed during loading?

Look up tables for modularity, portability

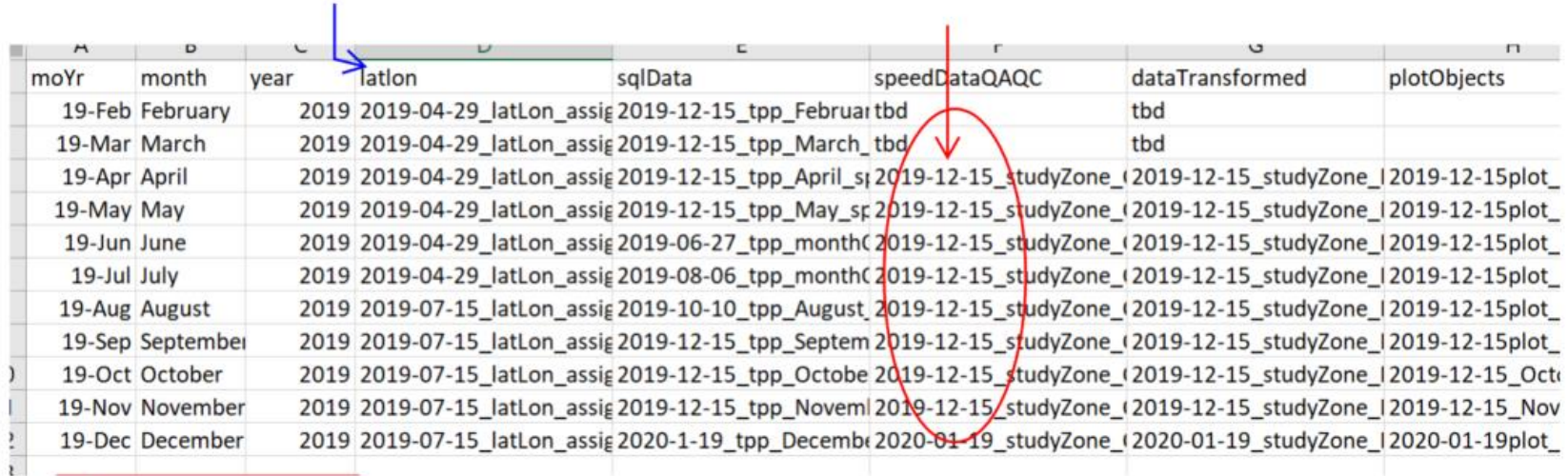
A	B	C	D	E	F	G	H
moYr	month	year	latlon	sqlData	speedDataQAQC	dataTransformed	plotObjects
19-Feb	February	2019	2019-04-29_latLon_assign	2019-12-15_tpp_Februar	tbd	tbd	
19-Mar	March	2019	2019-04-29_latLon_assign	2019-12-15_tpp_March	tbd	tbd	
19-Apr	April	2019	2019-04-29_latLon_assign	2019-12-15_tpp_April_s	2019-12-15_studyZone_	2019-12-15_studyZone_	2019-12-15plot
19-May	May	2019	2019-04-29_latLon_assign	2019-12-15_tpp_May_s	2019-12-15_studyZone_	2019-12-15_studyZone_	2019-12-15plot
19-Jun	June	2019	2019-04-29_latLon_assign	2019-06-27_tpp_monthC	2019-12-15_studyZone_	2019-12-15_studyZone_	2019-12-15plot
19-Jul	July	2019	2019-04-29_latLon_assign	2019-08-06_tpp_monthC	2019-12-15_studyZone_	2019-12-15_studyZone_	2019-12-15plot
19-Aug	August	2019	2019-07-15_latLon_assign	2019-10-10_tpp_August	2019-12-15_studyZone_	2019-12-15_studyZone_	2019-12-15plot
19-Sep	September	2019	2019-07-15_latLon_assign	2019-12-15_tpp_Septem	2019-12-15_studyZone_	2019-12-15_studyZone_	2019-12-15plot
19-Oct	October	2019	2019-07-15_latLon_assign	2019-12-15_tpp_Octobe	2019-12-15_studyZone_	2019-12-15_studyZone_	2019-12-15_Oct
19-Nov	November	2019	2019-07-15_latLon_assign	2019-12-15_tpp_Novem	2019-12-15_studyZone_	2019-12-15_studyZone_	2019-12-15_Nov
19-Dec	December	2019	2019-07-15_latLon_assign	2020-1-19_tpp_Decembe	2020-01-19_studyZone_	2020-01-19_studyZone_	2020-01-19plot

- `load(paste(filename, sqlData[row]))`
- This way you can parse steps/scripts for delegating to others- so long as **objects you are loading/referencing do not change names**

kind of neat? One script per step, load from look up table

spatial info for when
your Vendors change
things

update your look-up table
when you
correct/enhance things!



A	B	C	D	E	F	G	H
moYr	month	year	latlon	sqlData	speedDataQAQC	dataTransformed	plotObjects
19-Feb	February	2019	2019-04-29_latLon_assig	2019-12-15_tpp_Februar	tbd	tbd	
19-Mar	March	2019	2019-04-29_latLon_assig	2019-12-15_tpp_March_tbd		tbd	
19-Apr	April	2019	2019-04-29_latLon_assig	2019-12-15_tpp_April_s	2019-12-15_studyZone_	2019-12-15_studyZone_	2019-12-15plot_
19-May	May	2019	2019-04-29_latLon_assig	2019-12-15_tpp_May_sf	2019-12-15_studyZone_	2019-12-15_studyZone_	2019-12-15plot_
19-Jun	June	2019	2019-04-29_latLon_assig	2019-06-27_tpp_monthC	2019-12-15_studyZone_	2019-12-15_studyZone_	2019-12-15plot_
19-Jul	July	2019	2019-04-29_latLon_assig	2019-08-06_tpp_monthC	2019-12-15_studyZone_	2019-12-15_studyZone_	2019-12-15plot_
19-Aug	August	2019	2019-07-15_latLon_assig	2019-10-10_tpp_August	2019-12-15_studyZone_	2019-12-15_studyZone_	2019-12-15plot_
19-Sep	September	2019	2019-07-15_latLon_assig	2019-12-15_tpp_Septem	2019-12-15_studyZone_	2019-12-15_studyZone_	2019-12-15plot_
19-Oct	October	2019	2019-07-15_latLon_assig	2019-12-15_tpp_Octobe	2019-12-15_studyZone_	2019-12-15_studyZone_	2019-12-15_Octo
19-Nov	November	2019	2019-07-15_latLon_assig	2019-12-15_tpp_Novem	2019-12-15_studyZone_	2019-12-15_studyZone_	2019-12-15_Nov
19-Dec	December	2019	2019-07-15_latLon_assig	2020-1-19_tpp_Decembe	2020-01-19_studyZone_	2020-01-19_studyZone_	2020-01-19plot_

Reference Info
for captions etc.