dockr 0.8.6 is now available on CRAN. dockr is a minimal toolkit to build a  
lightweight Docker container image for your R package, in which the package  
itself is available. The Docker image seeks to mirror your R session as close as  
possible with respect to R specific dependencies. Both dependencies on CRAN  
R packages as well as local non-CRAN R packages will be included in the Docker  
container image.

If you want to know, how Docker works, and why you should consider using Docker,  
please take a look at the [Docker website](https://www.docker.com/why-docker).

**Installation**

install the version released on CRAN:

install.packages("dockr")

**Workflow**

When you work on an R project, it is often desirable to organize the code in the  
R package structure. dockr facilitates easy creation of a Docker container  
image that mirrors your current R session and includes all of the R dependencies  
needed to run your R package.

First, load the dockr package.

library(dockr)

In order do create the files, that constitute the Docker image, simply invoke  
the prepare\_docker\_image() function and point to the folder with your package.

The workflow of prepare\_docker\_image() is summarized below:

1. Build and install the package on your system
2. Identify R package dependencies of the package
3. Detect the version numbers of the loaded and installed versions of these  
   packages on your system
4. Write Dockerfile and create all other files needed to build the Docker image

Now, I will let dockr do its magic and create the files for a Docker image  
container, in which dockr is installed together with all of the R package  
dependencies, dockr needs to run.

Beware that the files are created as side-effects of the function call. Since  
my ‘dockr’ package lives in a folder called ‘docker’ misleadingly, I call the  
function like this:

image\_dockr <- prepare\_docker\_image("~/docker",

dir\_image = "~",

dir\_install = "auto")

#> v Deleting existing folder for files for Docker image: ~/dockr\_0.8.6

#> v Creating folder for files for Docker image: ~/dockr\_0.8.6

#> v Creating folder for source packages: ~/dockr\_0.8.6/source\_packages

#> v Creating empty Dockerfile: ~/dockr\_0.8.6/Dockerfile

#> --- Building, installing and loading package...

#> Writing NAMESPACE

#> Writing NAMESPACE

#> --- Writing Dockerfile...

#> v Preparing FROM statement

#> v Identifying and mirroring R package dependencies

#> v Matching dependencies with CRAN packages

#> v Preparing install statements for specific versions of CRAN packages

#> v Preparing install statement for the package itself

#> v Writing lines to Dockerfile

#> v Closing connection to Dockerfile

#> - in R :

#> => to inspect Dockerfile run:

#> dockr::print\_file("~/dockr\_0.8.6/Dockerfile")

#> => to edit Dockerfile run:

#> dockr::write\_lines\_to\_file([lines], "~/dockr\_0.8.6/Dockerfile")

#> - in Shell :

#> => to build Docker image run:

#> cd C:\Users\Lars\Documents\dockr\_0.8.6

#> docker build -t dockr\_0.8.6 .

#> Please note that Docker must be installed in order for you to build image.

Note, argument ‘dir\_image’ decides, where the files for the docker image will  
be saved. ‘dir\_install’ is the directory, where your package will be installed  
on your system. You can choose to install the package in a temporary folder  
by setting dir\_install = tempdir().

Great, all necessary files for the Docker image have been created, and you  
can build the Docker image right away by following the instructions. It is as  
easy as that! Yeah!

**Files for Docker image**

Let us just take a quick look into the folder with the files for the Docker  
image to see the works of dockr.

list.files(image\_dockr$paths$dir\_image)

#> [1] "Dockerfile" "source\_packages"

It contains a Dockerfile and a folder named ‘source\_packages’.

**Dockerfile**

The resulting Dockerfile can be printed with the print\_file() function, that  
comes with dockr:

print\_file(image\_dockr$paths$path\_Dockerfile)

#> # load rocker base-R image

#> FROM rocker/r-ver:3.6.0

#>

#> # install specific versions of CRAN packages from MRAN snapshots

#> RUN R -e 'install.packages("remotes")'

#> RUN R -e 'remotes::install\_version("askpass", "1.1", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("assertthat", "0.2.1", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("brew", "1.0-6", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("clisymbols", "1.2.0", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("commonmark", "1.7", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("crayon", "1.3.4", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("desc", "1.2.0", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("fs", "1.3.1", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("gh", "1.0.1", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("glue", "1.3.1", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("gtools", "3.8.1", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("ini", "0.3.1", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("jsonlite", "1.6", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("magrittr", "1.5", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("memoise", "1.1.0", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("pkgload", "1.0.2", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("prettyunits", "1.0.2", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("ps", "1.3.0", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("rcmdcheck", "1.3.3", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("rprojroot", "1.3-2", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("rstudioapi", "0.10", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("sessioninfo", "1.1.1", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("stringi", "1.4.3", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("stringr", "1.4.0", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("withr", "2.1.2", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("xopen", "1.0.0", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("yaml", "2.2.0", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("backports", "1.1.4", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("callr", "3.2.0", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("cli", "1.1.0", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("clipr", "0.6.0", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("curl", "3.3", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("devtools", "2.0.2", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("digest", "0.6.20", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("git2r", "0.25.2", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("httr", "1.4.0", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("mime", "0.6", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("openssl", "1.3", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("pkgbuild", "1.0.3", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("processx", "3.3.1", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("purrr", "0.3.2", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("R6", "2.4.0", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("Rcpp", "1.0.1", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("remotes", "2.0.4", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("rlang", "0.4.0", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("roxygen2", "6.1.1", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("sys", "3.2", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("usethis", "1.5.0", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("whisker", "0.3-2", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("xml2", "1.2.0", dependencies = FALSE)'

#>

#> # copy source packages (\*.tar.gz) to container

#> COPY source\_packages /source\_packages

#>

#> # install 'dockr' package

#> RUN R -e 'install.packages(pkgs = "source\_packages/dockr\_0.8.6.tar.gz", repos = NULL)'

#>

As you see, the versions of the R packages, that will be installed in the Docker  
container image, are all given explicitly. They will mirror the versions of the  
dependencies, that are in fact loaded or installed on your system. In this way,  
the Docker container image seeks to reflect your current R session as close as  
possible and by doing so create an environment, where you will be able to  
reproduce results from your current R session.

Also note, that CRAN R packages will be installed from relevant  
[MRAN snapshots](https://mran.microsoft.com/) – using the  
remotes::install\_version() function.

**Folder with Source Packages**

The ‘source\_packages’ folder contains the local (non-CRAN) packages, that have  
to be installed in the Docker container image in order for dockr to run.

Since dockr does not depend on any local (non-CRAN) packages,  
source\_packages only contains a source package version of dockr itself,  
i.e.:

list.files(image\_dockr$paths$dir\_source\_packages)

#> [1] "dockr\_0.8.6.tar.gz"

**How to edit Dockerfile further**

If there is need for adding additional lines to/editing the Dockerfile (e.g.  
if you have to install any non-R [dependencies](http://smaakage85.netlify.com/2019/12/21/dockr-easy-containerization-for-r/#nonr), this can  
be achieved with the write\_lines\_to\_file() function. write\_lines\_to\_file()  
enables you to add new lines to the beginning or the end of the Dockerfile.

Let us try it out and write a couple of additional lines to the Dockerfile.

# write three lines to beginning of file.

write\_lines\_to\_file(c("# set maintainer",

"MAINTAINER Lars KJELDGAARD ",

""),

image\_dockr$paths$path\_Dockerfile,

prepend = TRUE,

print\_file = FALSE)

# write lines to the end of the file.

write\_lines\_to\_file(c("# check out [smaakage85.netlify.com](http://smaakage85.netlify.com) >:-]~~"),

image\_dockr$paths$path\_Dockerfile,

prepend = FALSE,

print\_file = FALSE)

Take a look at the resulting Dockerfile.

print\_file(image\_dockr$paths$path\_Dockerfile)

#> # set maintainer

#> MAINTAINER Lars KJELDGAARD

#>

#> # load rocker base-R image

#> FROM rocker/r-ver:3.6.0

#>

#> # install specific versions of CRAN packages from MRAN snapshots

#> RUN R -e 'install.packages("remotes")'

#> RUN R -e 'remotes::install\_version("askpass", "1.1", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("assertthat", "0.2.1", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("brew", "1.0-6", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("clisymbols", "1.2.0", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("commonmark", "1.7", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("crayon", "1.3.4", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("desc", "1.2.0", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("fs", "1.3.1", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("gh", "1.0.1", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("glue", "1.3.1", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("gtools", "3.8.1", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("ini", "0.3.1", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("jsonlite", "1.6", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("magrittr", "1.5", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("memoise", "1.1.0", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("pkgload", "1.0.2", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("prettyunits", "1.0.2", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("ps", "1.3.0", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("rcmdcheck", "1.3.3", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("rprojroot", "1.3-2", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("rstudioapi", "0.10", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("sessioninfo", "1.1.1", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("stringi", "1.4.3", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("stringr", "1.4.0", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("withr", "2.1.2", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("xopen", "1.0.0", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("yaml", "2.2.0", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("backports", "1.1.4", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("callr", "3.2.0", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("cli", "1.1.0", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("clipr", "0.6.0", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("curl", "3.3", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("devtools", "2.0.2", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("digest", "0.6.20", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("git2r", "0.25.2", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("httr", "1.4.0", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("mime", "0.6", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("openssl", "1.3", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("pkgbuild", "1.0.3", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("processx", "3.3.1", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("purrr", "0.3.2", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("R6", "2.4.0", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("Rcpp", "1.0.1", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("remotes", "2.0.4", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("rlang", "0.4.0", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("roxygen2", "6.1.1", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("sys", "3.2", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("usethis", "1.5.0", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("whisker", "0.3-2", dependencies = FALSE)'

#> RUN R -e 'remotes::install\_version("xml2", "1.2.0", dependencies = FALSE)'

#>

#> # copy source packages (\*.tar.gz) to container

#> COPY source\_packages /source\_packages

#>

#> # install 'dockr' package

#> RUN R -e 'install.packages(pkgs = "source\_packages/dockr\_0.8.6.tar.gz", repos = NULL)'

#>

#> # check out [smaakage85.netlify.com](http://smaakage85.netlify.com) >:-]~~

**Dealing with local non-CRAN R package dependencies**

If your package depends on local non-CRAN R packages, dockr will also include  
these packages in the Docker container image. Local non-CRAN R packages must be  
available as source packages ([packageName]\_[packageVersion].tar.gz) in one  
or more user specified local directories. These paths have to be specified in the  
‘dir\_src’ argument, when invoking the prepare\_docker\_image(), e.g.:

# image for my package 'recorder'.

image\_recorder <- prepare\_docker\_image("~/recorder",

dir\_image = "~",

dir\_install = "auto",

dir\_src = c("~/src"))

Note, that you can store **multiple** versions of the same package in your local  
repos. In this way ‘dockr’ comes with a lot of flexibility.

**What about non-R dependencies?**

dockr does *not* deal with any non-R dependencies what so ever at this point.  
In case that, for instance, your package has any Linux specific dependencies,  
you will have to install them yourself in the Docker container image.