



Knowledge grows



Knowledge grows

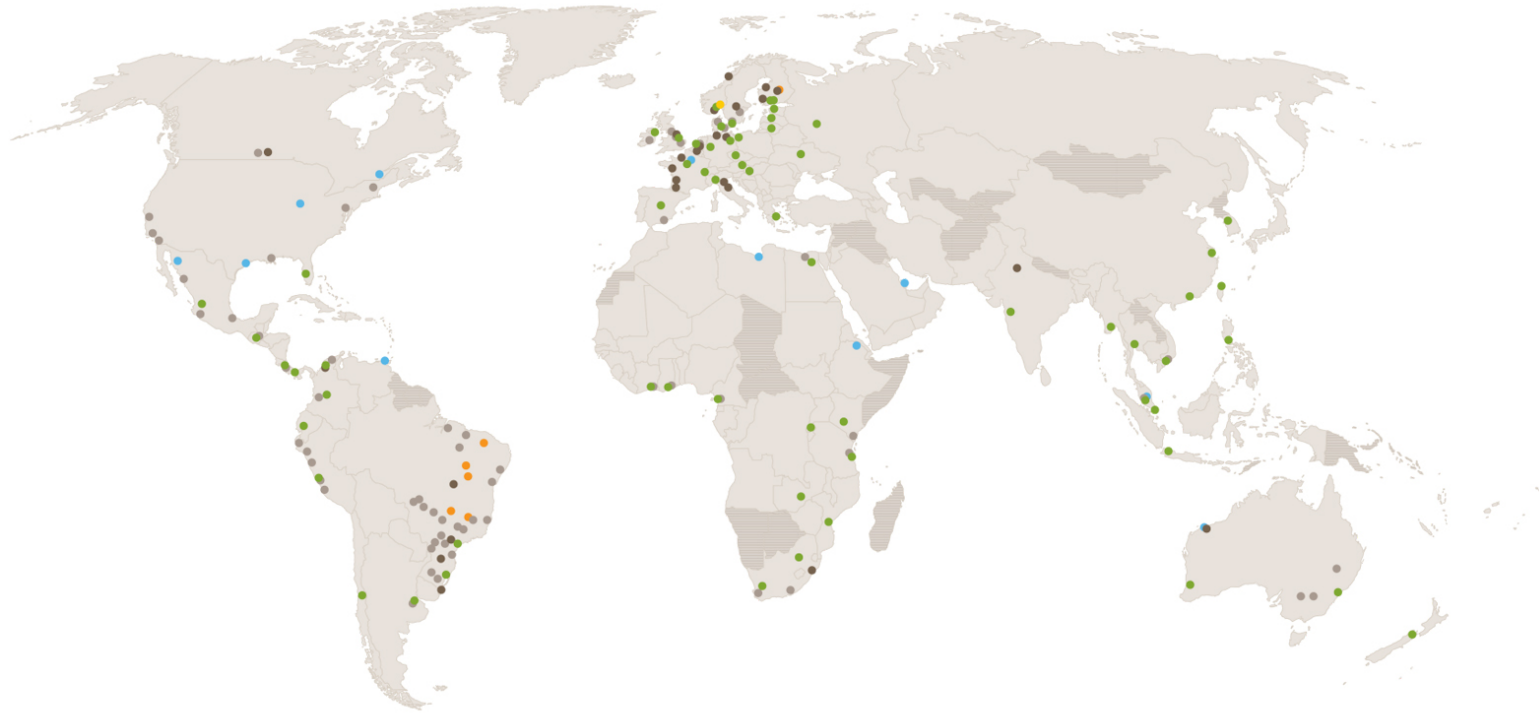
Richard Brosi _



Yara _

Crop Nutrition





Yara _ | 16.000+ employees | USD 11.4bn revenue in 2017 |
| farming (fertilizer) & industrial chemicals |

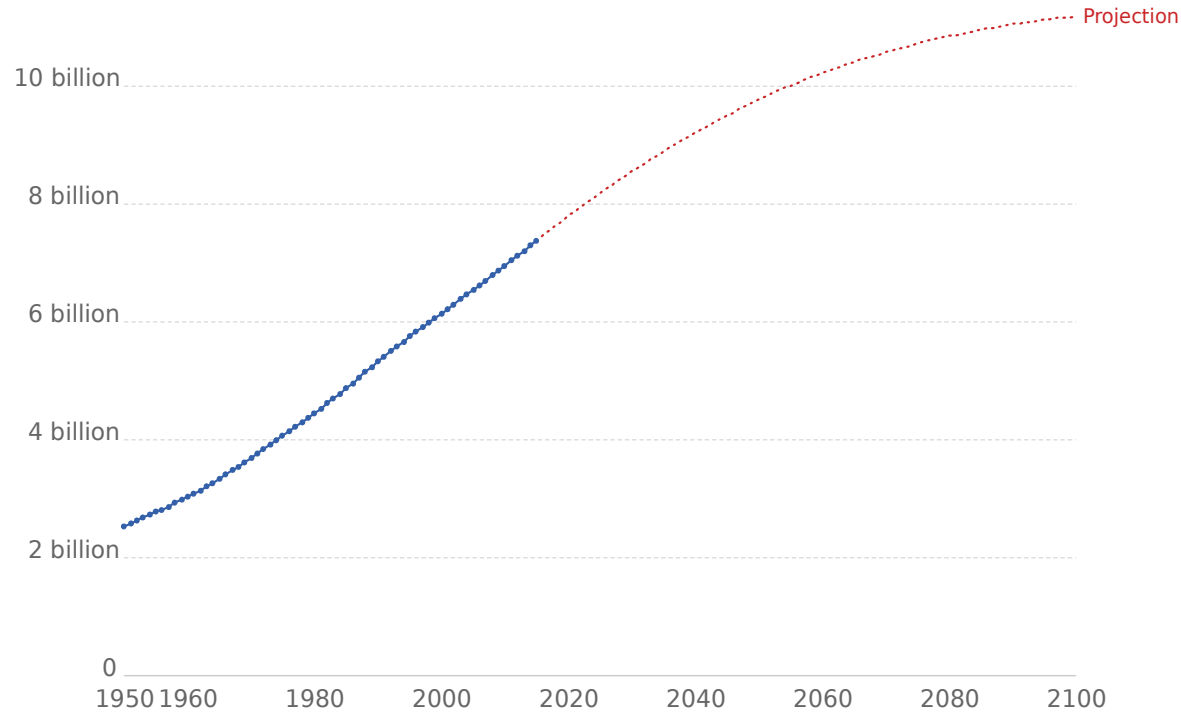


It's about food_

Population projection by the UN, World

Shown is the total population since 1950 and the Medium Variant projections by the UN Population Division until 2100.

Our World
in Data



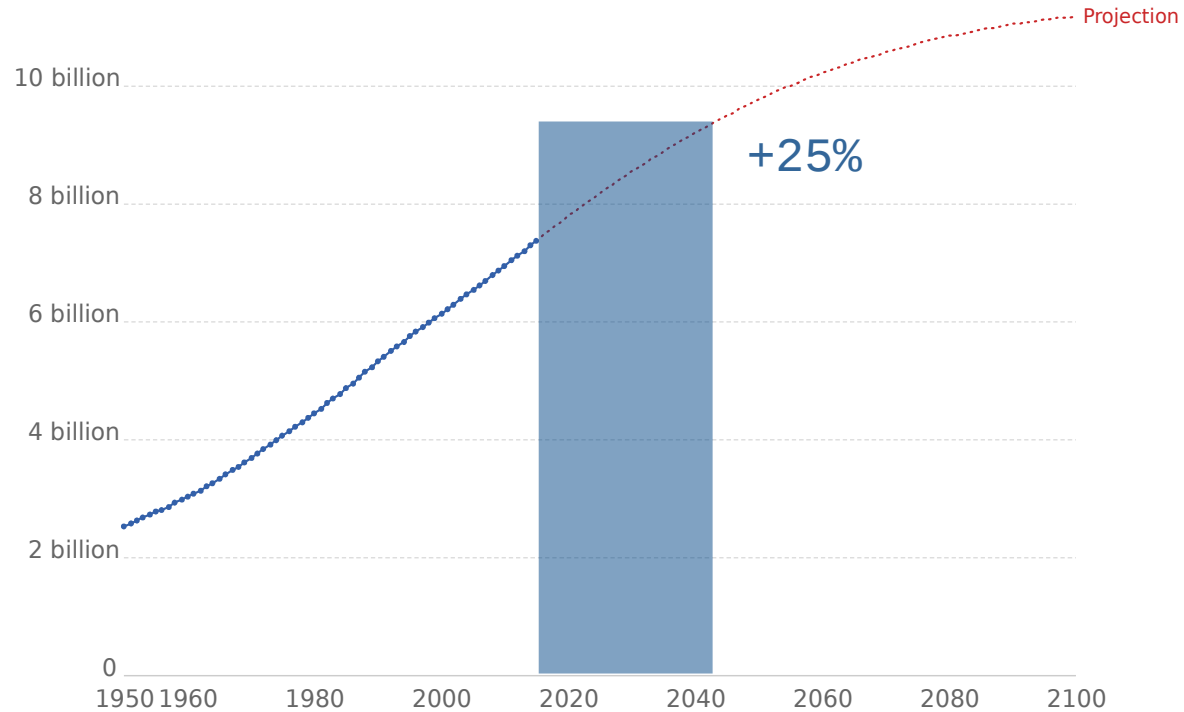
Source: UN Population Division (2017 Revision)

OurWorldInData.org/future-population-growth/ • CC BY

Population projection by the UN, World

Shown is the total population since 1950 and the Medium Variant projections by the UN Population Division until 2100.

Our World
in Data

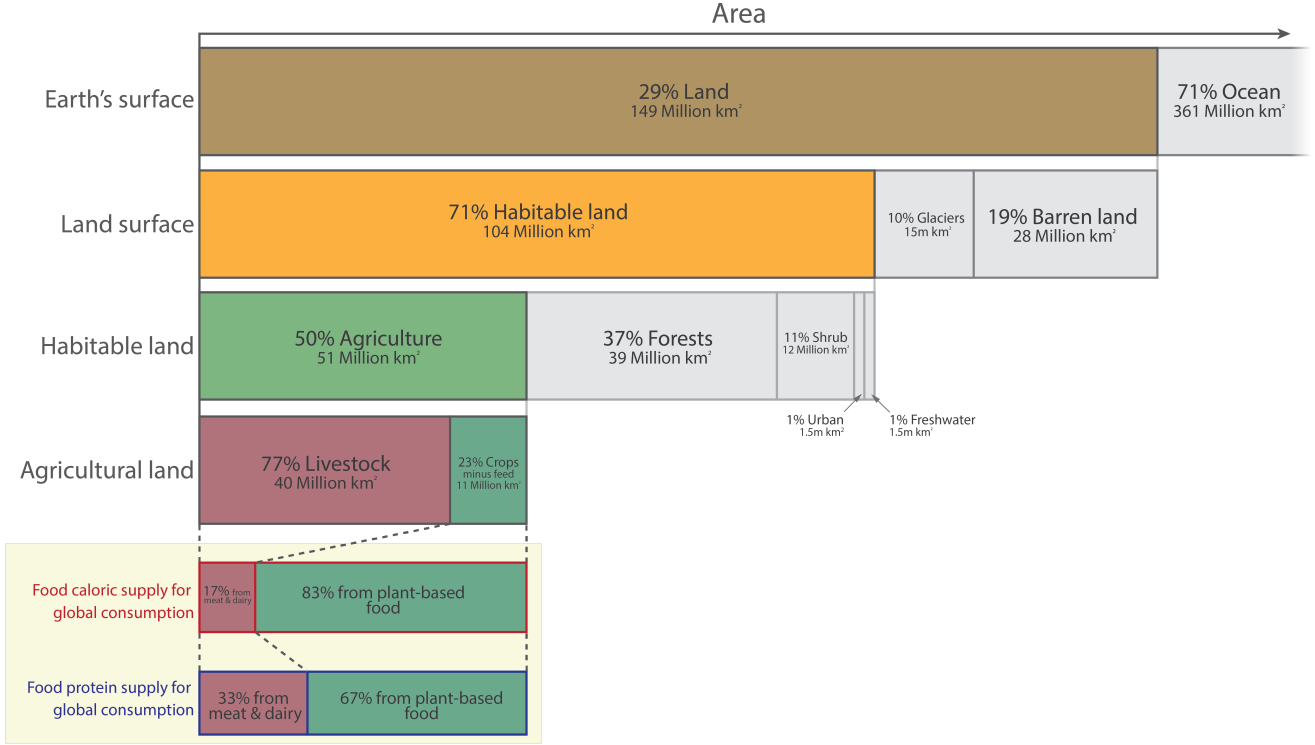


Source: UN Population Division (2017 Revision)

OurWorldInData.org/future-population-growth/ • CC BY

Global surface area allocation for food production

The breakdown of Earth surface area by functional and allocated uses, down to agricultural land allocation for livestock and food crop production, measured in millions of square kilometres. Area for livestock farming includes grazing land for animals, and arable land used for animal feed production. The relative production of food calories and protein for final consumption from livestock versus plant-based commodities is also shown.



Data source: based on UN Food and Agricultural Organization (FAO) Statistics. The data visualization is available at [OurWorldinData.org](https://ourworldindata.org). There you find research and more visualizations on this topic.

Licensed under CC-BY-SA by the authors Hannah Ritchie and Max Roser.



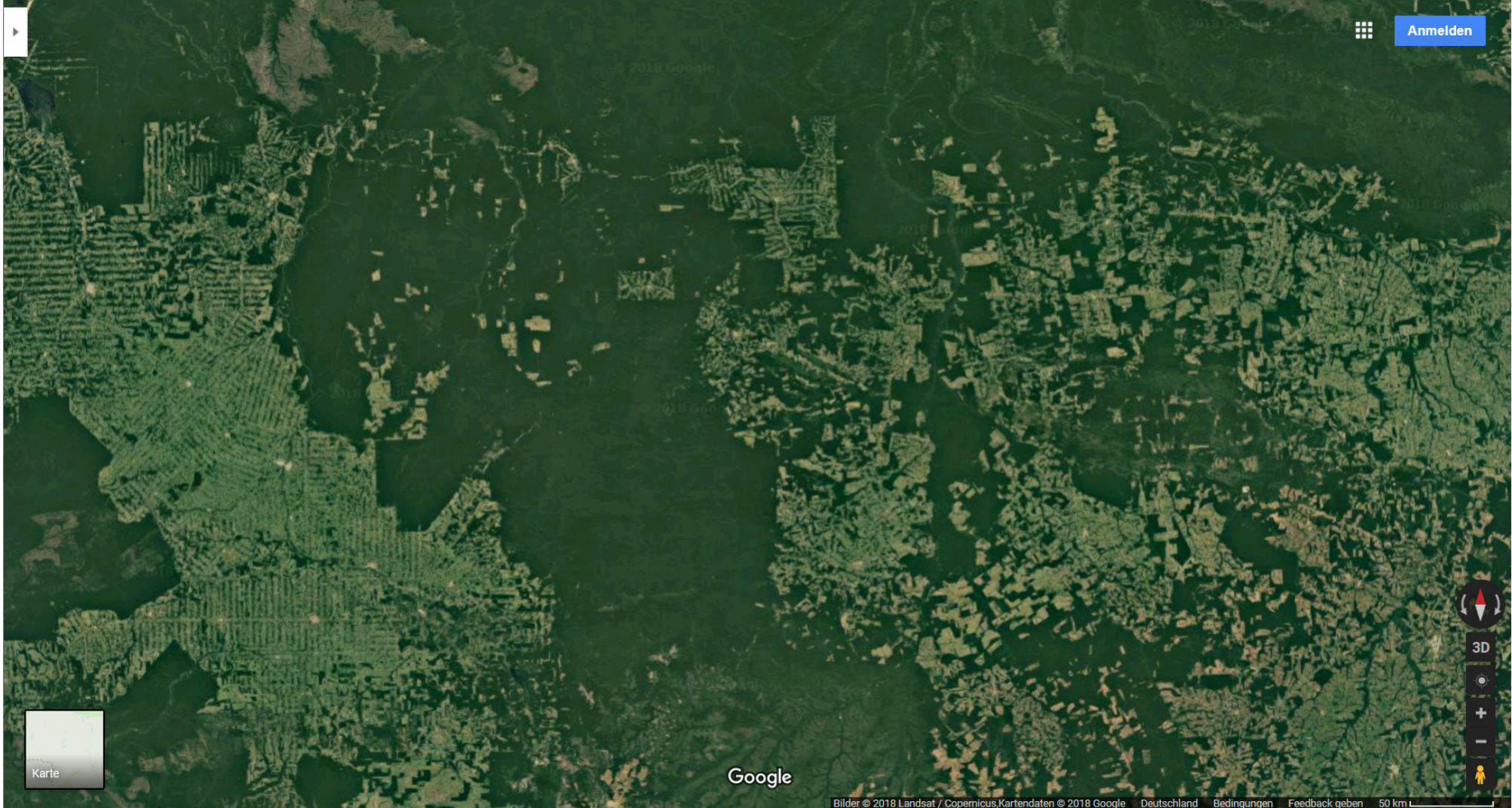
Anmelden



Google

Bilder © 2018 Landsat / Copernicus, Data SIO, NOAA, U.S. Navy, NGA, GEBCO, Kartendaten © 2018 Google Deutschland Bedingungen Feedback geben 200 km

<https://www.google.com/maps>



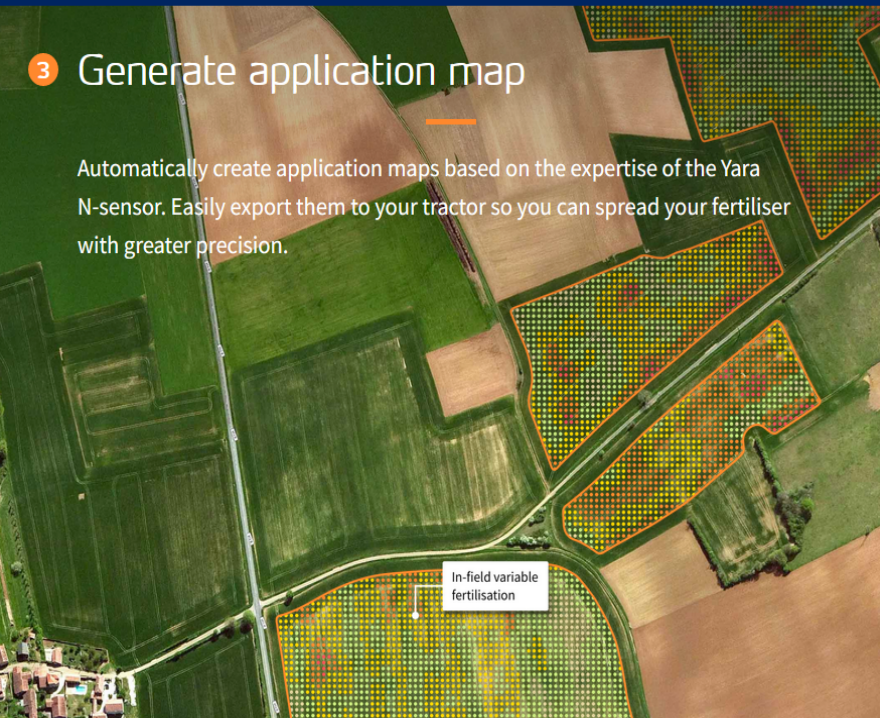
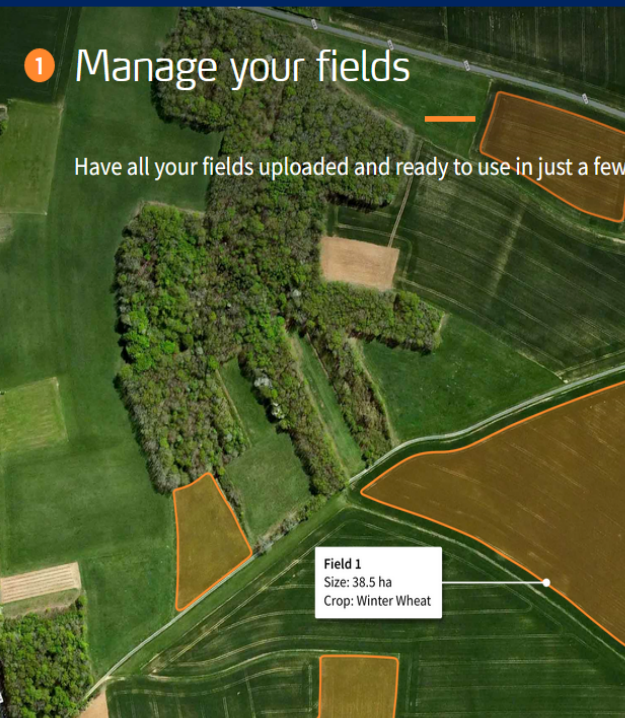
<https://www.google.com/maps>



It's about food & **sustainable agriculture**_

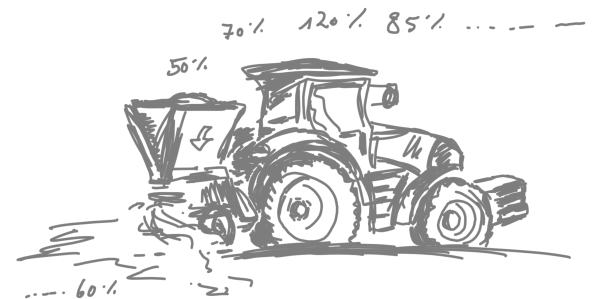


_ Yara | Digital Transformation



Precise Fertilization

- Satellite image analysis
- Apply just the optimum amount, no excess



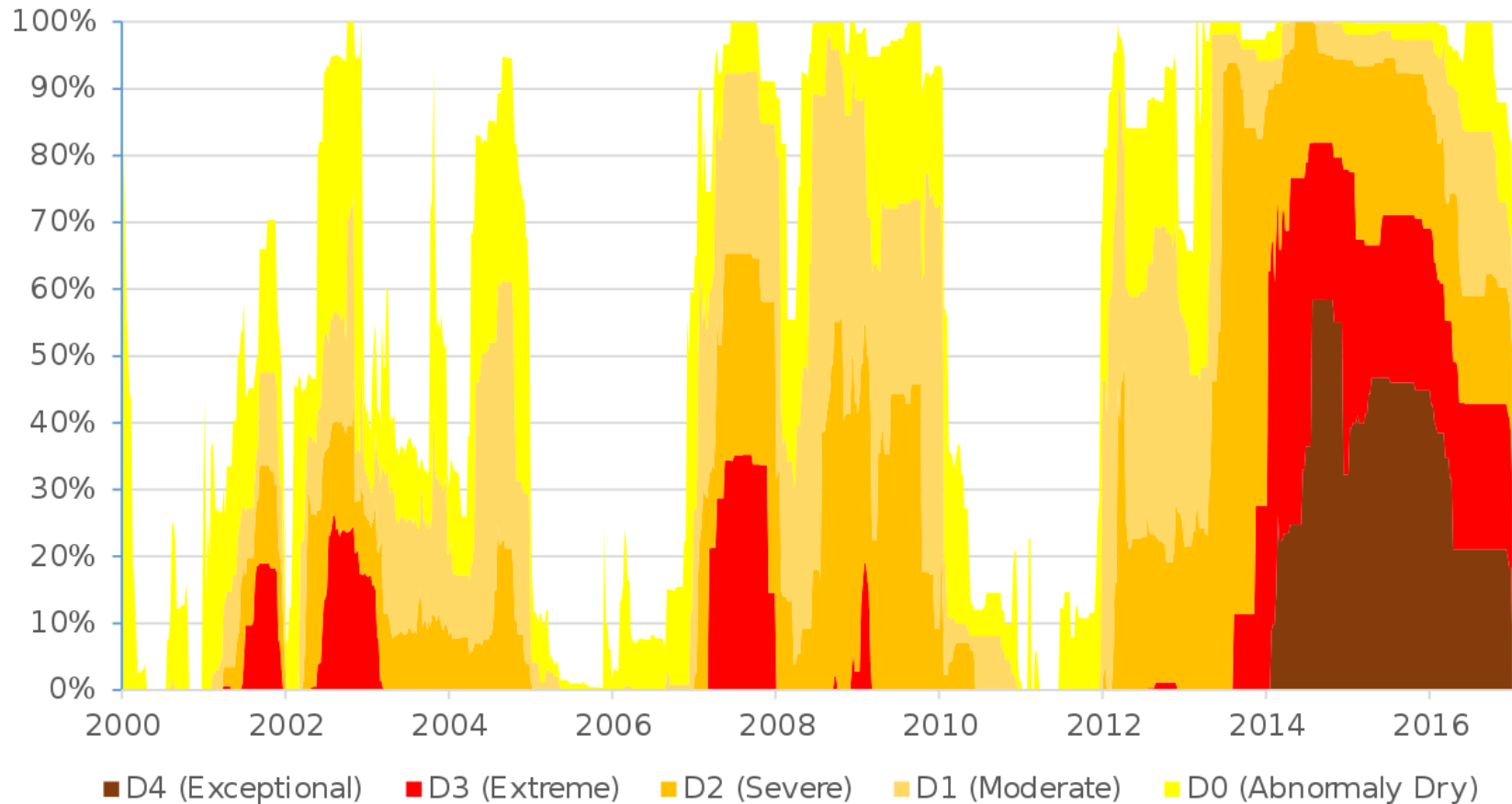
... currently covering more than 800'000 ha



Precise Irrigation







The Water Crises Aren't Coming—They're Here

For eons, the earth has had the same amount of water—no more, no less. What the ancient Romans used for crops and Nefertiti drank? It's the same stuff we bathe with. Yet with more than seven billion people on the planet, experts now worry we're running out of usable water. The symptoms are here: multiyear droughts, large-scale crop failures, a major city—Cape Town—on the verge of going dry, increasing outbreaks of violence, fears of full-scale water wars. The big question: How do we keep the H₂O flowing?



BY ALEC WILKINSON AUG 23, 2018

12.1K



Esquire

STYLE

NEWS

POLITICS

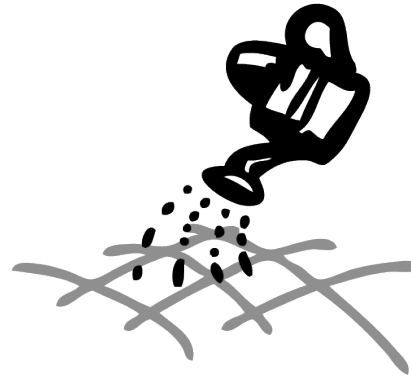
ENTERTAINMENT

FOOD & DRINK

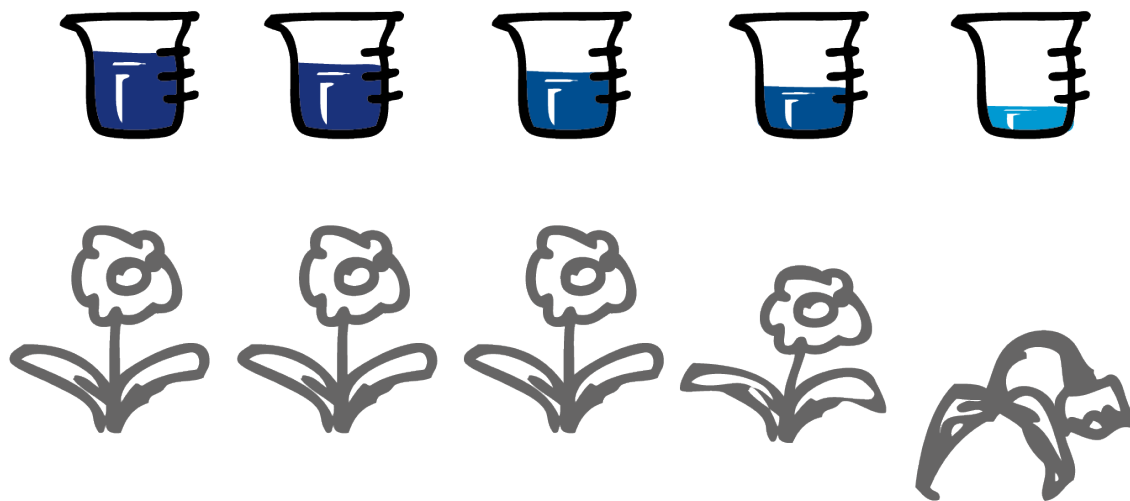
- > “We’re overdrafting groundwater”
- >
- > “You have to start doing things differently.”
- >
- > “There’s no more new water.”



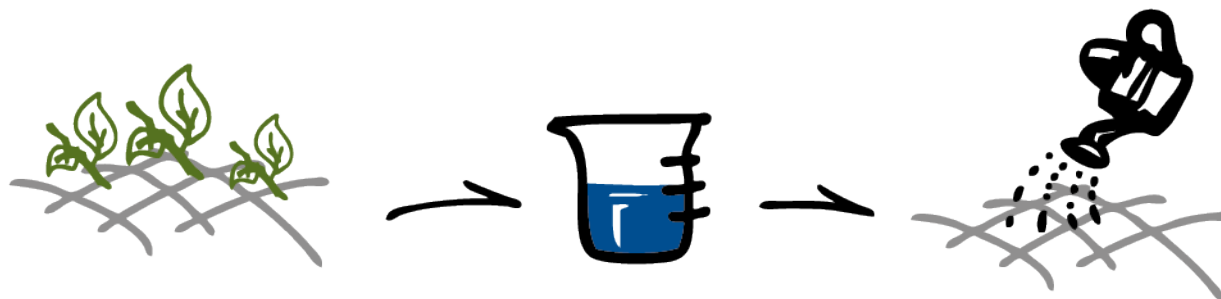
_ Adaptive Irrigation & Drought Management



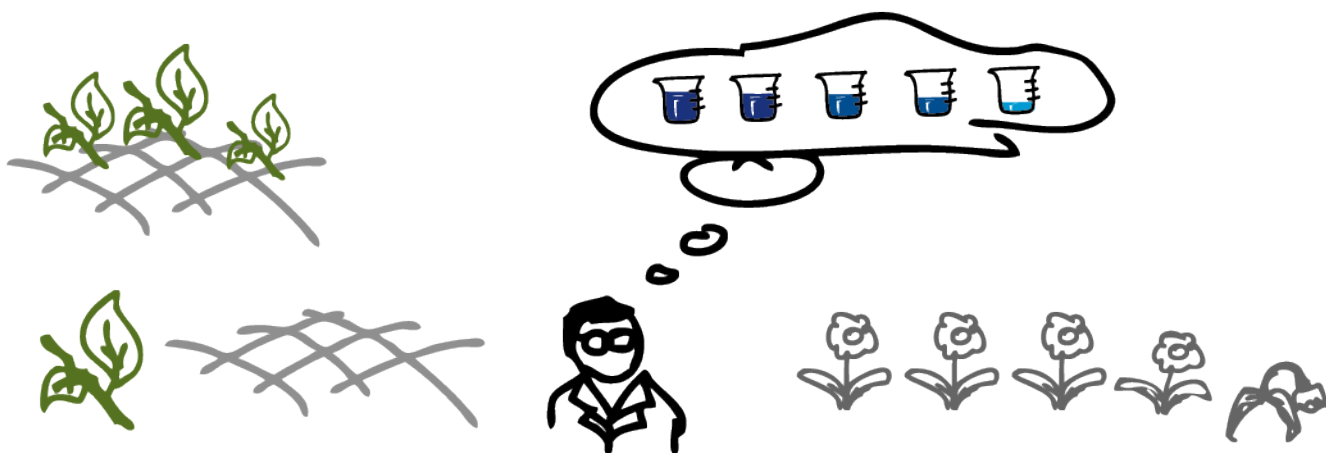
Aim: Precise & adaptive irrigation



_ How thirsty is the plant? - the plant's "water status"

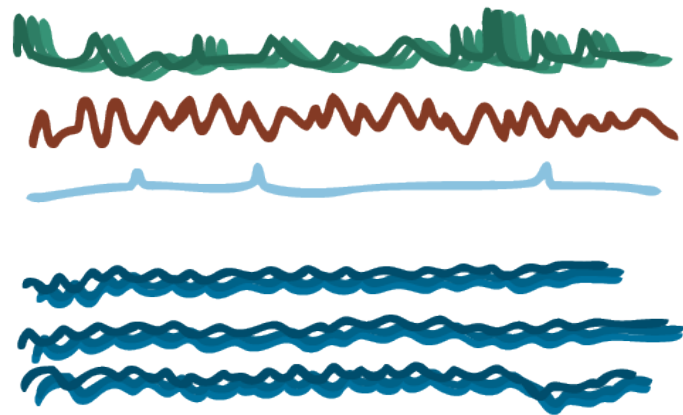


Using the water status for irrigation recommendations_



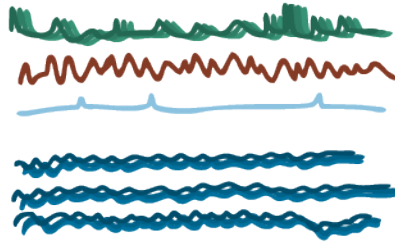


How to determine the water status automatically?

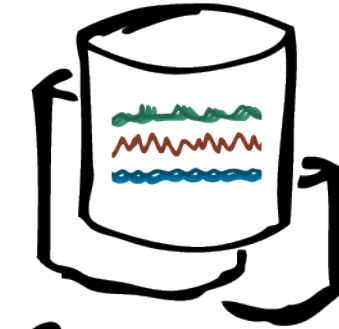




_ Using R in a mid-sized data scenario



IoT



SQL



Why R?

People in the Agronomic Sector tend to speak R*

Existing Body of Code

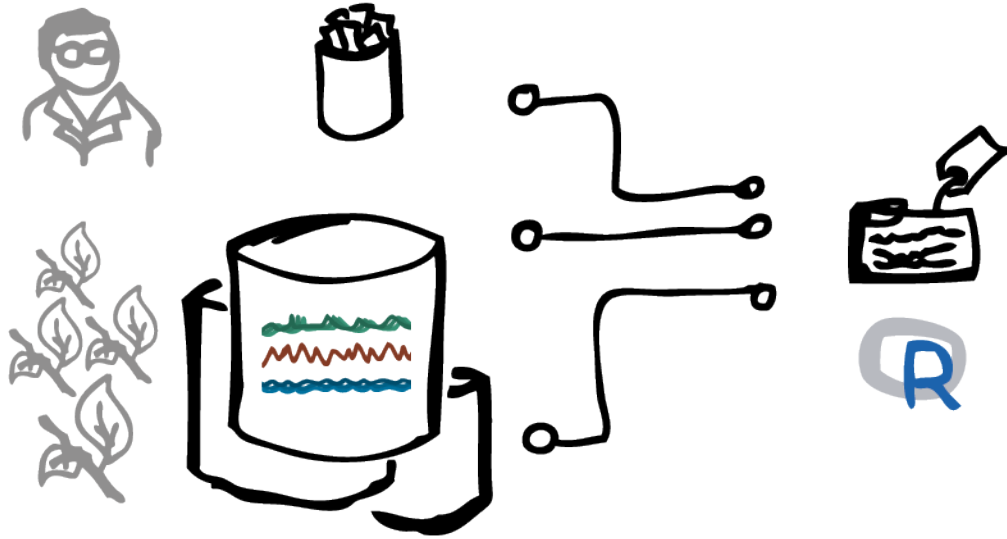
Broad array of CRAN packages in statistics, agronomy & biology

Very good documentation

Mid-sized data volumes

* a little bit _





Build model(s) from the labeled data_



R Interface to Python

The **reticulate** package provides a comprehensive set of tools for interoperability between Python and R. The package includes facilities for:

- Calling Python from R in a variety of ways including R Markdown, sourcing Python scripts, importing Python modules, and using Python interactively within an R session.
- Translation between R and Python objects (for example, between R and Pandas data frames, or between R matrices and NumPy arrays).
- Flexible binding to different versions of Python including virtual environments and Conda environments.



Reticulate embeds a Python session within your R session, enabling seamless, high-performance interoperability. If you are an R developer that uses Python for some of your work or a member of data science team that uses both languages, reticulate can dramatically streamline your workflow!

Python packages

You can install any required Python packages using standard shell tools like `pip` and `conda`. Alternately, reticulate includes a set of functions for managing and installing packages within `virtualenvs` and `Conda` environments. See the article on [Installing Python Packages](#) for additional details.

Calling Python

There are a variety of ways to integrate Python code into your R projects:

1. **Python in R Markdown** — A new Python language engine for R Markdown that supports bi-directional communication between R and Python (R chunks can access Python objects and vice-versa).
2. **Importing Python modules** — The `import()` function enables you to import any Python module and call it's functions directly from R.
3. **Sourcing Python scripts** — The `source_python()` function enables you to source a Python script the same way you would `source()` an R script (Python functions and objects defined within the script become directly available to the R session).
4. **Python REPL** — The `repl_python()` function creates an interactive Python console within R. Objects you create within Python are available to your R session (and vice-versa).

Why reticulate?

The package enables you to *reticulate* Python code into R, creating a new breed of project that weaves together the two languages.

Arrays

R matrices and arrays are converted automatically to and from NumPy arrays.

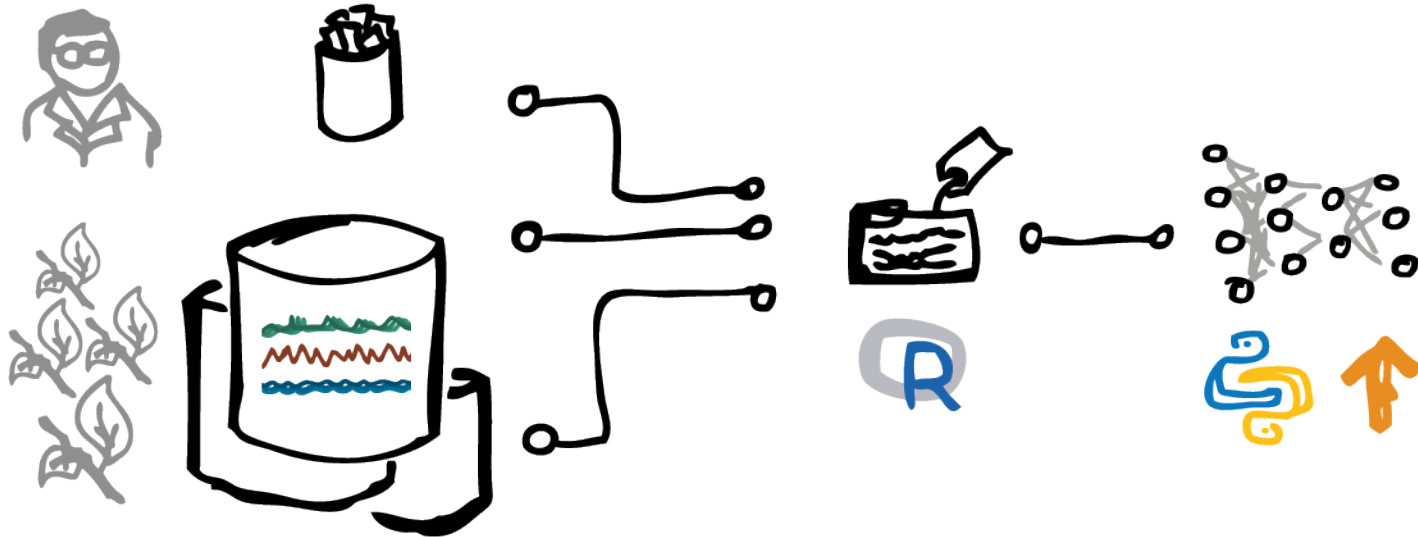
When converting from R to NumPy, the NumPy array is mapped directly to the underlying memory of the R array (no copy is made). In this case, the NumPy array uses a column-based in memory layout that is compatible with R (i.e. Fortran style rather than C style). When converting from NumPy to R, R receives a column-ordered copy of the NumPy array.

Data Frames

R data frames can be automatically converted to and from Pandas DataFrames. By default, columns are converted using the same rules governing R array <-> NumPy array conversion, but a couple extensions are provided:

R	Python
Factor	Categorical Variable
POSIXt	NumPy array with dtype = <code>datetime64[ns]</code>

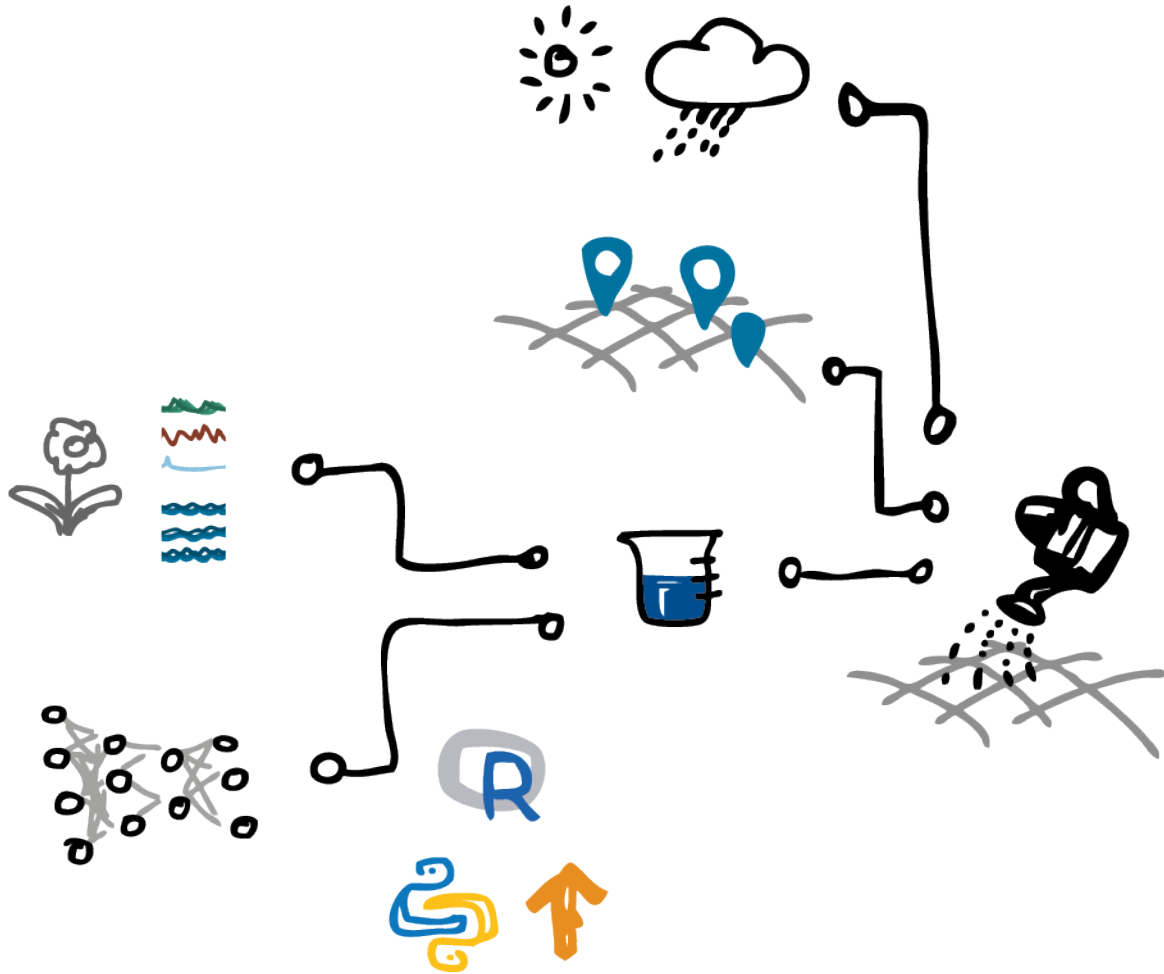
<https://rstudio.github.io/reticulate/>



<https://rstudio.github.io/reticulate/>

=> convenient access to Tensorflow, Keras, ...

Build the model(s)_

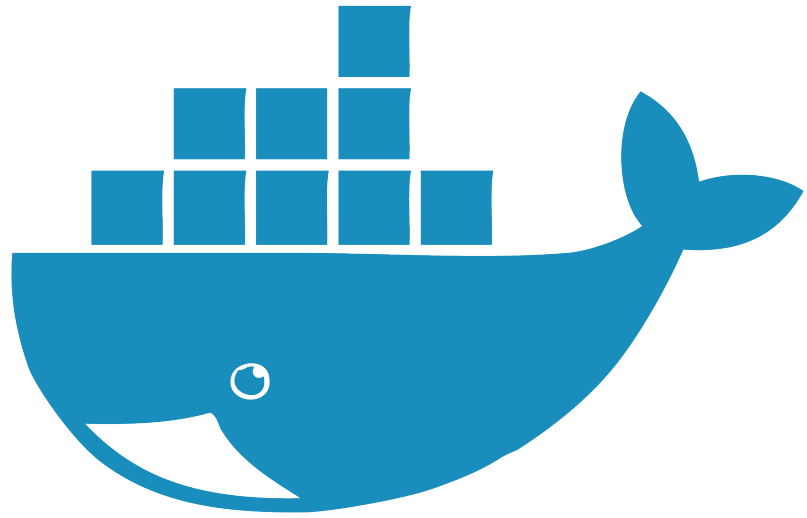


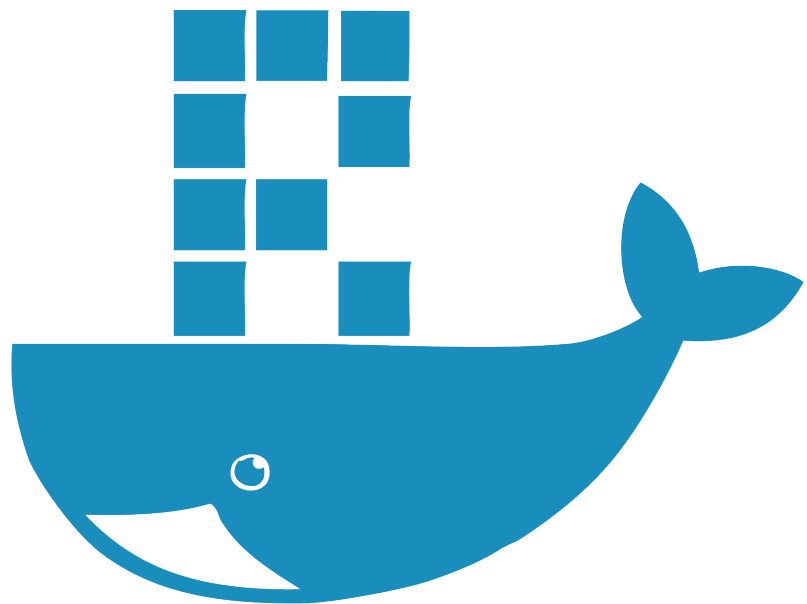


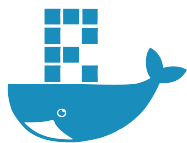
Reproducibility / Consistent environments

Portability / Scalability

All Is Code







Home · rocker-org/rocker Wiki X

GitHub, Inc. (US) https://github Search

rocker-org / rocker

Code Issues 3 Pull requests 0 Projects 0 Wiki Pulse

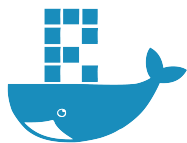
Base Docker Containers

image	description	size	metrics	build status
r-base	Current R via apt-get with <code>debian:testing & unstable</code> repos	276.6MB 11 layers	docker pulls 3M	docker build automated
r-devel	R-devel added side-by-side onto r-base (using alias <code>RD</code>)	1.2GB 20 layers	docker pulls 5k	docker build automated
drd	lighter r-devel, built not quite daily	772.9MB 16 layers	docker pulls 5k	docker build automated
r-ver	Specify R version in docker tag. Builds on <code>debian:stable</code>	237.9MB 8 layers	docker pulls 47k	docker build automated

Versioned stack (builds on r-ver)

These images build on `rocker/r-ver`. Each of these include tags to specify the desired version of R, from `3.1.0` - `3.4.1`, `:latest` and `:devel`. See [rocker-versioned](#) repo for details.

image	description	size	metrics	build status
rstudio	Adds rstudio	354MB 21 layers	docker pulls 2M	docker build automated
tidyverse	Adds tidyverse & devtools	629.9MB 22 layers	docker pulls 247k	docker build automated



CRAN Time Machine

For the purpose of **reproducibility**, MRAN hosts **daily snapshots** of the CRAN R packages and R releases as far back as Sept. 17, 2014.

Use our **Time Machine** to browse CRAN contents from the past.

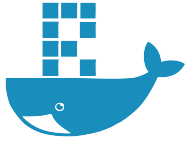
[Browse Snapshots](#)

<https://mran.microsoft.com/>

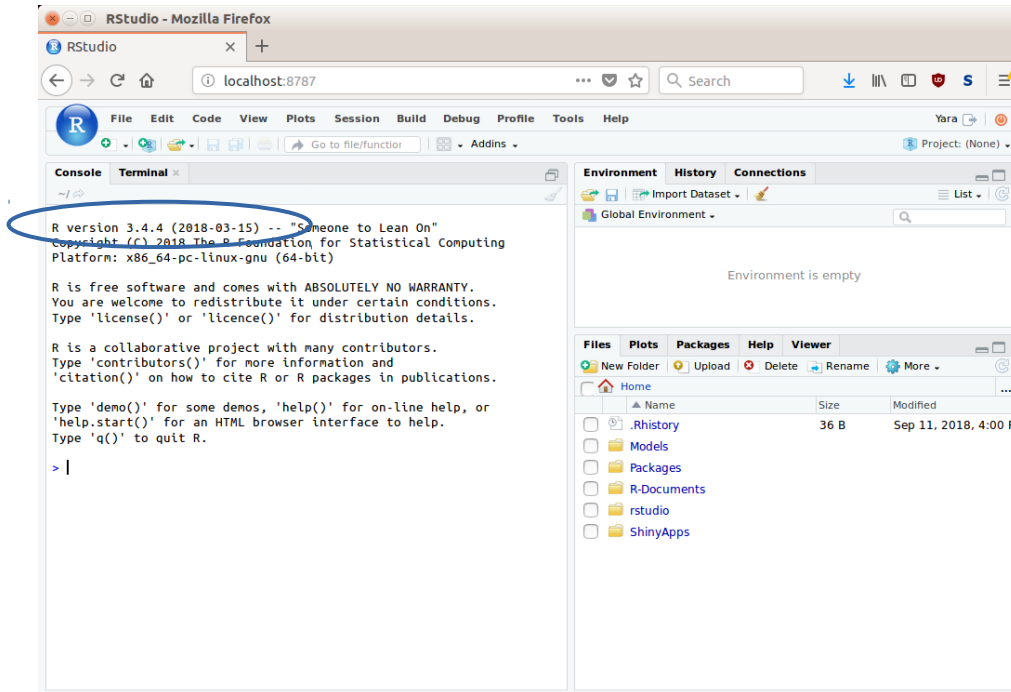
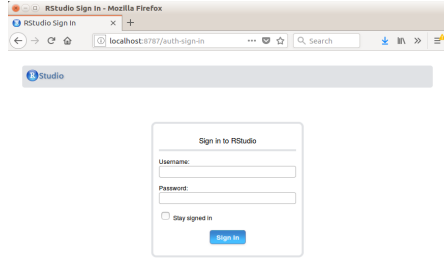
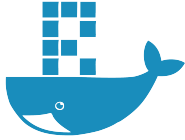
The screenshot shows the Docker Hub page for rocker-org/docker. The page is titled "Base Docker Containers" and lists several Docker images. Below this, there is a section for "Versioned stack (builds on r-ver)" which includes a table of images like rstudio and tidyverse.

image	description	size	metrics	build status
r-base	Current R via apt-get with debian:testing & unstable repos	276.6MB 11 layers	docker pulls 3M	docker build automated
r-devel	R-devel added side-by-side onto r-base (using alias RD)	1.2GB 20 layers	docker pulls 5k	docker build automated
drd	lighter r-devel, built not quite daily	771.9MB 16 layers	docker pulls 5k	docker build automated
r-ver	Specify R version in docker tag. Builds on debian:stable	237.9MB 8 layers	docker pulls 47k	docker build automated

image	description	size	metrics	build status
rstudio	Adds rstudio	354MB 21 layers	docker pulls 2M	docker build automated
tidyverse	Adds tidyverse & devtools	629.9MB 22 layers	docker pulls 247k	docker build automated



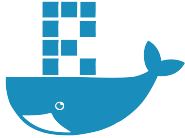
R-docker for both development and production



Dockerfile

```
FROM rocker/tidyverse:3.4.4
RUN apt-get update && \
    apt-get -y -no-inst... && \
    install2.r -deps TRUE \
        DBI \
        \
        ...
        logging \
        \
        testthat \
        \
        ...

$ docker build -f Dockerfile \
    -t tidyrocker .
```

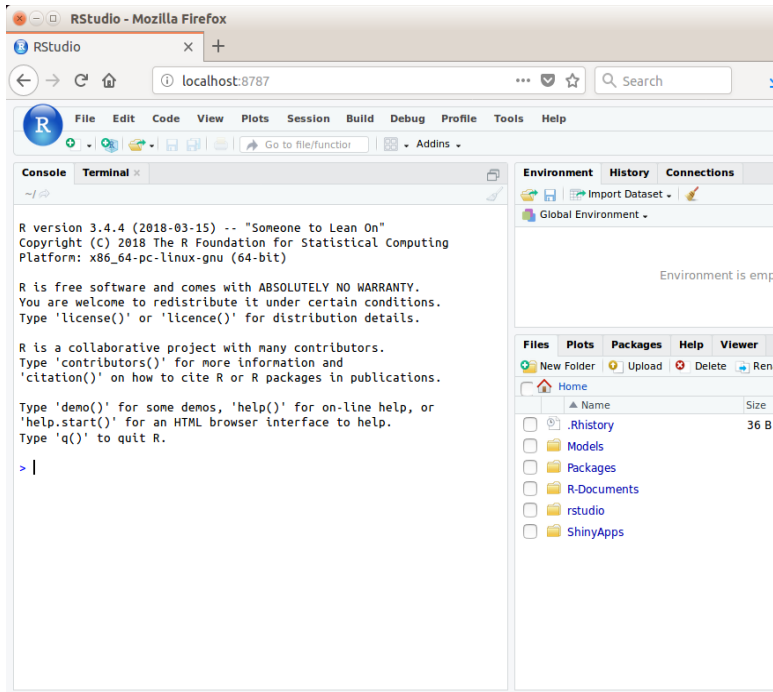


CAVEAT 1:

...this password is an

Environment Variable

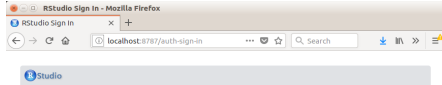
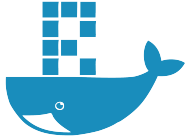
`$ docker inspect MyRStudio`



<https://github.com/rocker-org/rocker/wiki/FAQ>

I see RStudio using `http` instead of `https`. Is my password being transferred in plain text?

Nope. RStudio encrypts your credentials (using a JavaScript RSA implementation) It is possible to run RStudio server behind a proxy or over ssh tunnel instead (e.g. with all other ports behind a firewall) for additional security; see the RStudio documentation.



CAVEAT 1:
...this password is an

Environment Variable

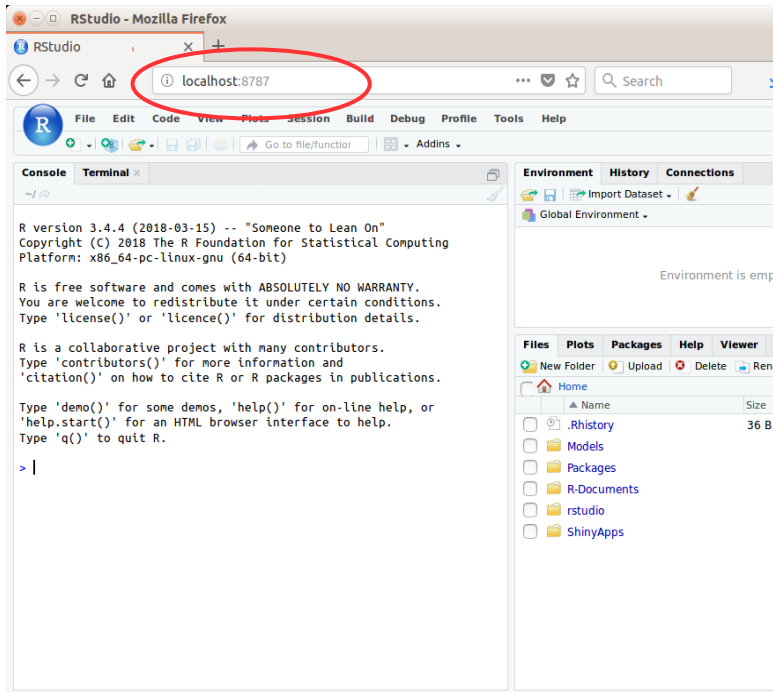
`$ docker inspect MyRStudio`

CAVEAT 2:
...no HTTPS

a) http behind ip filter/firewall

OR

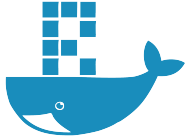
b) nginx as SSL reverse proxy



`https://myserver:80`



`http://myintserver:8787`



<https://stackoverflow.com/questions/41422826/install-python-of-specific-version-system-wide-with-pyenv/43321411>

<https://github.com/pyenv/pyenv>

<https://github.com/jaehyeon-kim/rocker-extra>

The screenshot shows a web browser window displaying the GitHub repository page for `jaehyeon-kim/rocker-extra`. The page title is "Rocker Extra". Below the title, it states: "Extra docker images from `rocker/tidyverse`. All images will be found in [DockerHub](#)."

The next section is titled "Structure of `rocker` project". It says: "All `rockerextra` images will be from `rocker/tidyverse`."

Below this, there is a code block showing the structure of the `rocker` project:

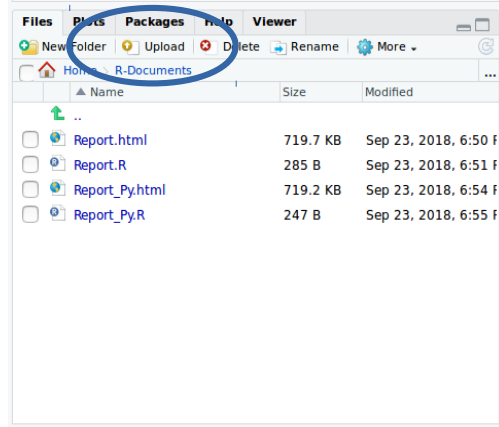
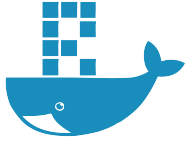
```
debian:stretch
--> rocker/r-ver
--> rocker/rstudio
--> rocker/tidyverse
```

The following section is titled "Structure of `rockerextra` project". It shows the structure of the `rockerextra` project:

```
rocker/tidyverse
--> rockerextra/spark
--> rockerextra/python
--> rockerextra/ssh
```

Below the code blocks, there is a list of project components:

- `rockerextra/spark`
 - hadoop 2.8.2 + spark 2.2.1
 - sources from [SingularitiesCR/hadoop-docker](#) and [SingularitiesCR/spark-docker](#)
 - see for [further details](#)
- `rockerextra/python` - python 3.6.4 + [pyenv](#)
- `rockerextra/ssh` - SSH connection + emacs 25
 - `docker run --name ssh -d -p 2222:22 -p 8787:8787 -e SSH_KEY="$(cat ./sample/id_rsa.pub)" rockerextra/p`



Use upload/download option

OR

<https://github.com/rocker-org/rocker/wiki/Sharing-files-with-host-machine>

RStudio Server - persisting files_

Linux

As docker runs natively on Linux, you can always link any volume to the container with the `-v` or `--volume` flag, as described in the [docker documentation](#). Note that this overwrites anything on the container that already exists in the target location, and will create the necessary directories if they do not already exist. We recommend running as a non-root user and linking to a subdirectory in the that user's home directory as shown below.

Avoiding permission changes when sharing volumes

By default, our docker containers run as the root user. (See [managing users in docker](#) for details.) Files created or modified by the container will thus become owned by the root user, even after quitting the container. To avoid this problem, it is necessary to run the container using a non-root user.

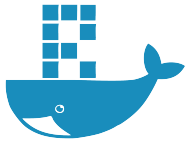
RStudio-server logins require a non-root user whether or not the container is sharing any volumes with the host. A default user is created automatically when these containers are run with the default command. When sharing volumes with the host, it may be necessary to make sure that the UID of the user created (whether the it is the default user, `rstudio` or a custom `username`) matches the UID of the host user (the username is irrelevant, only the `UID` must match). If your host machine has only one non-root user, chances are this is already fine, though you can still set the UID explicitly as described below.

Anyone running our containers as a non-root user can link a directory, including ones at or above `~/`, to the corresponding user's directory on the container. The container will only ever have one user.

If the host machine user has a UID other than 1000 (or 0, for root), the user should specify their UID when running docker, e.g.

```
docker run -d -P -v $(pwd):/home/$USER/foo -e USERID=$UID rocker/rstudio
```

to avoid changing the permissions in the linked volume on the host. This is designed for `rstudio`-based logins and runs only when the container is executed without a default command. (Note, optionally a custom user name can be given with the environmental variable argument `-e USER=someuser`, instead of the default `rstudio` user, but this is purely an aesthetic change. For matching permissions, merely ensure the UID is consistent.



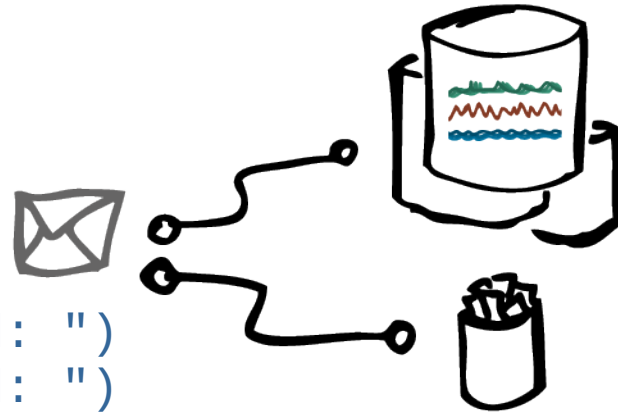
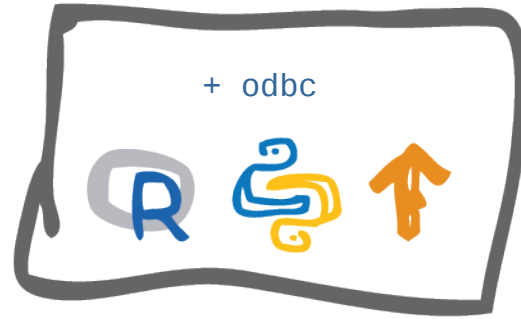
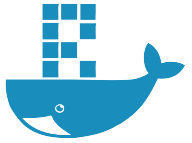
```
Another_Dockerfile_To_Add_MS_ODBC
```

```
FROM customrockerimage
```

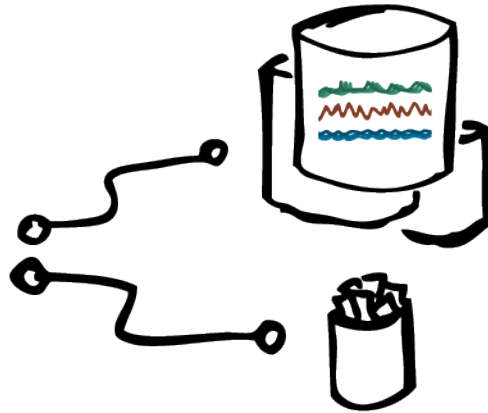
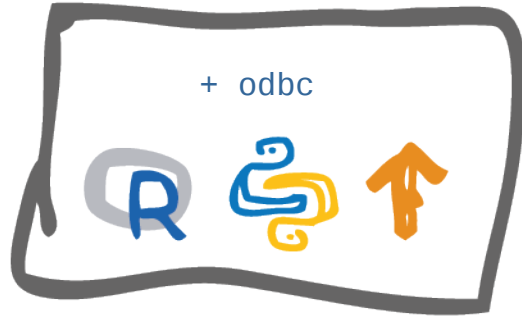
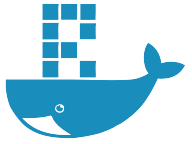
```
RUN apt-get update && ... \  
curl https://packages.microsoft.com/keys/microsoft.asc | apt-key add - && \  
curl https://packages.microsoft.com/config/debian/9/prod.list > \  
    /etc/apt/sources.list.d/mssql-release.list && \  
apt-get update && \  
ACCEPT_EULA=Y apt-get -y install \  
    msodbcsql17 \  
    unixodbc-dev && \  
        install2.r -deps TRUE \  
        RODBC
```

```
COPY odbc.ini /etc/odbc.ini
```

```
$ docker build -f Another_Dockerfile_To_Add_MS_ODBC -t customrockerimage/msodbc .
```



```
rstudioapi::askForPassword("pwd: ")  
  getPass::getPass("pwd: ")
```





Some things I found useful...



RStudio - Mozilla Firefox

RStudio

localhost:8787

File Edit Code View Plots Session Build Debug Profile Tools Help

Report_Py.R x

```
1 # Generate a Report from an existing R Script -----
2 |
3 suppressMessages(library(reticulate))
4 |
5 py_run_string("
6 def timestwo(a):
7     b = a * 2
8     return b
9 ")
10 |
11 py_run_string(paste(
12 "answer = timestwo(",
13     mtcars[1, 1],
14     ")"))
15 |
16 mtcars[1, ]
17 |
18 py$answer
```

Environment History Connections

Global Environment

Environment is empty

Files Plots Packages Help Viewer

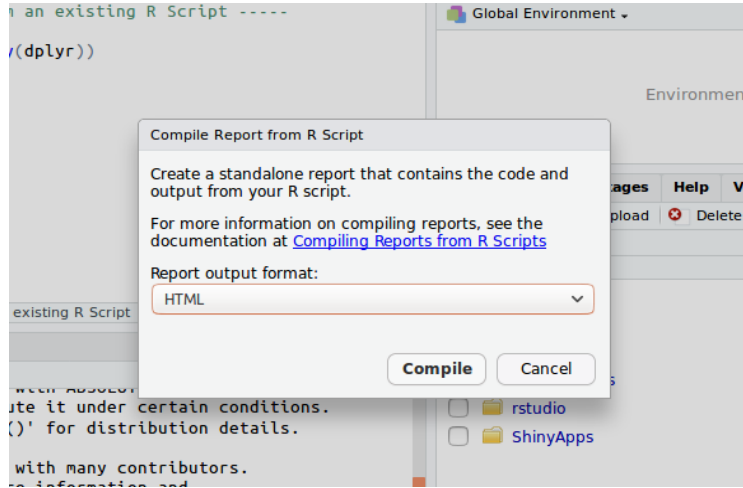
New Folder Upload Delete Rename More

Home > R-Documents

	Name	Size	Modified
	..		
	Report.html	719.7 KB	Sep 23, 2018, 6:50 F
	Report.R	285 B	Sep 23, 2018, 6:51 F
	Report_Py.html	719.2 KB	Sep 23, 2018, 6:54 F
	Report_Py.R	247 B	Sep 23, 2018, 6:55 F

2:1 Generate a Report from an existing R Script R Script

Console



~/R-Documents/Report_Py.html - Mozilla Firefox
localhost:8787/?view=rmarkdown
Report_Py.html Open in Browser Find Publish

Report_Py.R

Yara
Sun Sep 23 16:56:44 2018

```
# Generate a Report from an existing R Script -----  
  
suppressMessages(library(reticulate))  
  
py_run_string("  
def timestwo(a):  
    b = a * 2  
    return b  
")  
  
py_run_string(paste(  
    "answer = timestwo(",  
    mtcars[1, 1],  
    ")")  
  
mtcars[1, ]  
  
##           mpg cyl disp  hp drat   wt  qsec vs am gear carb  
## Mazda RX4  21   6  160 110  3.9 2.62 16.46  0  1   4    4  
  
py$answer  
  
## [1] 42
```

```
...  
options(shiny.reactlog=TRUE)  
...
```

→ Press Ctrl + F3

```
input$in.daterange = Date[1:2], format: "2018-06-07" "2018-09-05"
```

time elapsed: 0.14 ms
[isolate]

```
observeEvent(input$in.btn.Live = Classes 'integer', 'shinyActionButtonValue' int
```

[isolate]

```
observeEvent(input$in.btn.  
input$in.btn.Test = Classes 'integer', 'shinyActionButtonValue' int
```

- Normal
- Invalidated
- Running



<https://shiny.rstudio.com/articles/debugging.html>

```
<shiny-server.conf>

run_as shiny;

app_init_timeout 240;
app_idle_timeout 120;

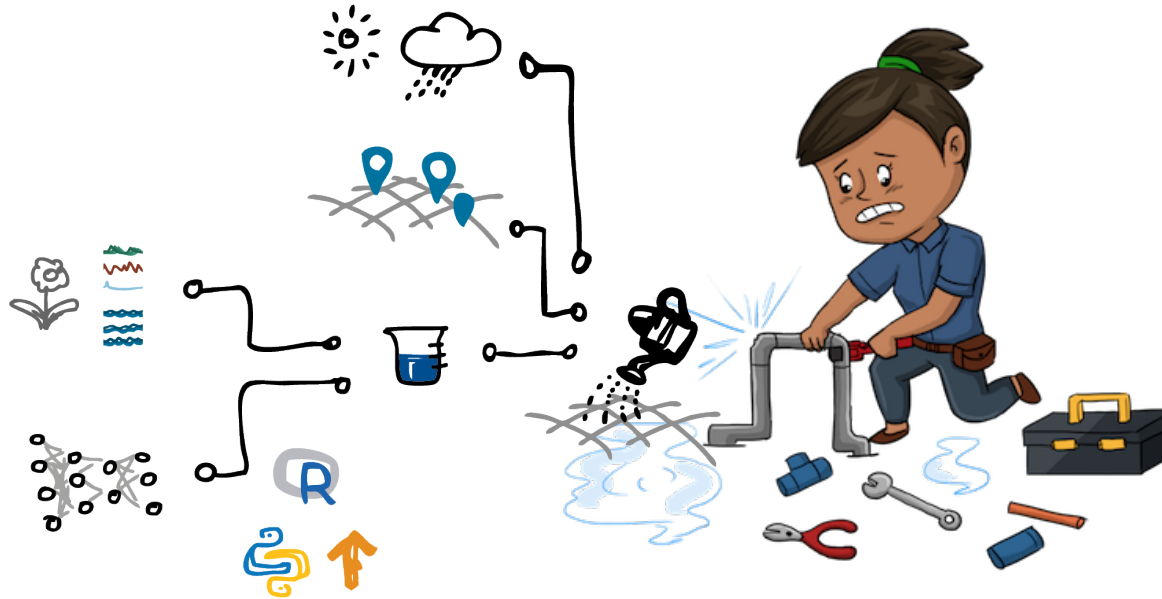
preserve_logs true;

Server {
  listen 12345;
  ...
}
```

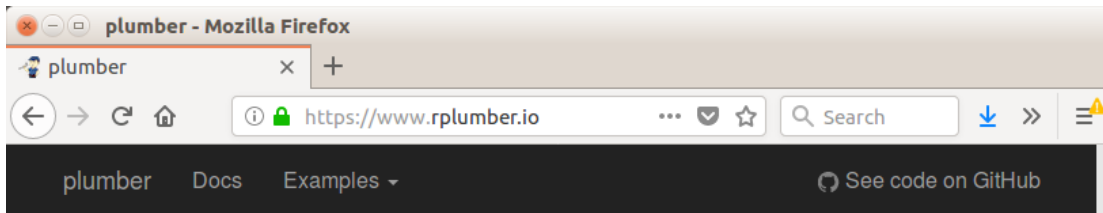


Dockerfile

```
...
COPY shiny-server.conf /etc/shiny-server/shiny-server.conf
...
```



<https://www.rplumber.io/>
<https://www.rplumber.io/docs/>



Introduction

plumber allows you to create a REST API by merely decorating your existing R source code with special comments. Take a look at an example.

```
# plumber.R

#* Echo back the input
#* @param msg The message to echo
#* @get /echo
function(msg=""){
  list(msg = paste0("The message is: '", msg, "'"))
}

#* Plot a histogram
#* @png
#* @get /plot
function(){
  rand <- rnorm(100)
  hist(rand)
}

#* Return the sum of two numbers
#* @param a The first number to add
#* @param b The second number to add
#* @post /sum
function(a, b){
  as.numeric(a) + as.numeric(b)
}
```



```
> library(plumber)
> r <- plumb("plumber.R") # Where 'plumber.R' is the lo
  cation of the file shown above
> r$run(port=8000)
```

You can visit this URL using a browser or a terminal to run your R function and get the results. For instance <http://localhost:8000/plot> will show you a histogram, and <http://localhost:8000/echo?msg=hello> will echo back the 'hello' message you provided.

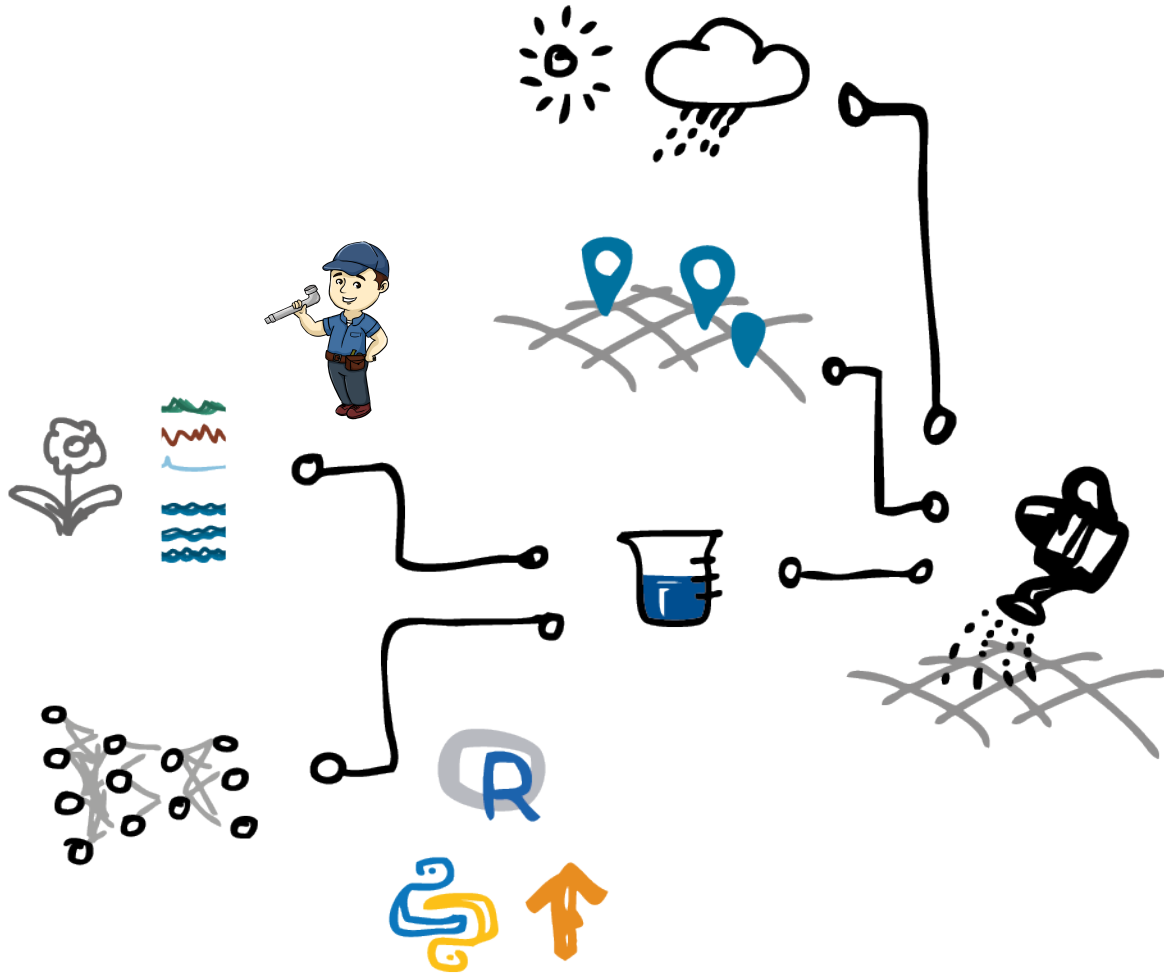
Here we're using `curl` via a Mac/Linux terminal.

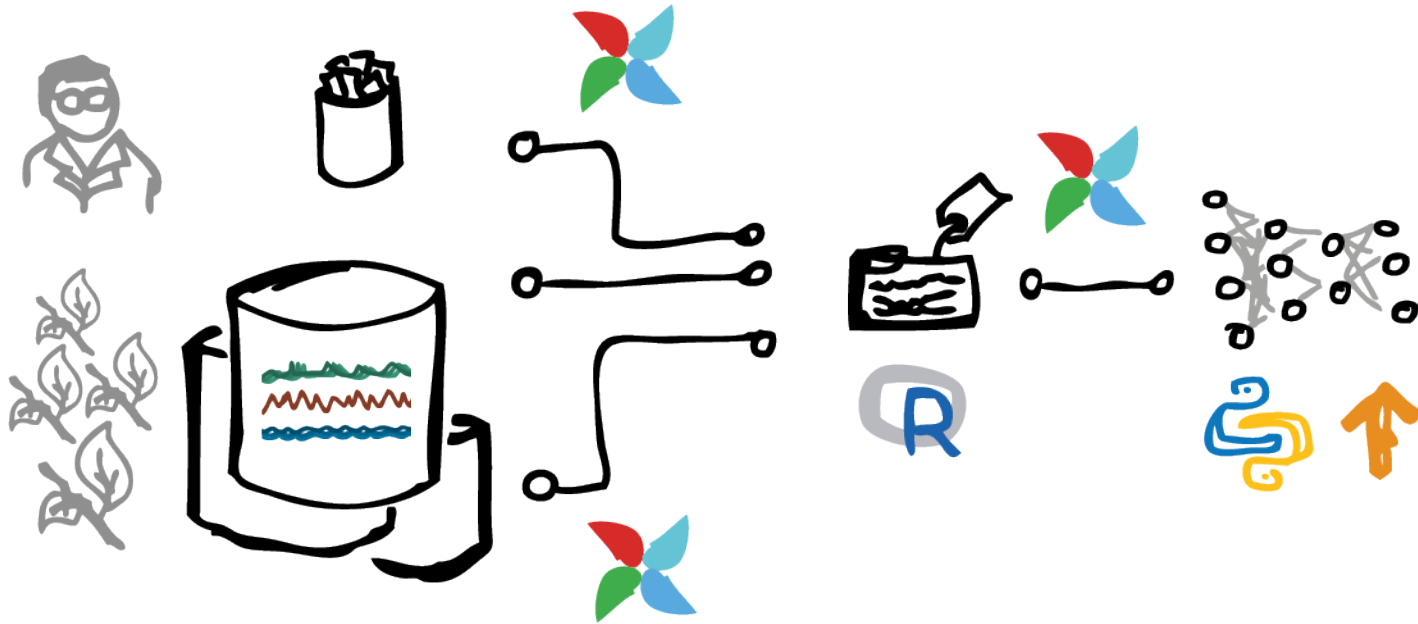
```
$ curl "http://localhost:8000/echo"
{"msg":["The message is: '']}
$ curl "http://localhost:8000/echo?msg=hello"
{"msg":["The message is: 'hello'"]}
```

As you might have guessed, the request's query string parameters are forwarded to the R function as arguments (as character strings).

```
$ curl --data "a=4&b=3" "http://localhost:8000/sum"
[7]
```

If you're still interested, check out our [more thorough documentation](#).





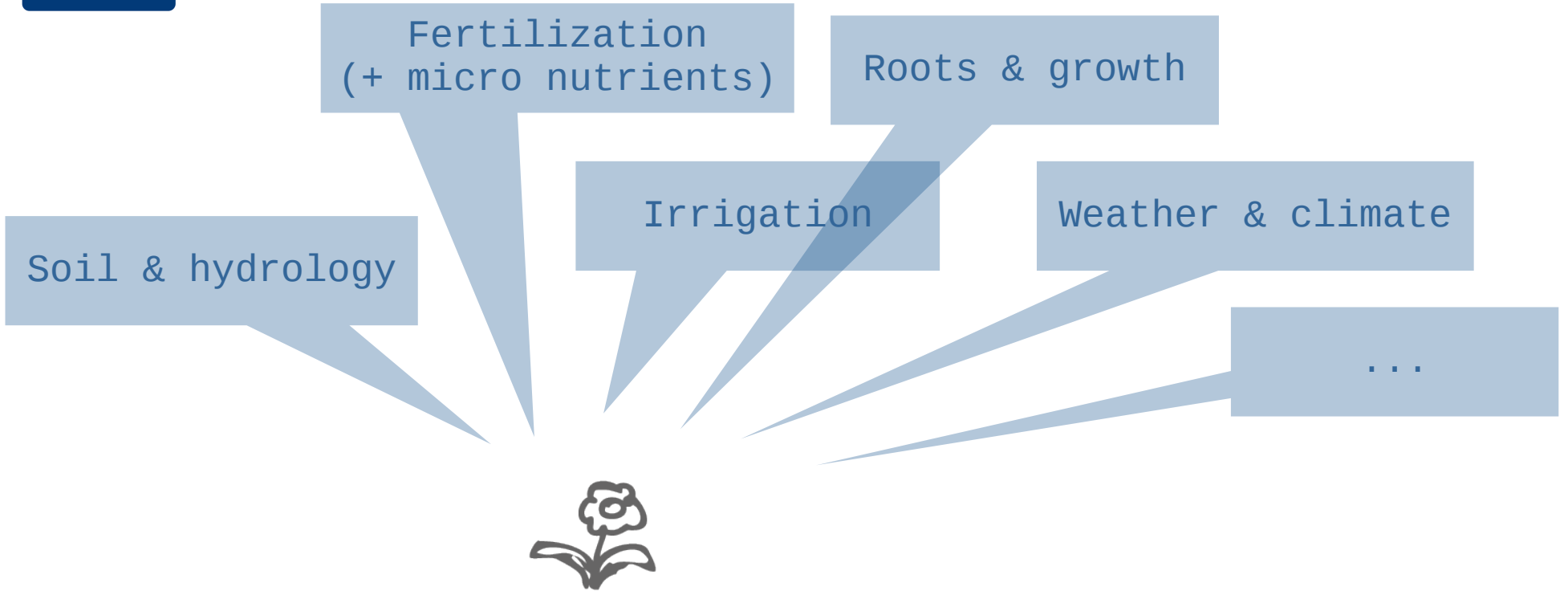
<https://airflow.apache.org/>
<https://airflow.apache.org/ui.html>



... from seed to harvest



Yara Digital is starting a new project initiative:
Complete characterization of the crop cycle





Knowledge grows

Come talk to us!

Claudia Stötzel
Jesus Martinez
Richard Brosi



