

# R로 만든 머신러닝 모델을 api로 제공하기

박찬엽

Data manager of finance / Korbit

# 목차

- 발표자 소개
- 데이터 과학자
  - 데이터 과학자란
  - 데이터 과학자가 하는 일
  - R 과 생태계 소개
- 커뮤니케이션
  - 휴먼 리더블한 R 도구
  - 머신 리더블한 api 서버
- plumber 소개
  - 함수 정의 방식의 api 작성
  - Serializers
  - 테스트 방법
  - 배포 방법
- 머신러닝 서비스를 위해
  - 고려해야 할 점
  - 지속적 학습을 위한 구조
- 마치며

## 박찬엽



- (현)코빗 재무팀 데이터 담당자
  - 재무DB 구축/관리 및 자동화
- (전)서울도시가스 선행연구팀 연구원
  - 챗봇 엔진 개발 및 서버 구축
- (전)2017년 패스트 캠퍼스 데이터 분석 R 강의
  - 데이터 분석을 위한 중급 R 프로그래밍
- N사 뉴스 크롤러 N2H4, D사 뉴스 크롤러 DNH4 관리자
  - ForkonLP 프로젝트
- FACEBOOK@mrchypark
- GITHUB@mrchypark

# 오늘 하고 싶은 이야기는 plumber 패키지 소개

# 데이터 과학자

# 데이터 과학자란

# MODERN DATA SCIENTIST

Data Scientist, the sexiest job of 21th century requires a mixture of multidisciplinary skills ranging from an intersection of mathematics, statistics, computer science, communication and business. Finding a data scientist is hard. Finding people who understand who a data scientist is, is equally hard. So here is a little cheat sheet on who the modern data scientist really is.

## MATH & STATISTICS

- ★ Machine learning
- ★ Statistical modeling
- ★ Experiment design
- ★ Bayesian inference
- ★ Supervised learning: decision trees, random forests, logistic regression
- ★ Unsupervised learning: clustering, dimensionality reduction
- ★ Optimization: gradient descent and variants

## DOMAIN KNOWLEDGE & SOFT SKILLS

- ★ Passionate about the business
- ★ Curious about data
- ★ Influence without authority
- ★ Hacker mindset
- ★ Problem solver
- ★ Strategic, proactive, creative, innovative and collaborative

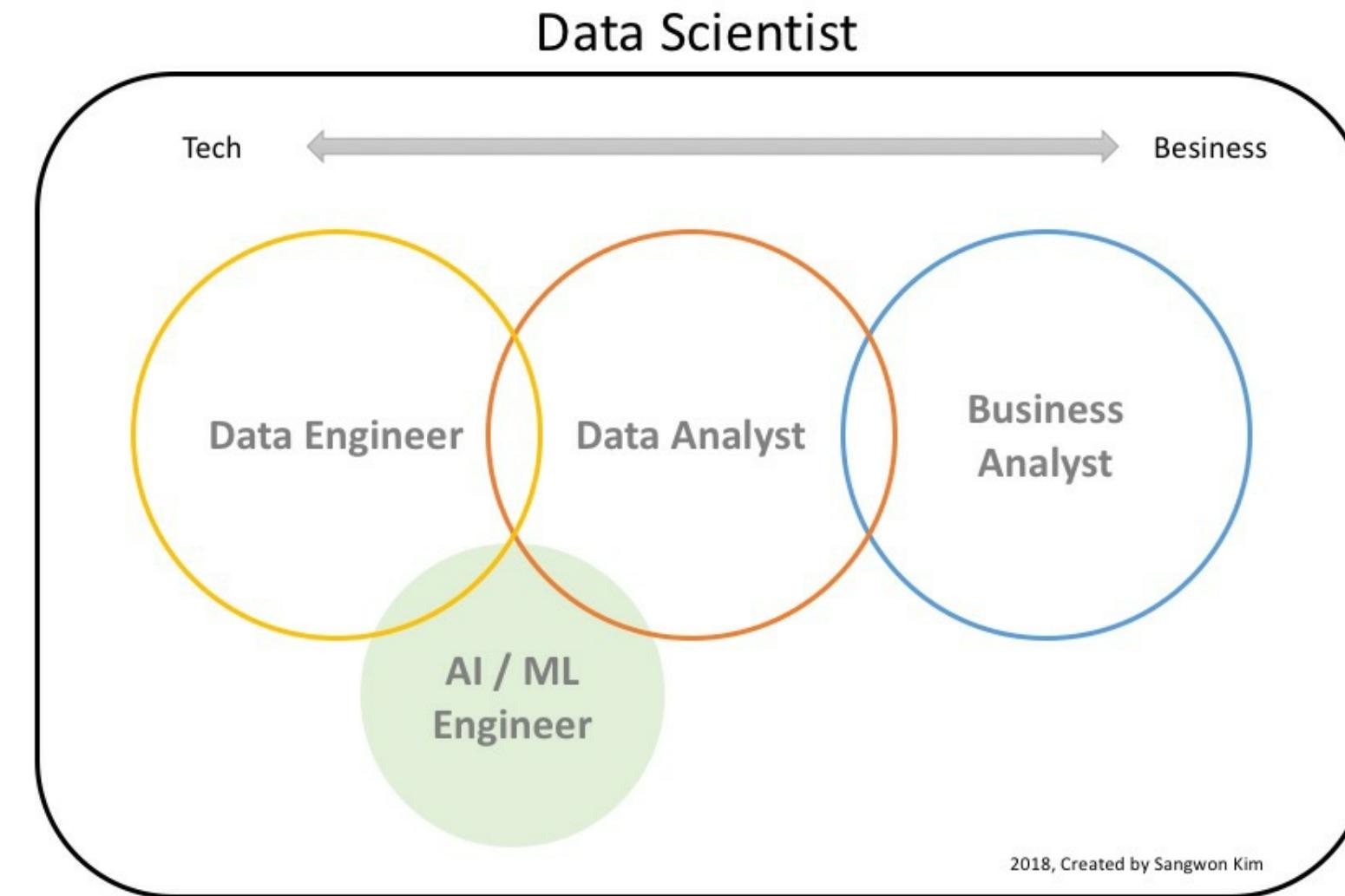


## PROGRAMMING & DATABASE

- ★ Computer science fundamentals
- ★ Scripting language e.g. Python
- ★ Statistical computing package e.g. R
- ★ Databases SQL and NoSQL
- ★ Relational algebra
- ★ Parallel databases and parallel query processing
- ★ MapReduce concepts
- ★ Hadoop and Hive/Pig
- ★ Custom reducers
- ★ Experience with xaaS like AWS

## COMMUNICATION & VISUALIZATION

- ★ Able to engage with senior management
- ★ Story telling skills
- ★ Translate data-driven insights into decisions and actions
- ★ Visual art design
- ★ R packages like ggplot or lattice
- ★ Knowledge of any of visualization tools e.g. Flare, D3.js, Tableau



출처 : <https://brunch.co.kr/@hero4earth/3>

두산백과

# 과학자

[scientist , 科學者]

**요약** 이론적 또는 실험적 연구를 통하여 과학지식을 탐구하는 사람.

연구를 통하여 과학 지식  
을 탐구하기 위해

통계 지식이 필요합니다.

## 과학자

데이터 확보가 어려움

- 실험
- 설문

## 데이터 과학자

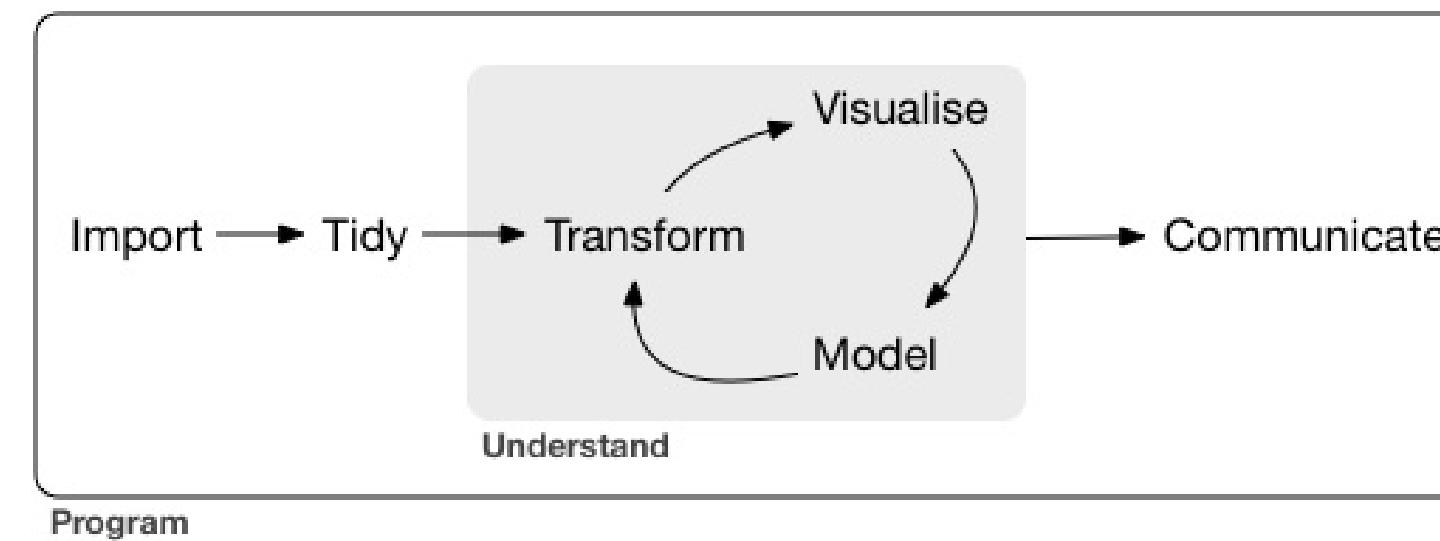
디지털화된 데이터를 확보

- A/B 테스트
- 사용자 로그

이제, 연구를 통하여 과  
학 지식을 탐구하기 위해

개발 지식이 필요합니다.

# 데이터 과학자가 하는 일



출처 : <https://r4ds.had.co.nz/introduction.html#what-you-will-learn>

# R 과 생태계 소개

R ❤ Human

# 강점

## 1. 인덱트가 1로 시작함

```
num <- 1:10  
num[3]
```

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

# 강점

## 2. 패키지 시스템이 단순함

- 콘솔에서 설치
- 글로벌 설치가 기본값

```
install.packages("tidyverse")
```

# 강점

## 3. 문서등을 사전 검수하는 공식 패키지 저장소 cran

- 등록하는 모든 패키지를 사전 검수함
- 마이너 버전 호환을 보장하며, 판올림 마다 각 패키지 테스트가 깨지면 알림
- **?함수명**으로 동작하는 개별 함수 설명서 필수 / 동작 예시 코드 제공
- 패키지에 따라 비네트(vignettes)라는 이름의 튜토리얼, 워크플로우 설명을 제공

# R은 사용자 친화적이다.

R ❤ Local files

R ❤ Web data

R ❤ Database

# Local files



```
library(rio)
dat <- import("path/what/you/read")
```

csv, xlsx, sav 등 대부분의 입출력을 바로 지원

writexl 패키지는 안정적인 엑셀 형식 파일 저장 기능을 제공

# Web data

`httr` - GET, POST, PUT 등 http 명세에 맞는 요청을 지원

`rvest` - xml을 %>% 이용해 쉽게 처리할 수 있는 함수를 제공

`RSelenium` - Selenium을 R에서 사용할 수 있게 작성

## driver list (DBI supported)

- SQLite
- MySQL(+MariaDB)
- bigquery
- Oracle
- ...

## + sergeant (with Apache Drill)

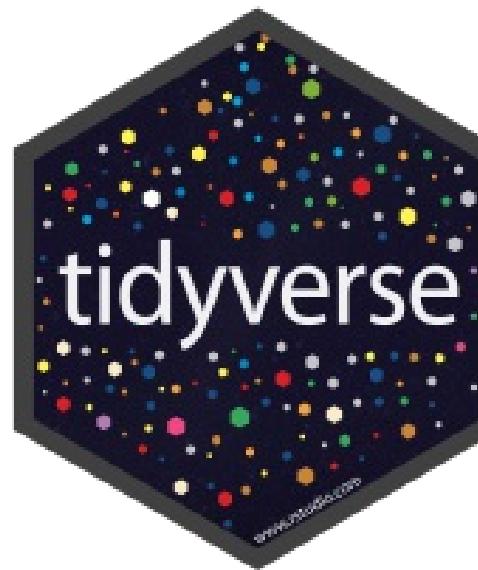
- HBase
- MapR-DB
- HDFS
- MapR-FS
- Amazon S3
- ...

## + noDBI supported

- MongoDB
- Redis
- CouchDB
- Elasticsearch
- etcd
- ...

## + JDBC & ODBC

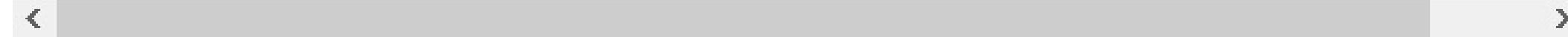
R ❤️ table



# 파이프 연산자(%>%)

함수를 중첩해서 사용할 일이 점점 빈번해짐

```
plot(diff(log(sample(rnorm(10000,mean=10,sd=1),size=100,replace=FALSE),  
           col="red",type="l"))
```



## %>%를 사용하면

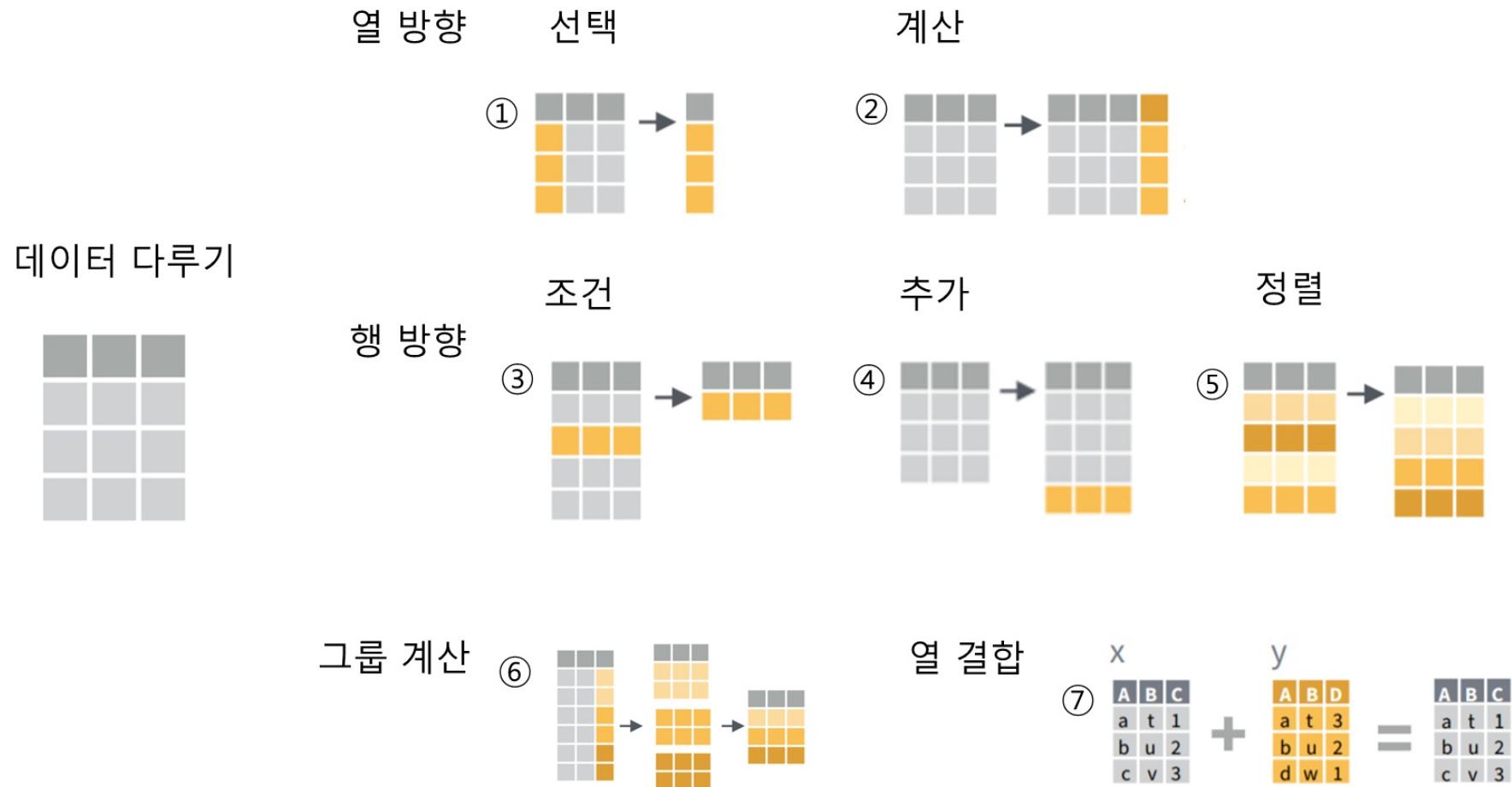
- 생각의 순서대로 함수를 작성할 수 있음
- 중간 변수 저장을 할 필요가 없음
- 순서가 읽이 용이하여 기억하기 좋음

```
rnorm(10000,mean=10,sd=1) %>%  
  sample(size=100,replace=FALSE)  
  log %>%  
  diff %>%  
  plot(col="red",type="l")
```





# 데이터를 다루는 7가지 동작



# 코드 품질

읽기 쉬운 코드가 좋은 코드

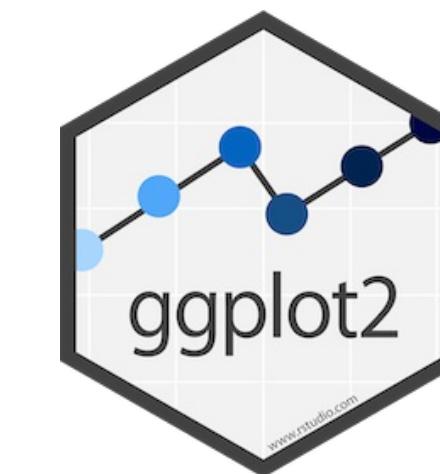
```
popular_dests %>%
  filter(arr_delay > 0) %>%
  mutate(prop_delay = arr_delay / sum(arr_delay)) %>%
  select(year:day, dest, arr_delay, prop_delay)
```

# 커뮤니케이션

# 휴먼 리더블한 R 도구

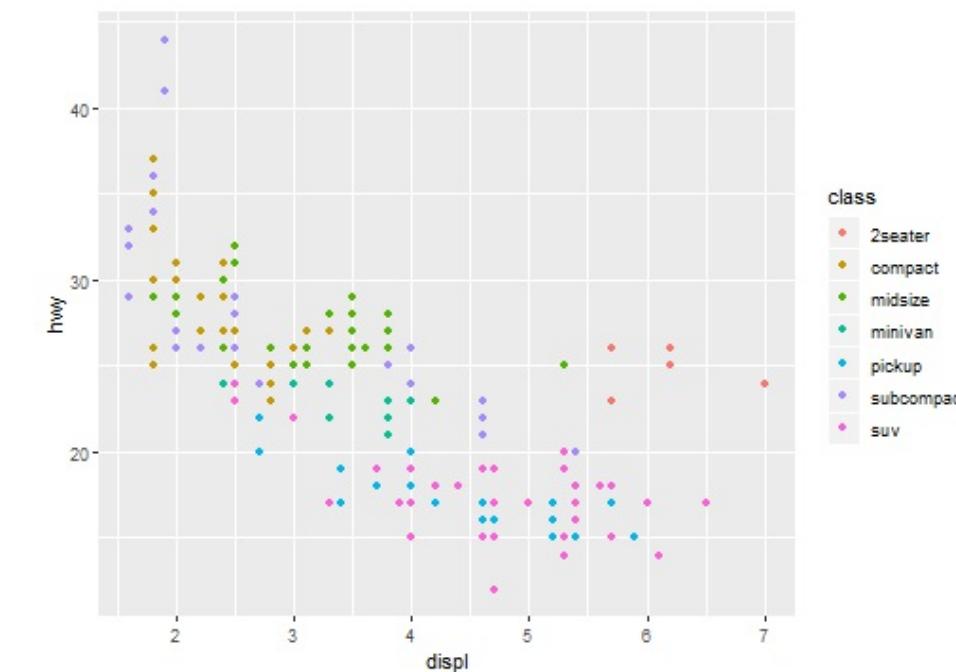
# 차트

색상 팔레트, 테마, autoplot를 지원하는 방대한 생태계 / publish quality



```
library(ggplot2)

ggplot(mpg,
       aes(displ,
           hwy,
           colour = class))
       ) +
geom_point()
```



# MS office

Presentation as Code

R code로 Word와 Power Point 결과물을 작성



# rmd 기반 문서

Rmd를 기반으로 출판 저작물 작성

publish as Code

논문([rticles](#))

책([bookdown](#))

블로그([blogdown](#), [radix](#))

등



## as Code

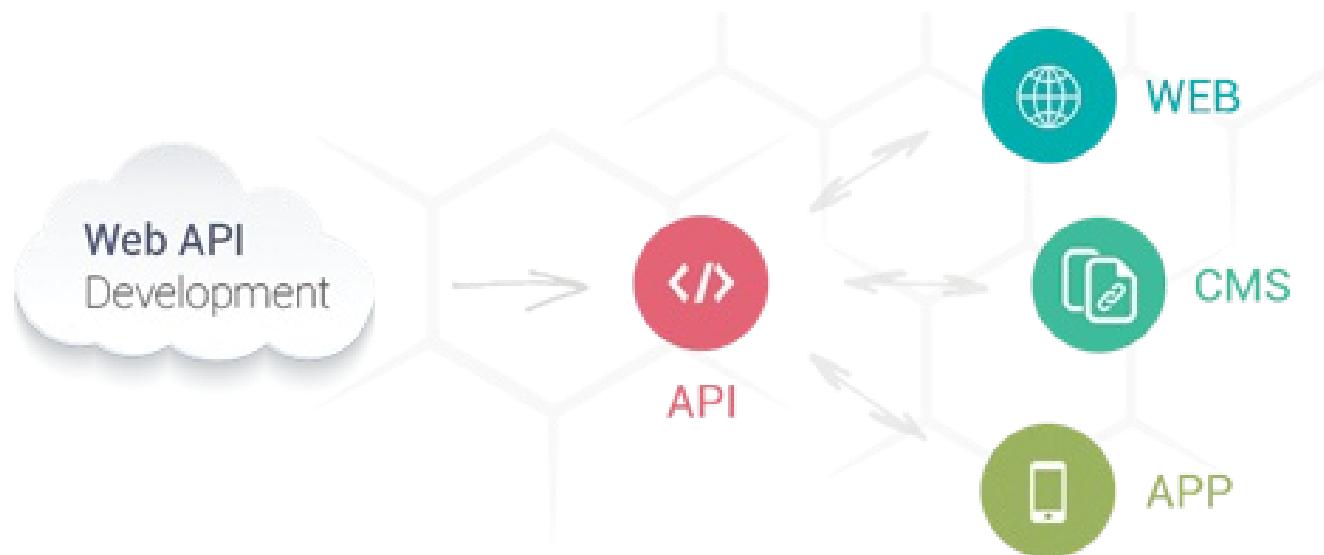
Publish Quality Chart 패키지를 활용하여 출판과 보고서를 code로 관리하면 분석 결과와 발표 자료를 통일 시킬 수 있음.

code를 수정해야 하는 스스로, .docx / .pptx / .xlsx / .html 등의 rendering 결과물은 손을 대지 않는 습관을 들이는 것이 중요함.

# 머신 리더블한 api 서버

# web api

- json, xml 등 code에서 처리하기 좋은 포맷의 데이터로 통신
- gRPC, http 명세 등 통신 프로토콜 정의

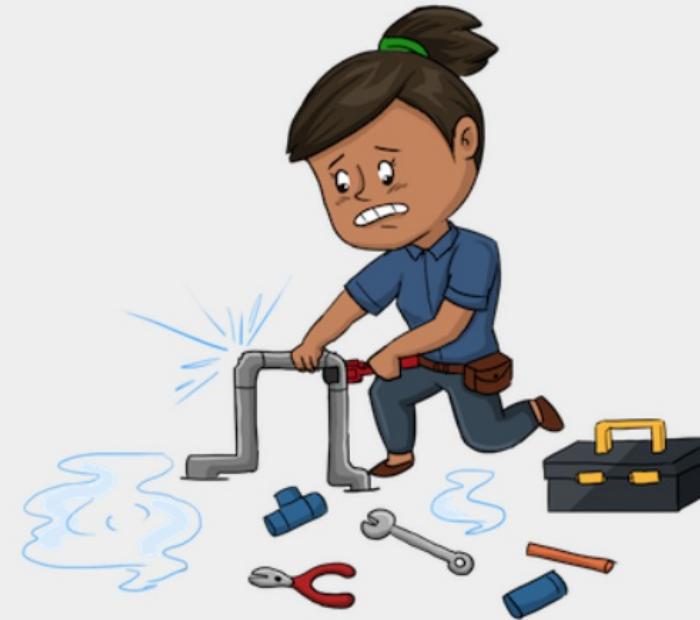


# R의 결과물로 api 서버 를 만들 수는 없을까

# plumber 소개

# plumber

An R package that converts your existing R code to a web API using a handful of special one-line comments.



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

# 함수 정의 방식의 api 작성

## 함수 정의와 같은 문법

패키지 작성시 사용하는 roxygen 패키지 문법을 사용하여 코드내 문서화

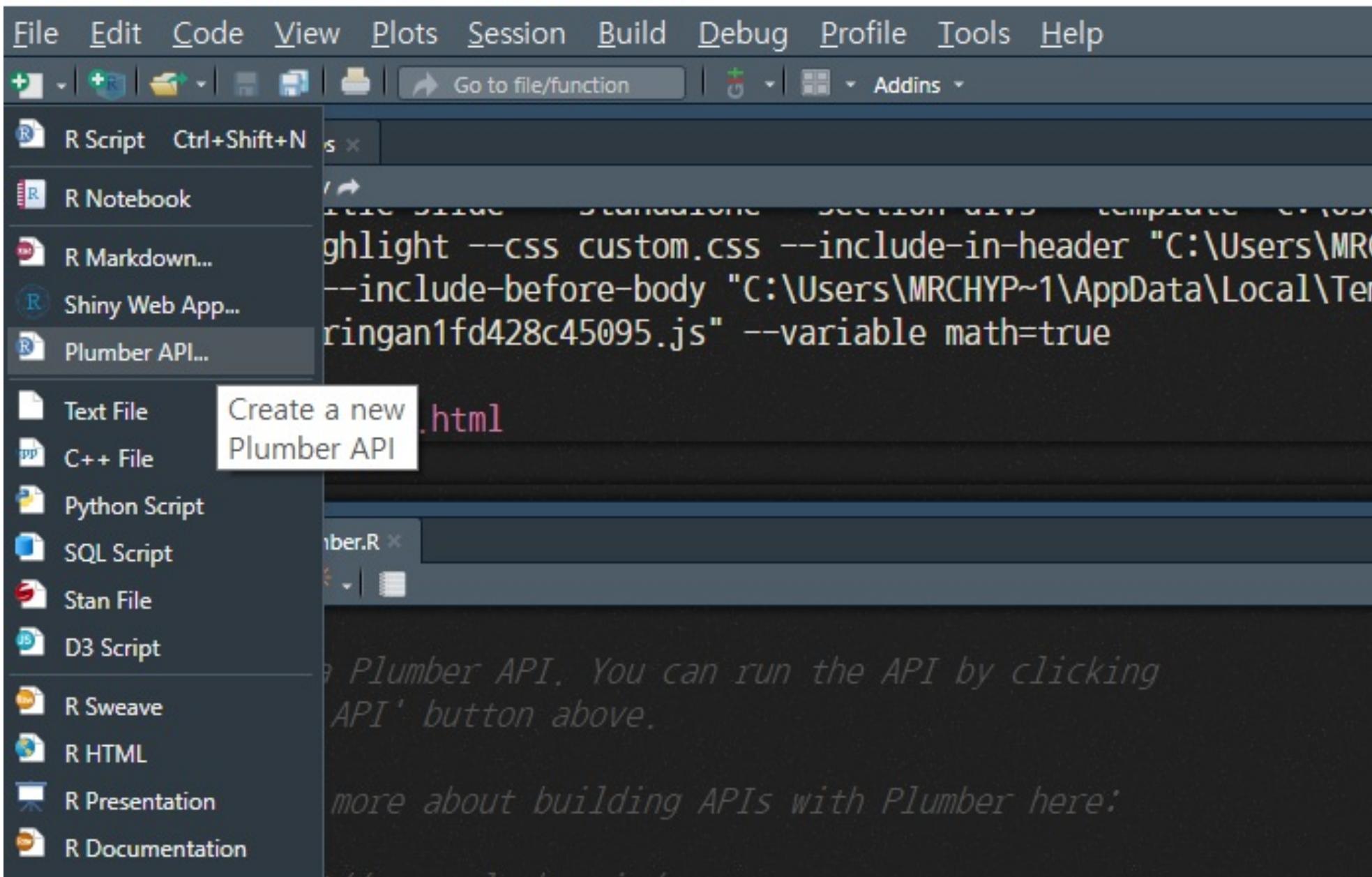
```
#' Return the sum of two numbers          # endpoint 설명
#' @param a The first number to add      # params 이름과 설명
#' @param b The second number to add     # endpoint 정의
#' @post /sum                            # 함수 작성
sum_api <- function(a, b) {
  as.numeric(a) + as.numeric(b)
}
```

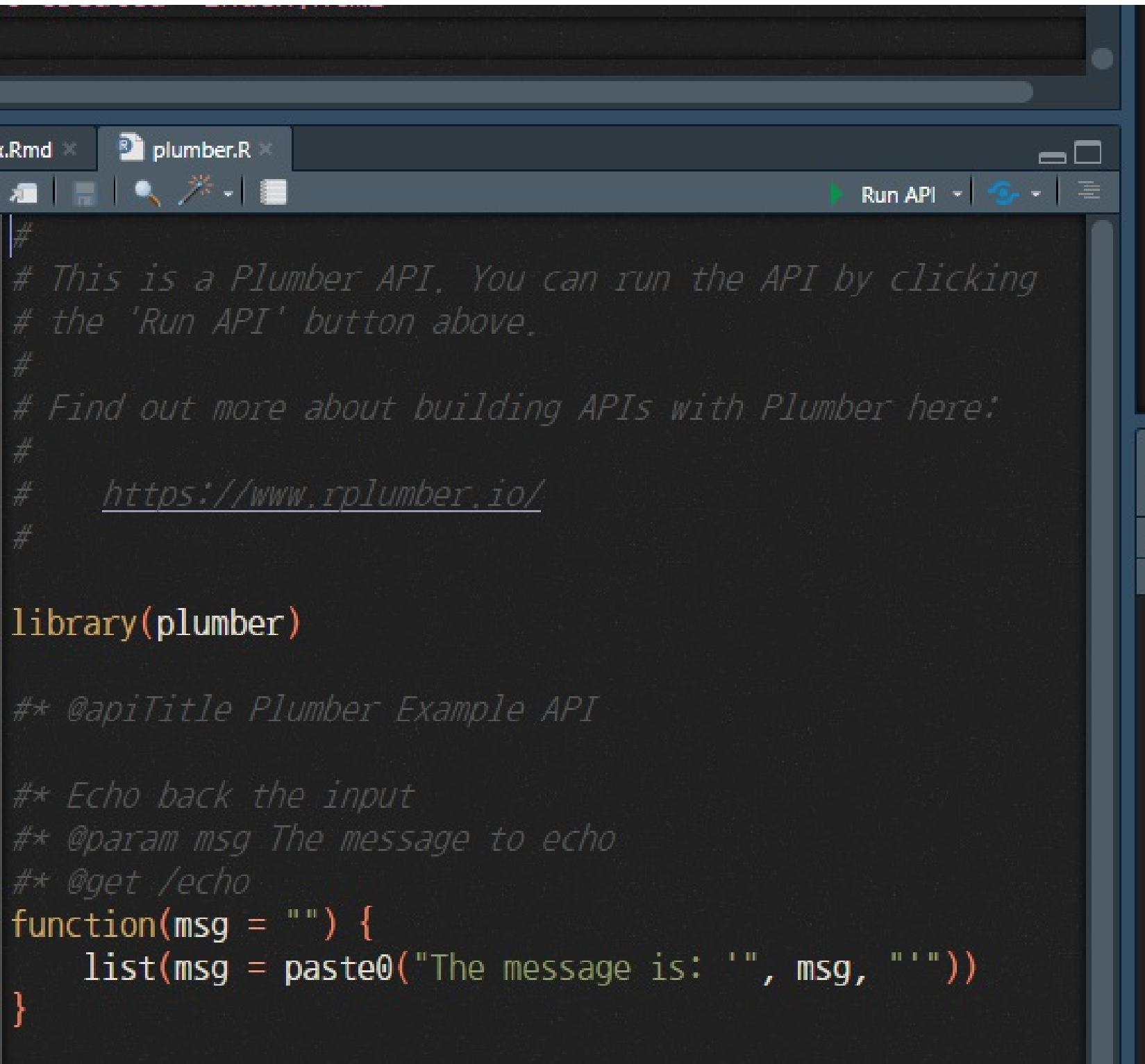
## 실행방법

```
library(plumber)
plumb("plumber.R")$run()
```

## 실행방법2

R ~/project/r-api-with-plumber - master - RStudio





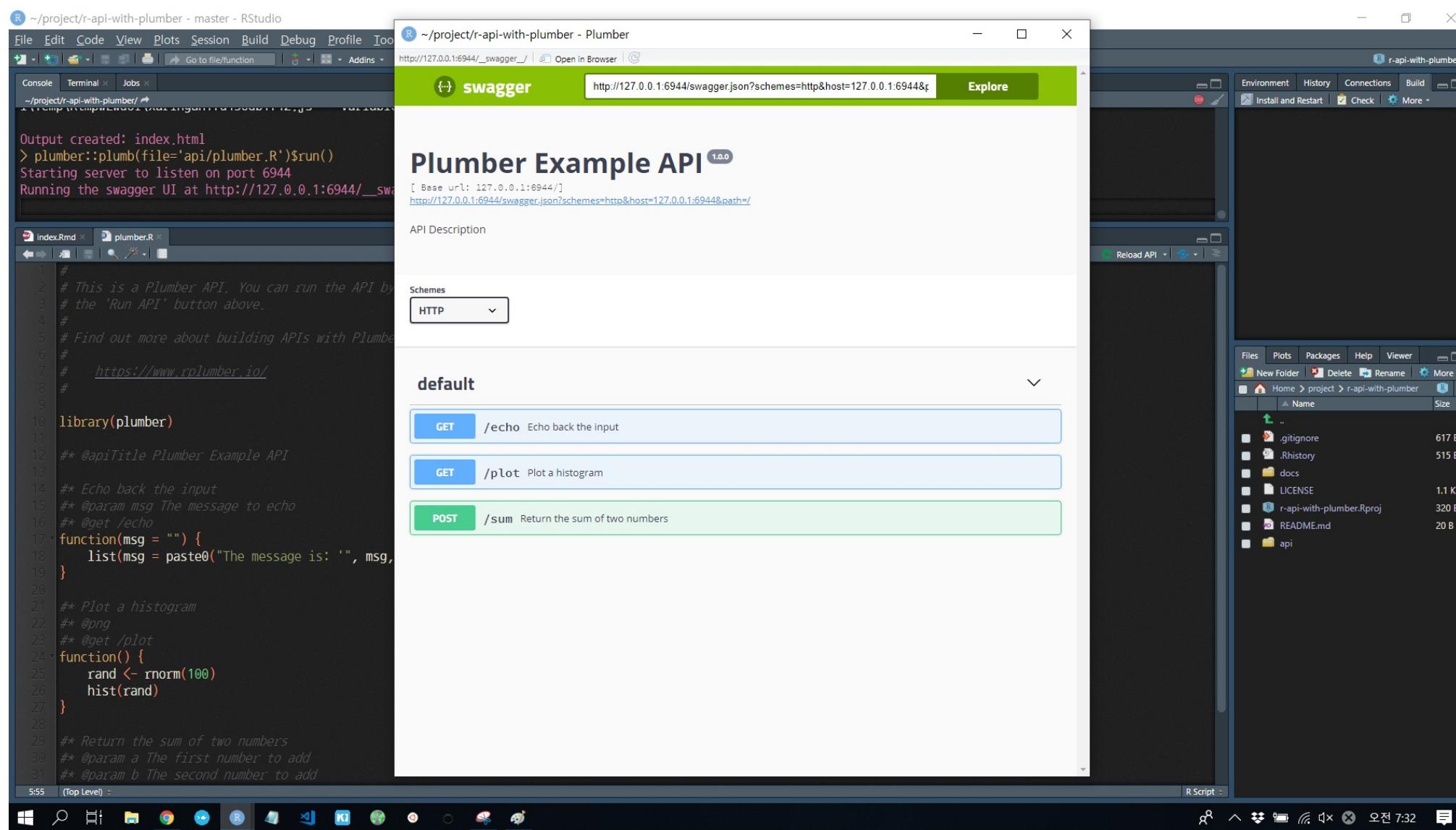
The screenshot shows an RStudio interface with two tabs open: 'x.Rmd' and 'plumber.R'. The 'plumber.R' tab contains the following R code:

```
# This is a Plumber API. You can run the API by clicking
# the 'Run API' button above.
#
# Find out more about building APIs with Plumber here:
#
# https://www.rplumber.io/
#
library(plumber)

#> @apiTitle Plumber Example API

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

# swagger 지원



## swagger 내 문서화 및 테스트

R ~/project/r-api-with-plumber - Plumber

http://127.0.0.1:6944/\_swagger\_ | Open in Browser | C

**POST** /sum Return the sum of two numbers

**Parameters**

Name	Description
b	The second number to add string (query) 1
a	The first number to add string (query) 5

**Execute** **Cancel**

**Responses**

Response content type application/json ▾

Curl

Page 48

```
curl -X POST "http://127.0.0.1:6944/sum?b=1&a=5" -H "accept: application/json"
```

## 다이나믹 라우트

로 작성하여 타입을 지정할 수 있음. 지정하지 않으면 모두 글자이며, bool, numeric, int 등 지원

```
#' Lookup a user
#' @get /users/<id>
function(id) {
  subset(users, uid==id)
}
```

# Serializers

# 다양한 방식을 지원

서버에서 지원하기를 기대하는 동작들을 거의 지원함.

1. 텍스트
2. 이미지(jpg, png)
3. htmlwidget

# 이미지

브라우저를 통해 이미지 결과물을 전송

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

# htmlwidget

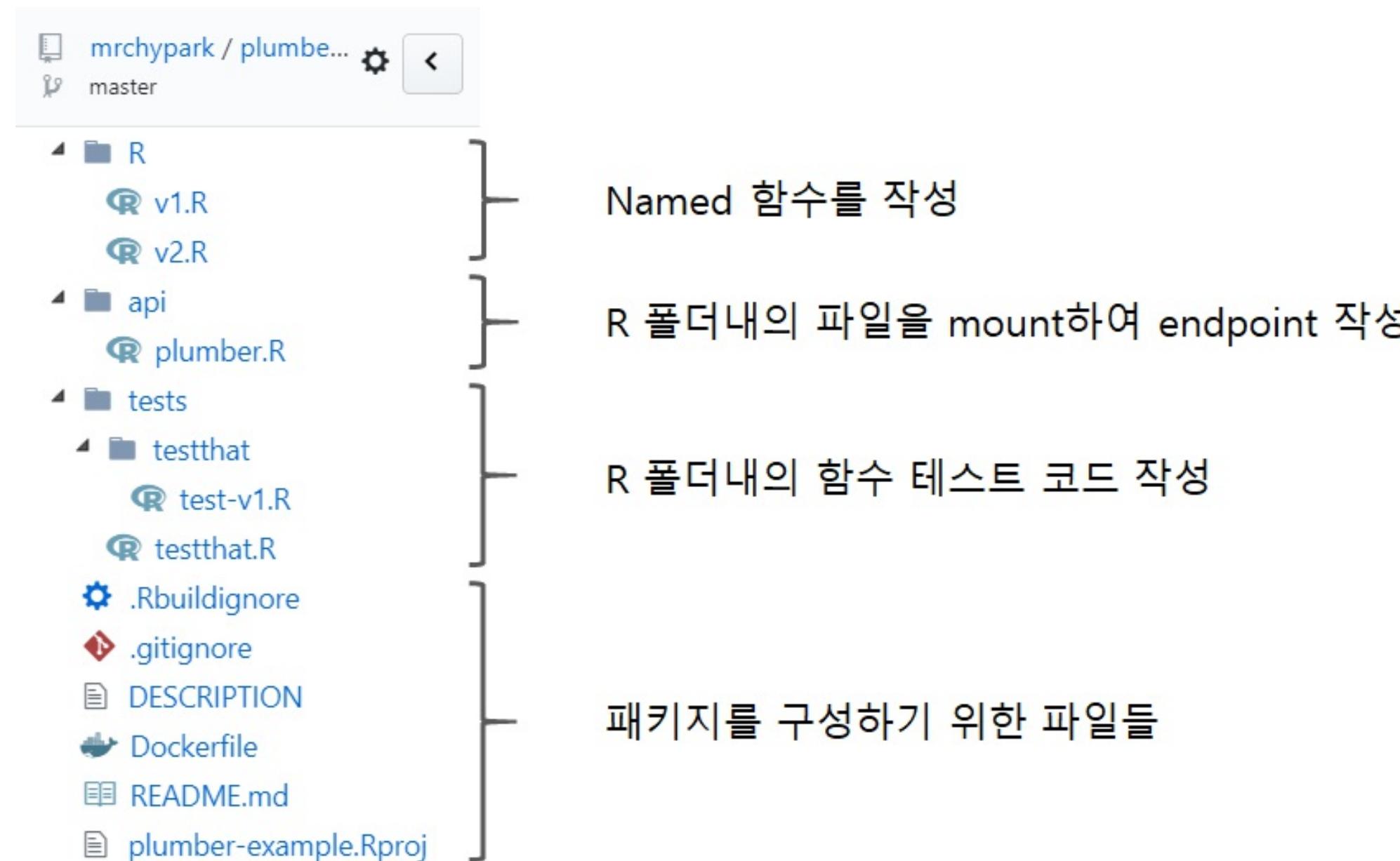
R에 htmlwidget을 이용해 동적 문서 작성 패키지의 결과물을 전송할 수 있음.

```
library(dygraphs)
#' @get /spots/<from>/<to>/graph
#' @serializer htmlwidget
function(from, to) {
  dygraph(datasets::sunspots, main = "썬스팟 데이터 셋") %>%
    dyRangeSelector(dateWindow = c(
      paste0(from, "-01-01"),
      paste0(to, "-12-31")
    )))
}
```

# 테스트 방법

테스트를 추가해 봅시다.

# 테스트를 위한 프로젝트 구조 예시



# 함수 이름을 작성

Branch: master ▾

[plumber-example / R / v1.R](#)

mrchypark Update v1.R

1 contributor

27 lines (22 sloc) | 470 Bytes

```
1 library(plumber)
2
3 #* @apiTitle Plumber Example API
4
5 #* Echo back the input
6 #* @param msg The message to echo
7 #* @get /echo
8 echo <- function(msg = "") {
9   list(msg = paste0("The message is: '", msg, "'"))
10 }
11
```

# 테스트 환경 설정

Branch: master ▾

plumber-example / tests / testthat.R



mrchypark init folder structure

1 contributor

5 lines (3 sloc) | 62 Bytes

```
1 library(testthat)
2 library(arbitrary)
3
4 test_check("arbitrary")
```

# 테스트 케이스 작성

Branch: master ▾

[plumber-example / tests / testthat / test-v1.R](#)

mrchypark init folder structure

1 contributor

7 lines (5 sloc) | 117 Bytes

```
1 context("test-v1")
2
3 test_that("multiplication works", {
4   tar <- echo("test")
5   expect_equal(class(tar), "list")
6 })
```

# 함수로 작성된 파일들을 실행할 스크립트

Branch: master ▾

[plumber-example / api / plumber.R](#)

mrchypark init folder structure

1 contributor

11 lines (7 sloc) | 173 Bytes

```
1 library(plumber)
2
3 v1 <- plumber$new("R/v1.R")
4 v2 <- plumber$new("R/v2.R")
5
6 master_pr <- plumber$new()
7 master_pr$mount("/v1", v1)
8 master_pr$mount("/v2", v2)
9
10 master_pr$run()
```

# DESCRIPTION

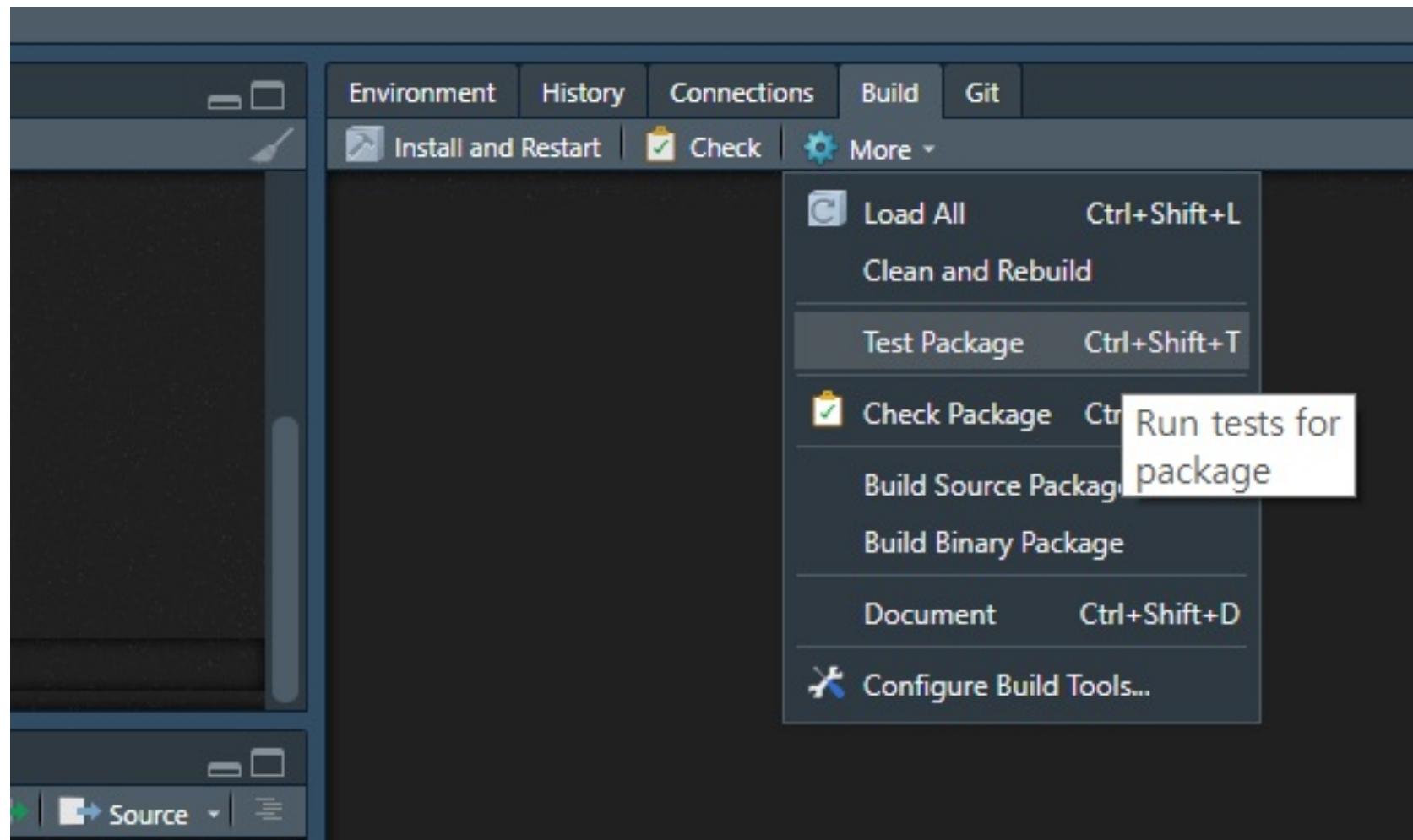
DESCRIPTION 파일은 프로젝트가 패키지임을 인식하기 위해 꼭 필요하기 때문에 error 가 발생하지 않을 최소한의 가짜 정보를 입력

Package: arbitrary

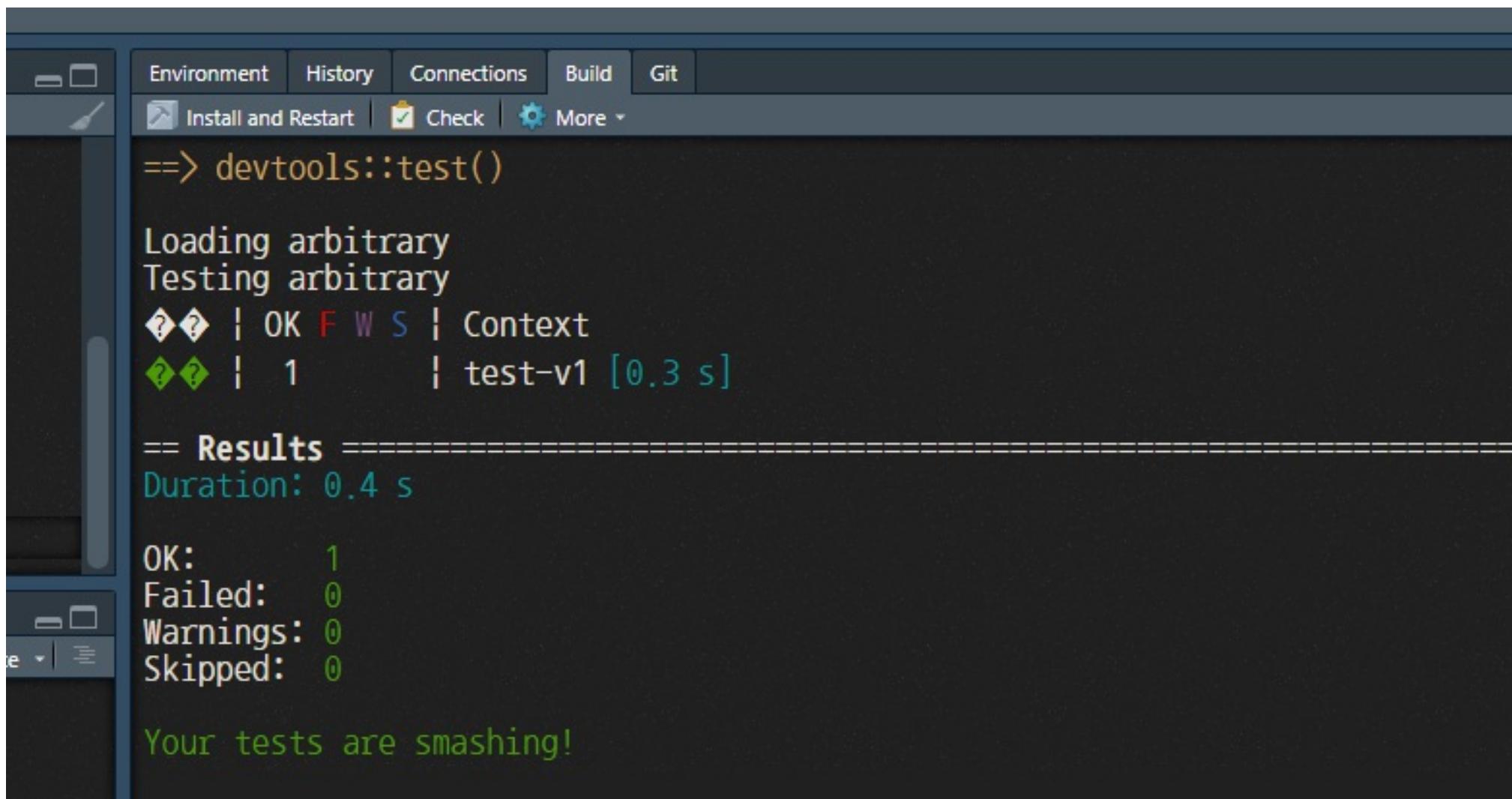
Title: Does not matter.

Version: 0.0.1

# Build 패널



## 테스트 결과



A screenshot of a terminal window displaying test results. The window has tabs at the top: Environment, History, Connections, Build, and Git. Below the tabs are buttons for Install and Restart, Check, and More. The main area of the terminal shows the following output:

```
--> devtools::test()

Loading arbitrary
Testing arbitrary
? ? | OK F W S | Context
? ? | 1          | test-v1 [0.3 s]

== Results ==
Duration: 0.4 s

OK:      1
Failed:   0
Warnings: 0
Skipped:  0

Your tests are smashing!
```

# CI에 올리기

R은 travis-ci, gitlab appveyor 등의 CI를 지원함.

배포를 다양한 환경에 할 것이 아니라면 각 상황에 맞는 툴을 사용할 것.

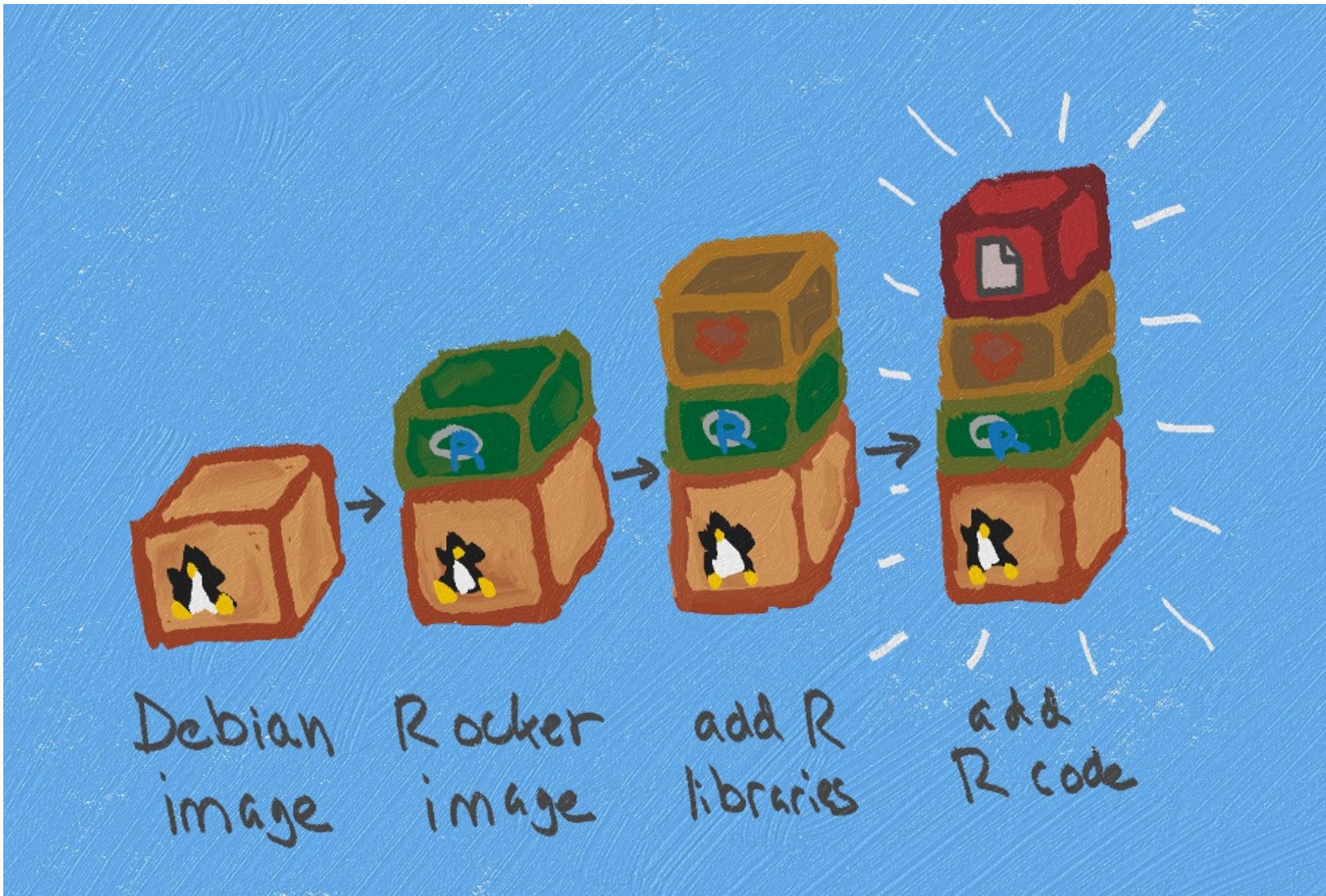
24 lines (19 sloc) | 322 Bytes

```
1 # R for travis: see documentation at https://docs.travis-ci.com/user/languages/r
2
3 language: R
4 sudo: false
5 cache: packages
6
7 addons:
8   apt:
9     packages:
10    - libcurl4-openssl-dev
11    - libxml2-dev
12    - libv8-3.14-dev
13
14 r:
15 - 3.3
16 - 3.4
17 - 3.5
```

## 배포 방법

역시 환경 격리에는 docker 입니다.

## 환경 격리



# containerit

containerit은 [sysreqs](#) 프로젝트에 기반하고 있음.

rocker는 r-base의 안정 버전 docker 이미지를 제공하는 [프로젝트](#).

debian 기반의 이미지로 환경이 통일되어 있으므로 각 패키지의 시스템 수준 의존성 정보 제공 프로젝트인 [sysreqs](#)가 가능해짐.

containerit은 연구재현성을 위해 개발 중



# rocker

## Base Docker Containers

image	description	size	metrics	build status
r-base	Current R via apt-get with debian:testing & unstable repos	274.6MB 12 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 6k	docker build automated
drd	lighter r-devel, built not quite daily	841.6MB 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 54k	docker build automated

# sysreqs

## sysreqs(7) – Map SystemRequirements for R packages

### SYNOPSIS

```
GET /
GET /get/<mapping>
GET /list
GET /map/<string>
GET /populate
GET /pkg/<package>[ /<os>]
```

### DESCRIPTION

**sysreqs** is a database of system requirements mappings for installing and running R packages on various operating systems.

R packages may specify **SystemRequirements** in their **DESCRIPTION** files to list software that must or should be installed on the system to use the package. This is not a standardized field, and various packages specify the same system requirements differently.

**sysreqs** is a hand-curated database of the **SystemRequirements** entries, and their resolutions on various operating systems. The data itself is stored in a GitHub repository, at <https://github.com/r-hub/sysreqs>, and its public API is at <https://sysreqs.r-hub.org>.

All endpoint responses (except for the / help page) are JSON encoded.

# Dockerfile

```
1 ## Autu-gen using containerit package. It's not work for plumber right now.  
2  
3 FROM rocker/r-ver:3.5.1  
4 LABEL maintainer="mrchypark"  
5 RUN export DEBIAN_FRONTEND=noninteractive; apt-get -y update \  
6     && apt-get install -y git-core \  
7         libcurl4-openssl-dev \  
8         libssh2-1-dev \  
9         libssl-dev \  
10        make \  
11        zlib1g-dev  
12 RUN ["install2.r", "assertthat", "backports", "clisymbols", "crayon", "curl", "digest", "dygraphs",  
13 RUN ["installGithub.r", "r-lib/desc@7c12d369c202018dfc3dfa3f975f03c8b6c18437", "RcppCore/Rcpp@a669a  
14 WORKDIR /payload/  
15 CMD ["R"]
```

이걸로 끝이라고는...

상황에 맞게 수정해서 사용해야 합니다.

# 머신러닝 서비스를 위해

# 고려해야 할 점

## 팀의 크기

머신러닝 서비스를 위해 최소 3인(도메인 전문가, 머신러닝 엔지니어, 소프트웨어 엔지니어)이 필요함

언제나처럼 제한된 자원 내에서 혼자 진행해야 할 수도 있음

## 팀의 크기

머신러닝 서비스를 위해 최소 3인(도메인 전문가, 머신러닝 엔지니어, 소프트웨어 엔지니어)이 필요함

언제나처럼 제한된 자원 내에서 혼자 진행해야 할 수도 있음

상황에 따라 데이터 과학자에 해당하는 사람이 얼마나 알고 있어야 하는가?

혹은 자신에게 부족한 부분에서 어떻게 조직내 협업을 이끌어 낼 것인가?

# 얼마나 자주 변경해야 하는가

머신러닝의 성능은 학습 데이터에 의존적임

데이터의 유효기간이 지나거나, 새로운 고객군이 기존 고객군과 성격이 달라질 수 있음.

# 얼마나 자주 변경해야 하는가

머신러닝의 성능은 학습 데이터에 의존적임

데이터의 유효기간이 지나거나, 새로운 고객군이 기존 고객군과 성격이 달라질 수 있음.

얼마나 자주 학습해야 할까?

테스트는 어떻게 진행해야 할까?

## 사람의 개입

머신러닝 프로젝트는 완전 자동화를 목표로 하는 것은 위험함

특히 새로운 변수를 발굴하거나, 알고리즘을 바꾸는 등의 상황은 전부 사람이 판단해야 함

## 사람의 개입

머신러닝 프로젝트는 완전 자동화를 목표로 하는 것은 위험함

특히 새로운 변수를 발굴하거나, 알고리즘을 바꾸는 등의 상황은 전부 사람이 판단해야 함

어떤 작업을 사람이 판단해야 안전한가?

사람의 작업이 어떻게 설계되어야 scale up에 유리한가?

# 지속적 학습을 위한 구조

## 지속적 학습에 대해 고려해야 할 사항

데이터 확보: 보통지도 학습 모델을 많이 사용하기 때문에 확보 비용이 매우 높음

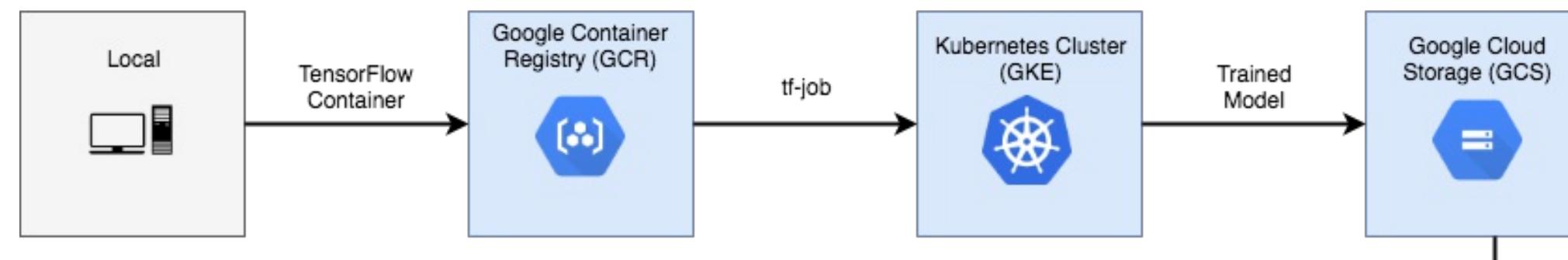
학습: 데이터 상황에 맞는 알고리즘 및 변수 선택 등

배포: 테스트 케이스나 정확도를 기준으로 배포 모델을 결정

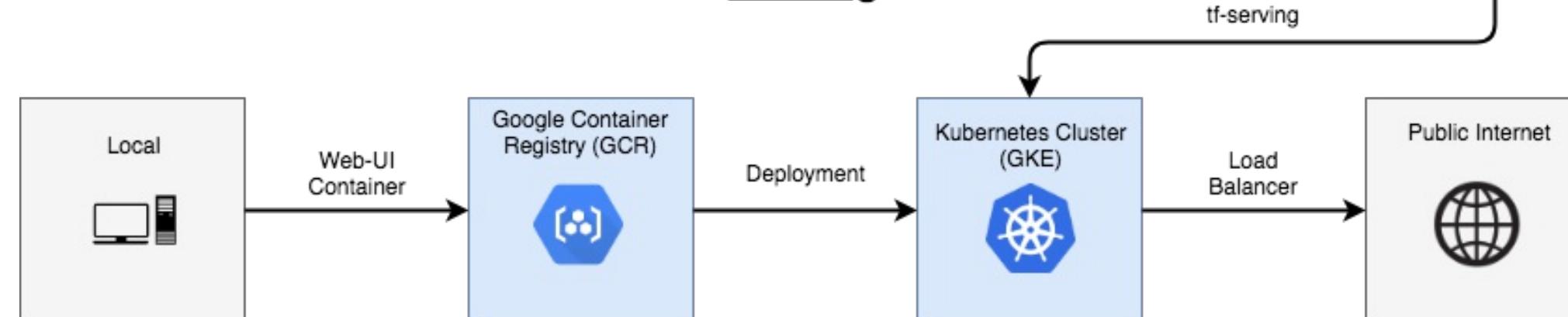
# 수동 학습과 수동 배포

서비스 초기에는 모두 수동 작업이 필요함

## Training

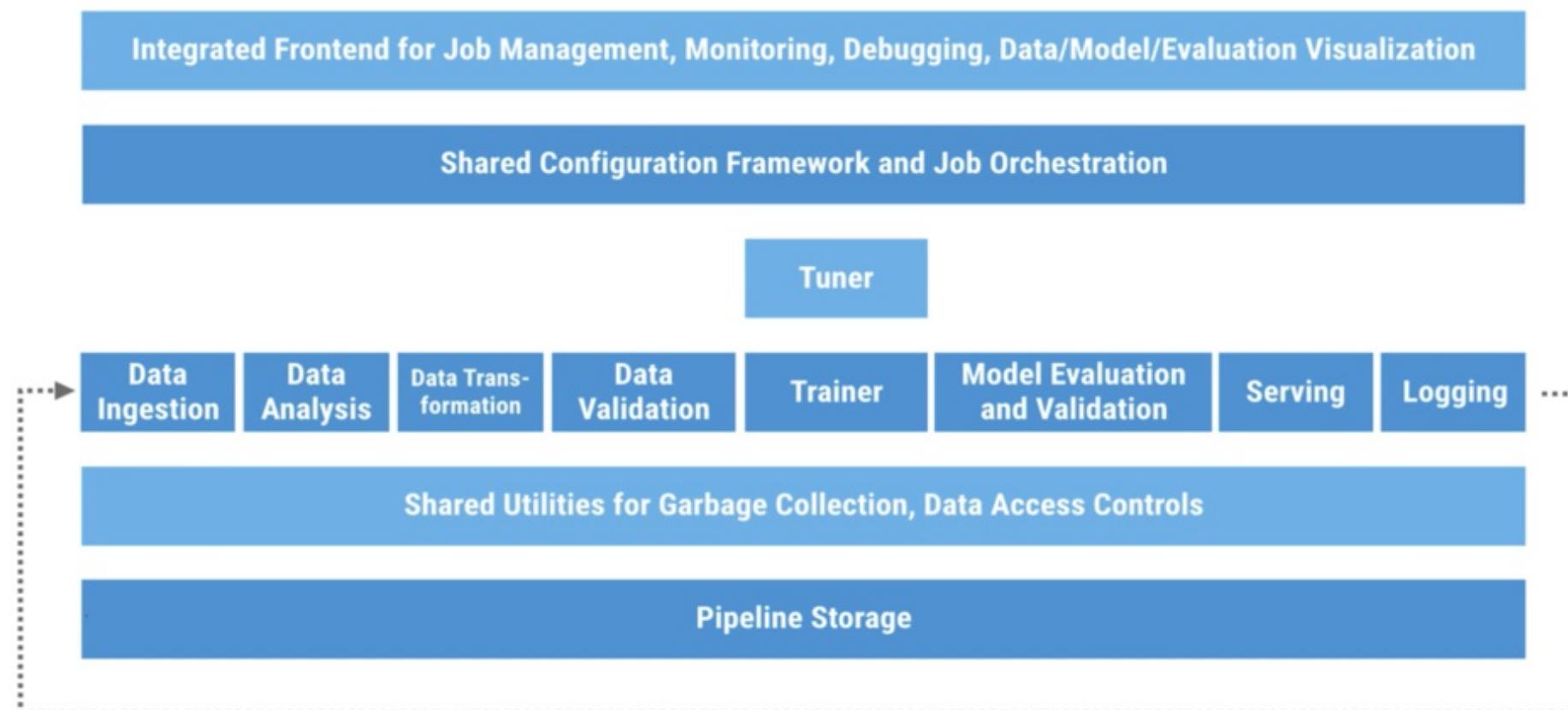


## Serving



# 지속적 데이터 확보

사용자 이벤트를 기반으로 지도학습용 데이터 파이프라인 구축. 가능한한 사람 손을 타지 않거나, 타는 것이 필수인 구조인 것이 좋음



# 마치며

# plumber는 어느 위치에 있는가

R의 위치와 함께 plumber는 빠르게 프로토 타이핑이 가능

빠르게 변하고 리소스가 부족한 경우 적합할 수 있음

회사내 튜터가 있다면 도메인 전문가가 도전해 볼만함

# plumber는 어느 위치에 있는가

R의 위치와 함께 plumber는 빠르게 프로토 타이핑이 가능

빠르게 변하고 리소스가 부족한 경우 적합할 수 있음

회사내 튜터가 있다면 도메인 전문가가 도전해 볼만함

물론 배포 등은 엔지니어의 도움을 받아야

# Q&A