프로덕션 수준의
리포트 자동화 시스템 만들기

김상현, 이민호 데이터라이즈

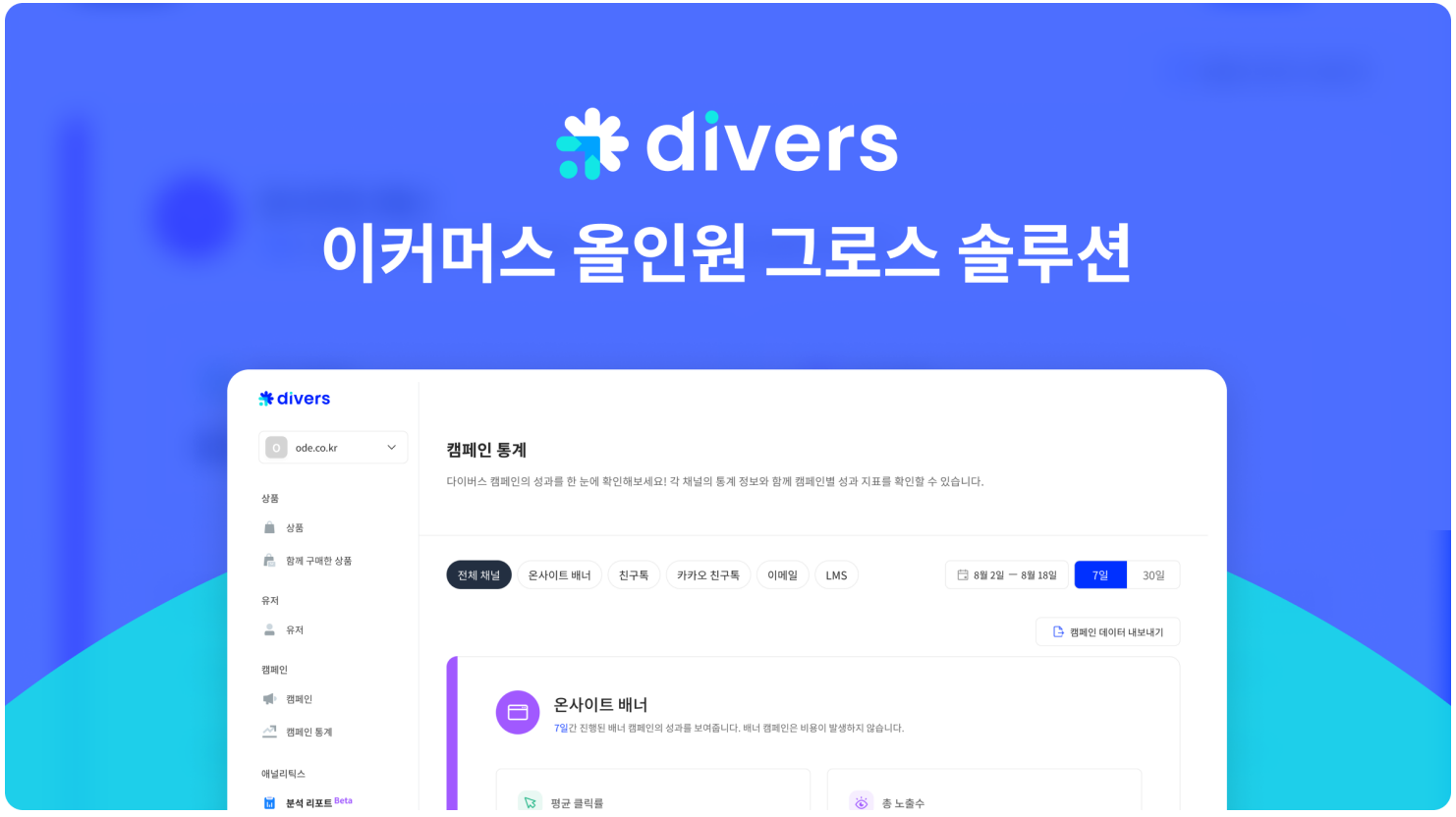
Korea R Conference 2021

PART1

리포트 자동화 시스템

우리가 하고 있는 일 : Divers!

- 이커머스를 위한 B2B SaaS 올인원 그로스 솔루션
- 데이터 수집부터, **분석과 진단**, 자동화 된 개인화 캠페인으로 연결되는 파이프라인 구축



왜 리포트 시스템을 구축했는가?

- 커머스 성장을 위한 데이터 애널리틱스 페이지를 개발하고자 함
- 스타트업이기에 해야할 것은 산더미고 항상 리소스가 부족함
- 열심히 기획하고 디자인하고 개발해서 페이지를 배포했는데 쓰지 않는다면?

왜 리포트 시스템을 구축했는가? : 🔮 실험실

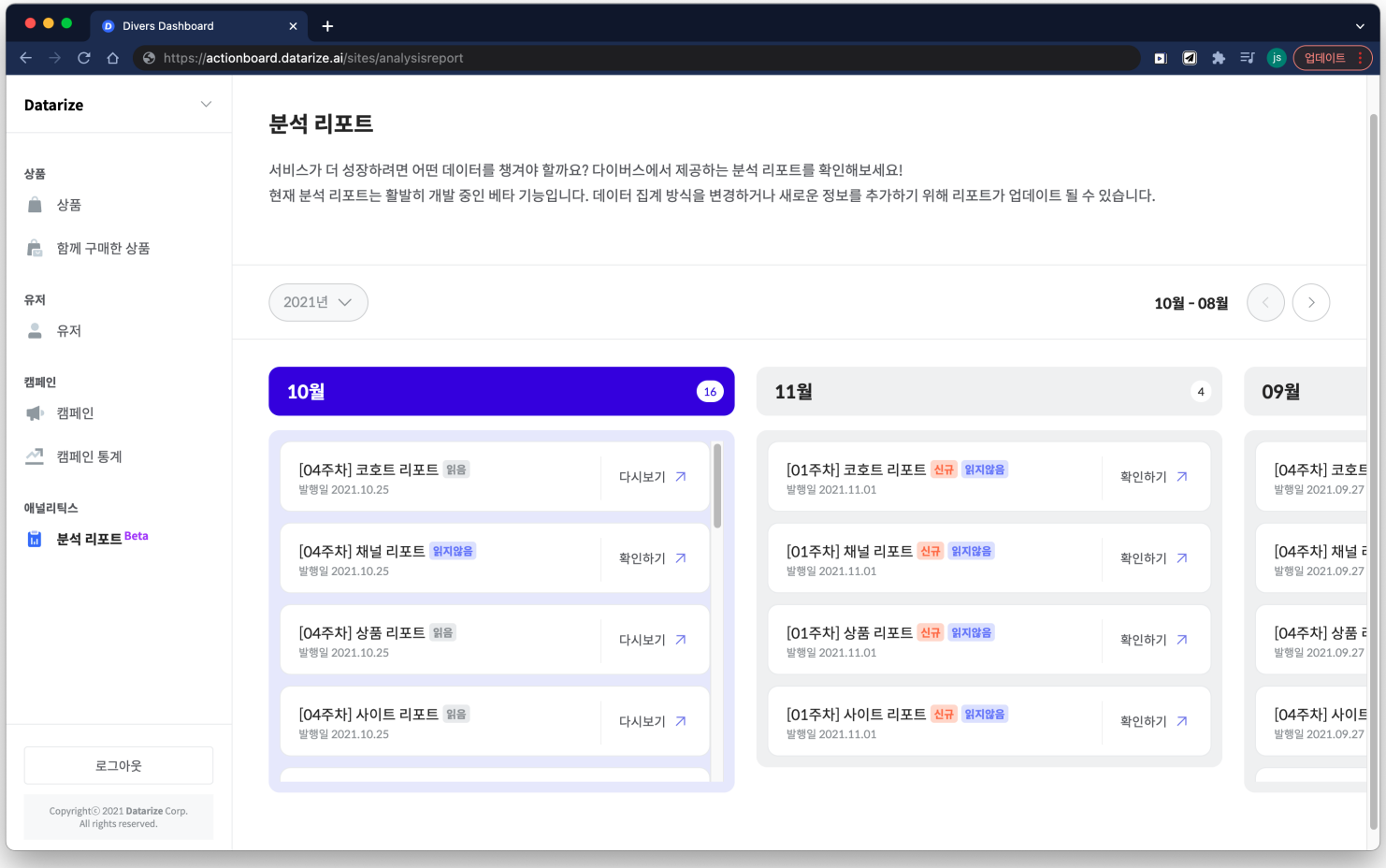
- 정식 페이지 개발 프로세스를 태우기 전, 미리 리포트 형태로 분석을 제공하자
- 이를 기반으로 정량, 정성적인 피드백을 얻고, 첫 단추를 잘 꿴 페이지를 개발해보자!

왜 R 마크다운인가?

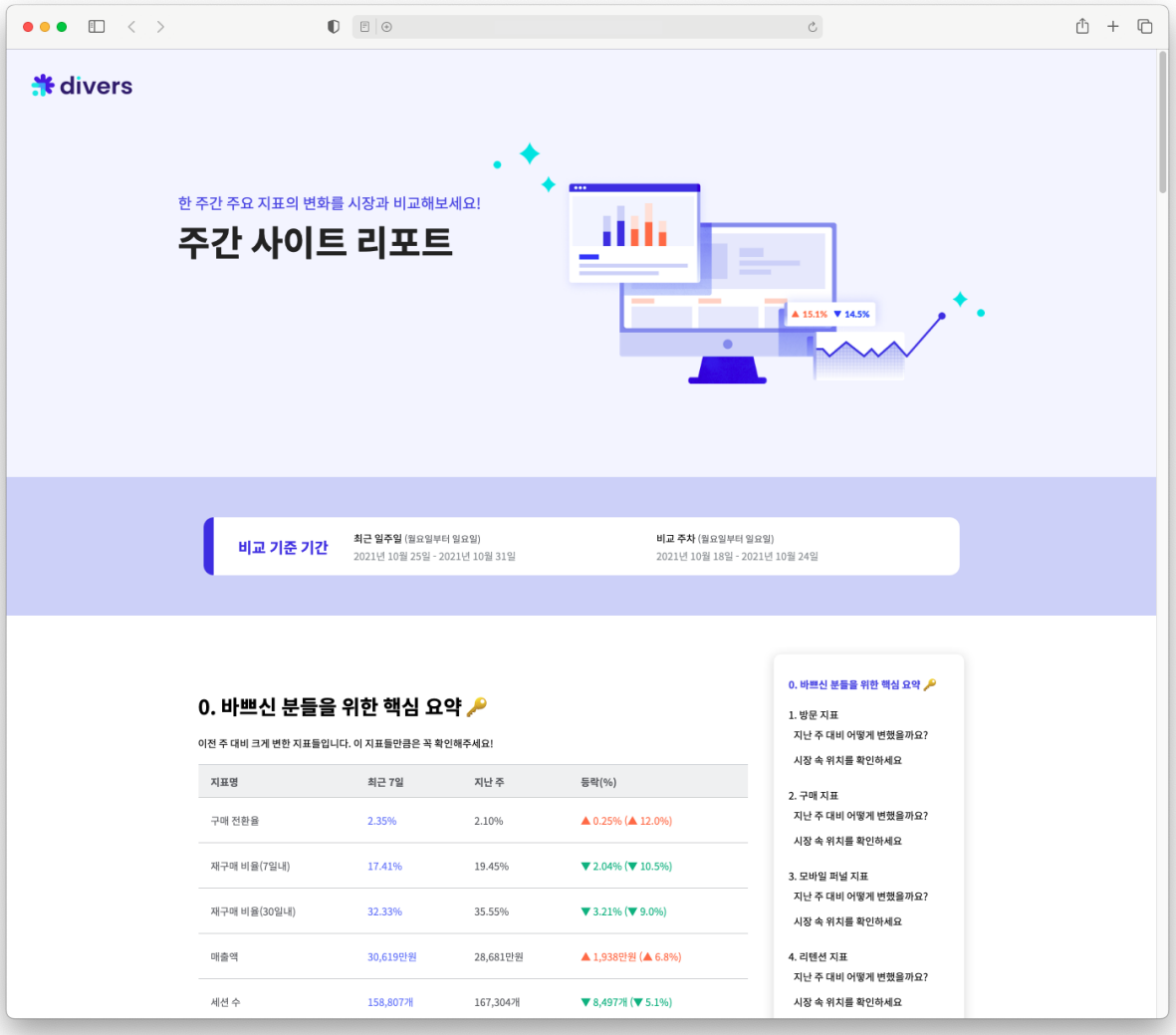
- 디자인이나 개발 리소스를 크게 들이지 않고, 분석가 혼자서도 만들 수 있음
- ggplot2, kableExtra, DT과 같은 패키지들을 통해 다양한 리포트 컨텐츠를 빠르게 만들어 볼 수 있음
- HTML로 Export하면 바로 웹 상에 호스팅할 수 있음
- HTML, CSS를 직접 수정하여 높은 수준의 커스터마이징이 가능함
- ~~그냥 내가 R 유저라서~~

리포트 시스템 결과물

- 고객사의 어드민 화면에서 리포트 목록 제공



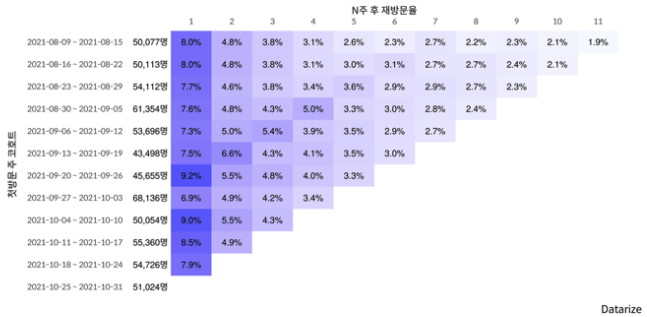
- 다양한 타입의 분석 리포트를 빠르게 생성하고 변화시키며 고객사에게 제공



1. 첫방문자 코호트의 재방문 리텐션

첫방문자의 재방문율이 높았던 기간은 언제인지 살펴보고 해당 기간동안 했던 마케팅 정책이나 기획전, 이벤트, 상품 구성이 무엇이 있었는지 살펴보세요. 그리고 첫방문자의 재방문율이 높았던 기간에 했던 것들을 자동화할 수 있는 방안을 고민해보세요! 다이버스와 함께 하고 싶은 좋은 아이디어가 있다면 hello@datarize.ai로 제안 주세요 :)

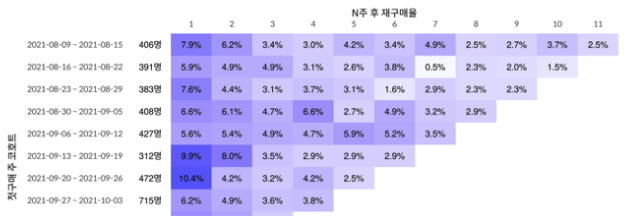
혹시 다이버스에 가입한 지 얼마되지 않으셨나요? 초반 방문 주 코호트의 재방문율이 유난히 높게 잡히는 것이 정상입니다. 시간이 지남에 따라 지표가 정상화 되어갑니다. 구매, 회원가입 코호트 분석 위주로 살펴보세요!



첫방문 주 코호트		1	2	3	4	5	6	7	8	9	10	11
2021-08-09 ~ 2021-08-15	50,077명	8.0%	4.8%	3.8%	3.1%	2.6%	2.3%	2.7%	2.2%	2.3%	2.1%	1.9%
2021-08-16 ~ 2021-08-22	50,113명	8.0%	4.8%	3.8%	3.1%	3.0%	3.1%	2.7%	2.7%	2.4%	2.1%	
2021-08-23 ~ 2021-08-29	54,112명	7.7%	4.6%	3.8%	3.4%	3.6%	2.9%	2.9%	2.7%	2.3%		
2021-08-30 ~ 2021-09-05	61,354명	7.6%	4.8%	4.3%	5.0%	3.3%	3.0%	2.8%	2.4%			
2021-09-06 ~ 2021-09-12	53,696명	7.3%	5.0%	5.4%	3.9%	3.5%	2.9%	2.7%				
2021-09-13 ~ 2021-09-19	43,498명	7.5%	6.6%	4.3%	4.1%	3.5%	3.0%					
2021-09-20 ~ 2021-09-26	45,655명	9.2%	5.5%	4.8%	4.0%	3.3%						
2021-09-27 ~ 2021-10-03	68,136명	6.9%	4.9%	4.2%	3.4%							
2021-10-04 ~ 2021-10-10	50,054명	9.0%	5.5%	4.3%								
2021-10-11 ~ 2021-10-17	55,360명	8.5%	4.9%									
2021-10-18 ~ 2021-10-24	54,726명	7.9%										
2021-10-25 ~ 2021-10-31	51,024명											

2. 첫구매자 코호트의 재구매 리텐션

장기간의 매출 성장을 위해서 매우 중요한 리포트입니다. 첫구매 후 재구매율이 낮은 eCommerce는 신규방문자를 끊임없이 유입시키지 않으면 성장이 쉽게 정체되기 때문입니다. 첫구매 후 재구매율이 계속해서 상승하는 추세이면 가장 좋고, 그렇지 않고 하락한다면(=파란색이 점점 연해진다면) 대책을 고민해야 합니다.



첫구매 주 코호트		1	2	3	4	5	6	7	8	9	10	11
2021-08-09 ~ 2021-08-15	406명	7.9%	6.2%	3.4%	3.0%	4.2%	3.4%	4.9%	2.5%	2.7%	3.7%	2.5%
2021-08-16 ~ 2021-08-22	391명	5.9%	4.9%	4.9%	3.1%	2.6%	3.8%	0.5%	2.3%	2.0%	1.5%	
2021-08-23 ~ 2021-08-29	383명	7.6%	4.4%	3.1%	3.7%	3.1%	1.6%	2.9%	2.3%	2.3%		
2021-08-30 ~ 2021-09-05	408명	6.6%	6.1%	4.7%	6.6%	2.7%	4.9%	3.2%	2.9%			
2021-09-06 ~ 2021-09-12	427명	5.6%	5.4%	4.9%	4.7%	5.9%	5.2%	3.5%				
2021-09-13 ~ 2021-09-19	312명	9.9%	8.0%	3.5%	2.9%	2.9%	2.9%					
2021-09-20 ~ 2021-09-26	472명	10.4%	4.2%	3.2%	4.2%	2.5%						
2021-09-27 ~ 2021-10-03	715명	6.2%	4.9%	3.6%	3.8%							

리포트 시스템 결과물

- 오픈율과 같은 정량적 데이터 + 고객사의 피드백을 통해 분석 컨텐츠에 대한 니즈 파악!
- 디자이너와의 협업을 통해 자체 Rmd 템플릿을 개발하여, 리포트 퀄리티 상승!

```
---
title: '주간 사이트 리포트'
subtitle: '한 주 동안 무슨 일이 있었는지 데이터로 확인하세요!'
output: **diversReportTemplate::divers_report_html_format**
---
```

리포트 생성기 구조

```
divers_report/
├── renv/
├── .Rprofile, renv.lock, .gitignore
├── Dockerfile, docker-compose.yaml
├── **run.R**
└── src/
    ├── site/
    |   ├── report.Rmd
    |   └── utils.R
    ├── cohort/
    ├── data.R
    └── utils.R
```

- `run.R` : 최종 코드 실행
- `src/data.R` : 데이터 로딩 및 전처리 함수
- `src/utils.R` : 범용적으로 사용되는 유틸 함수
- `src/<report-type>/report.Rmd` : 해당 리포트의 R 마크다운
- `src/<report-type>/utils.R` : 해당 리포트에서 사용되는 유틸 함수

리포트 생성 플로우

1. 리포트 생성에 필요한 데이터를 불러오고 전처리
2. Rmd를 HTML 타입으로 로컬 서버에 랜더링
3. 랜더링된 HTML 파일을 AWS S3 버킷에 업로드
4. S3 버킷에 업로드된 리포트 링크를 고객사 어드민 화면과 연결
5. 고객사에게 리포트 발행 알림 이메일 발송

리포트 생성 플로우 : Rmd → HTML

- `knitr::render` 함수를 사용하여, 로컬 서버에 HTML 형식으로 Rmd를 랜더링
- **[Tip]** `params` 옵션을 사용하면 외부에서 Rmd 내부로 변수를 전달할 수 있음

```
# Input
# ㄴ 어떤 고객사인가? : target_site
# ㄴ 어떤 리포트인가? : target_report_type
# ㄴ 언제 발행하는가? : target_date

dir_name  <- target_site
file_name <- glue::glue("{target_report_type}_report_{target_date}.html")

knitr::render(
  glue::glue("src/{target_report_type}/report.Rmd"),
  params      = list(target_site = target_site, target_date = target_date),
  output_file = file_name,
  output_dir  = glue::glue("tmp_result/{dir_name}")
)
```

리포트 생성 플로우 : HTML → AWS S3 버킷

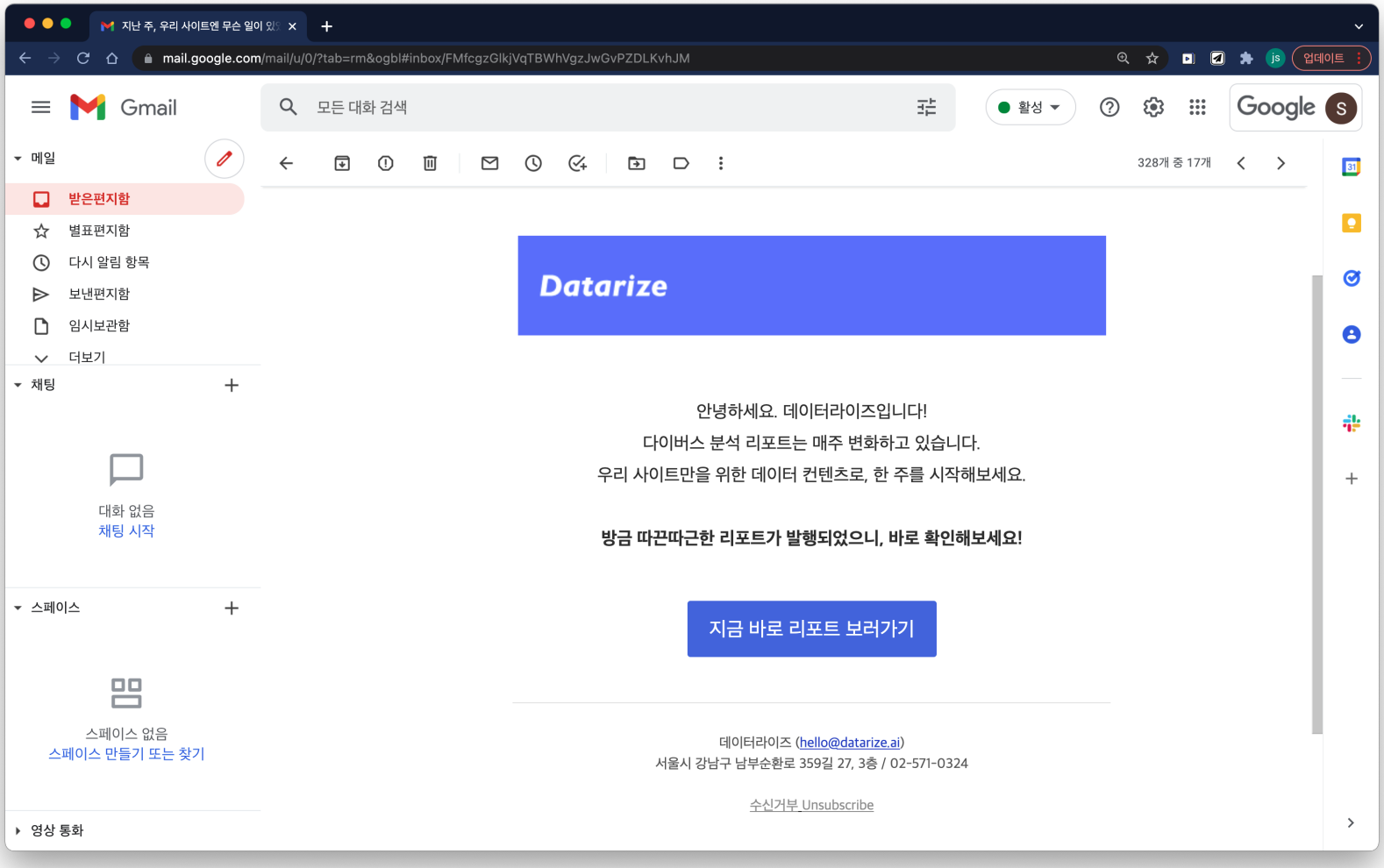
- `aws.s3::put_object` 함수를 사용하여, S3 버킷에 HTML 파일 업로드
- S3 버킷에 업로드된 리포트 링크를 각 고객사 화면에서 접속할 수 있도록 연결

```
aws.s3::put_object(
  file   = glue("tmp_result/{dir_name}/{file_name}"),
  object = glue("reports/{dir_name}/{file_name}"),
  bucket = "<bucket-name>"
)
```

리포트 생성 플로우 : 리포트 발행 알림 이메일 발송

- 현재는 자체 이메일 발송기를 사용하고 있으나, Stibee와 같은 이메일 솔루션도 활용 가능

```
STIBEE_API_KEY <- "<my-api-key>"
STIBEE_API_URL <- "<my-api-url>"

sending_info <- tibble(
    name = c("user_1", "user_2"),
    subscriber = c("user_1@gmail.com", "user_2@gmail.com")
)

for (i in seq_len(nrow(sending_info))) {

  request_body <- list(
    name        = sending_info[i, ]$name,
    subscriber  = sending_info[i, ]$subscriber
  )

  response <- httr::POST(
    url  = STIBEE_API_URL,
    body = request_body,
    encode = "json",
    add_headers(
      .headers = c(
        "Content-Type" = "application/json",
        "AccessToken" = STIBEE_API_KEY)
    )
  )

  print(glue("{sending_info[i, ]$name}/{sending_info[i, ]$subscriber} : {content(response)}"))
}
```


mail.google.com/mail/u/0/?tab=rm&ogbl#inbox/FMfcgzGlkjVqTBWhVgzJwGvPZDLKvhJM
업데이트
Gmail
모든 대화 검색
활성
Google
메일
받은편지함
별표편지함
다시 알림 항목
보낸편지함
임시보관함
더보기
채팅
대화 없음
채팅 시작
스페이스
스페이스 없음
스페이스 만들기 또는 찾기
영상 통화
328개 중 17개
Datarize
안녕하세요. 데이터라이즈입니다!
다이버스 분석 리포트는 매주 변화하고 있습니다.
우리 사이트만을 위한 데이터 컨텐츠로, 한 주를 시작해보세요.
방금 따끈따근한 리포트가 발행되었으니, 바로 확인해보세요!
지금 바로 리포트 보러가기
데이터라이즈 (hello@datarize.ai)
서울시 강남구 남부순환로 359길 27, 3층 / 02-571-0324
수신거부 Unsubscribe

PART2

운영 환경 구축하기

프로덕션 수준의 운영 환경에서 필요한 것들

- 로컬, 서버, 쿠버네티스 등 어떤 환경에서도 일관적으로 동작해야 한다.
- 시간이 지나도 현재와 동일하게 동작해야 한다.
- 문제가 발생했을 때, 어디서 어떤 에러가 발생했는지 바로 확인할 수 있어야 한다.
- 다른 서비스에 의도치 않은 영향을 미치지 않도록 환경이 서로 분리되어 있어야 한다.

프로덕션 수준의 운영 환경을 구축하기 위한 고민

- 일관성을 위한 최고의 도구 : 도커 컨테이너
- 분명 저번달에는 잘 돌았는데... : 패키지 버전 관리
- 짜임새 있는 코드 구성을 위해 : 스크립트 모듈화
- 에러 기록하고 빠르게 대응하기 : 로깅과 알림

(1) 도커 컨테이너 : 왜 필요할까?

- 특정 R 버전을 사용해야 하거나, 프로젝트마다 서로 다른 R 버전이 필요한 경우
- 로컬, 서버, 쿠버네티스 환경 모두에서 일관적으로 동작해야 하는 경우

(1) 도커 컨테이너 : 어떻게 해야 할까?

- 프로젝트의 특성에 맞는 도커 세팅을 준비해둔다.
  - plumber API 서버 → `rocker/r-ver:4.1.1`
  - 분석 환경 → `rocker/tidyverse:latest`
  - 모델링 배치 작업 → `rocker/tidyverse:4.1.1` 또는 r-ver + 필요한 라이브러리만 설치
- 필요한 환경에 맞게 docker-compose 스크립트를 설정한다.
  - Prod → 일반적인 동작을 위한 설정
  - Dev → 로컬에서 개발하기 위한 설정
  - Amazon ECR → 컨테이너 레지스트리를 통해 관리해야 하는 경우 별도로 작성

(2) 패키지 버전 관리 : 왜 필요할까?

- 가능하면 최신 버전의 라이브러리를 사용할 것을 권장하는 R
- 라이브러리 업데이트로 인해 갑자기 만들어둔 기능이 동작하지 않으면 어떻게 하지?

(2) 패키지 버전 관리 : 어떻게 해야 할까?

- `renv` 를 사용한다.
  - renv는 lockfile을 통해 특정 시점의 라이브러리 상태를 저장한다.
  - lockfile 에는 다음과 같은 정보가 담겨있다.
    - renv 라이브러리 버전
    - 프로젝트에 사용된 R 버전
    - R 레포지토리 주소
    - 패키지별 정보와 설치 방식
- `renv` 가 동작하는 방식
  - `renv::snapshot()` 으로 lockfile 을 생성한다. → `renv.lock`
  - `renv::restore()` 를 사용해 저장된 환경에 맞게 다시 설치한다.

- 도커와 함께 사용하기
  - renv.lock 파일을 통해 라이브러리를 설치하도록 한다.
  - lock 파일이 변경되지 않았다면 도커의 캐시로 인해 빌드 시간을 단축할 수 있다.

```
# install dependencies
COPY ./renv.lock .
RUN Rscript -e "renv::restore()"

# copy the app
COPY app .
```

(3) 스크립트 모듈화 : 왜 필요할까?

- 소스 코드가 길어지고 내용이 복잡해지면 유지 관리가 어려워진다.
- `source` 를 통해 분리된 R 스크립트를 불러오면 전체 스코프에 영향을 줄 수 있다.
- `library` 는 패키지 내의 모든 함수를 attach 하기 때문에 함수가 꼬이는 경우가 발생한다.
- 모듈화를 통해 함수를 재사용하고 싶지만 패키지로 작성하고 싶지는 않을 때가 있다.

(3) 스크립트 모듈화 : 어떻게 해야 할까?

modules 와 box 를 사용한 모듈화

- modules

```
#### src/data.R ####
import("DBI")
import("RMySQL")
import("lubridate")
import("dplyr")

export("get_data_from_db")

get_data_from_db <- function() {
  # ...
}

#### run.R ####
Module <- modules::use("src/data.R")
data <- Module$get_data_from_db()
```

- box

```
#### src/data.R ####
box::use(
  DBI[dbGetQuery],
  RMySQL,
  lubridate,
  dplyr
)

#' @export
get_data_from_db <- function() {
  # ...
}

#### run.R ####
box::use(Module = src/data)
data <- Module$get_data_from_db()
```

(3) 모듈화에 대한 고민

- **회사 내 모든 구성원들이 사용하는 기능은 패키지로 모아서 작성해둔다.**
    - DB 컨넥션 및 데이터 전처리 함수
    - 그래프 디자인 템플릿
    - Rmarkdown 문서 디자인 템플릿
- **특정 프로젝트 내에서 반복적으로 사용되는 함수는 기능별로 모아서 스크립트를 분리한다.**
    - 기능 단위로 스크립트를 나누어 두면 프로젝트가 복잡해지더라도 원하는 부분을 쉽게 찾을 수 있다.
    - Git 으로 협업하기 용이해진다.
- **스크립트를 작성할 때 가능하면 라이브러리 전체를 임포트하는 것을 피하고, 라이브러리의 특정 함수를 임포트하도록 한다.**
    - 모든 함수를 attach하면, 현재 스크립트에서 해당 라이브러리가 실제로 사용되고 있는지 확인하는 것이 어렵다.
    - 따라서 조금 불편하더라도 최대한 명시적으로 함수가 어느 라이브러리를 통해 임포트되었는지 코드에 기록하도록 한다.

(4) 로깅과 알림 : 왜 필요할까?

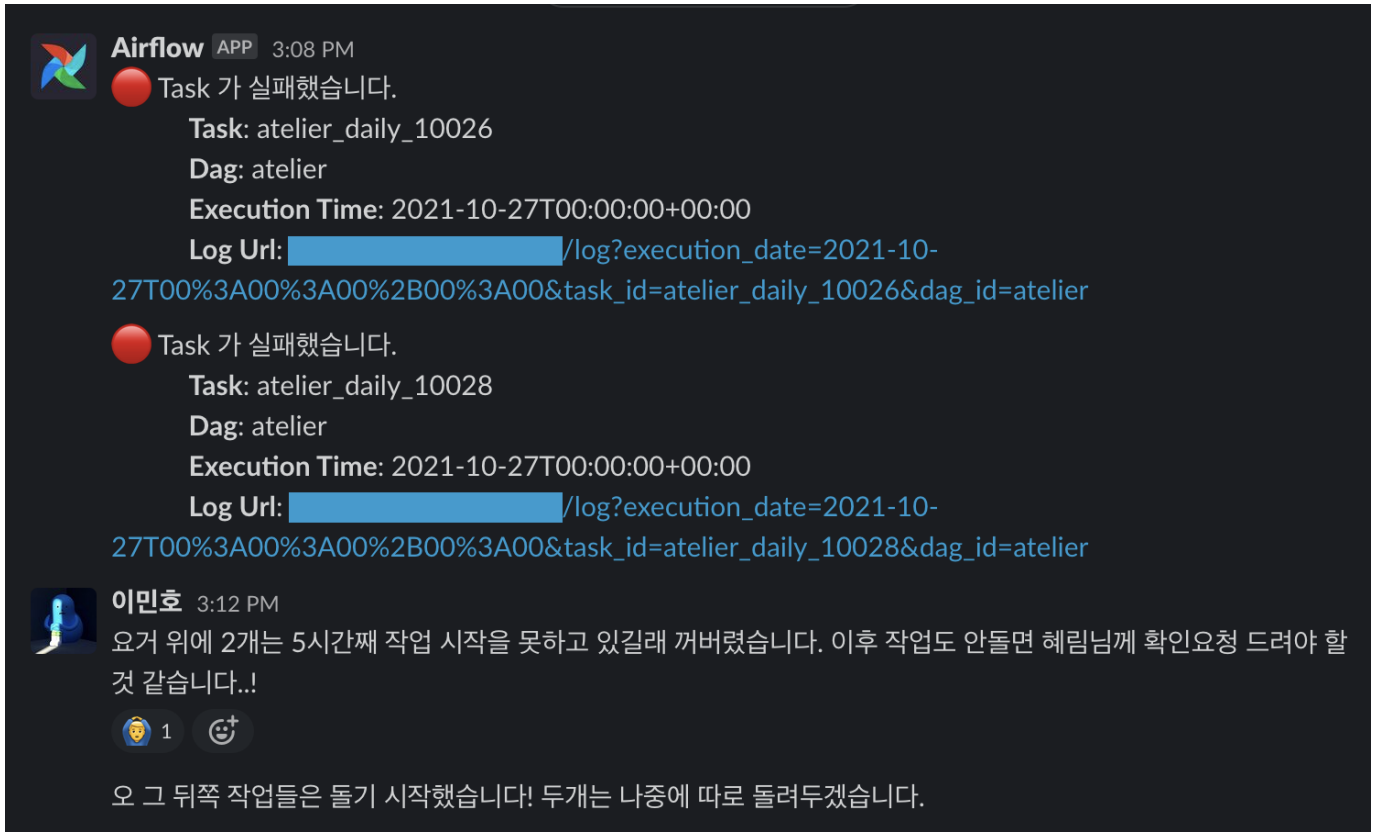
- 분석을 할 때는 직접 보면서 에러가 발생하지 않도록 해결하면 된다.
- 운영환경에서는 정해진 주기로 자동으로 동작하다보니, 언제 어디서 어떤 에러가 발생했는지 잘 기록해서 담당자가 빠르게 확인할 수 있어야 한다.

(4) 로깅과 알림 : 어떻게 해야 할까?

- logger 로 출력해두기
  - 도커와 쿠버네티스 환경을 사용하다보니, 에러가 발생할 때 콘솔에 출력해두면 문제가 생겼을 때 확인할 수 있다.
  - logger 라이브러리를 통해 에러가 발생한 시간, 로그 레벨 등을 쉽게 출력할 수 있다.

```
logger::log_info('Loading data')
logger::log_warn('The function is deprecated')
logger::log_error('Failed to save RDS file')
# INFO [2021-11-04 11:16:41] Loading data
# WARN [2021-11-04 11:16:41] The function is deprecated
# ERROR [2021-11-04 11:16:41] Failed to save RDS file

name <- 'conference'
logger::log_info('Loading data for {name}')
# INFO [2021-11-04 11:17:57] Loading data for conference
```

- 슬랙을 사용한 에러 메시지 발송
  - 현재는 주기적으로 동작하는 배치 작업을 Airflow 로 관리한다.
  - 작업이 실패할 경우 Airflow 에서 슬랙으로 메시지를 보내도록 하고 있으며, 첨부된 Log URL에서 출력해둔 에러 메시지를 확인할 수 있다.



R로 프로덕션 수준의 운영 환경을 구성하면서 느낀점

- 운영에 필요한 도구가 많이 부족하지만, 그래도 조금씩 생태계가 확장되고 있다는 것을 체감하고 있다.
- 다른 언어를 사용하다 돌아와보니, R 고유의 특성과 생태계로 인해 개발하는 것이 쉽지 않다.
- 하지만 시각화나 rmarkdown 등 R의 강점을 온전히 누리기 위해 많은 사람들이 고민하고 있으며, 그 결과 R로도 다양한 개발 프로젝트를 시도할 수 있는 환경이 갖추어지고 있는 것 같다.

R을 사랑하는 데이터 분석가, 데이터 사이언티스트 항상 환영합니다!

데이터라이즈는 데이터로 쏘아올릴 🚀**로켓에 함께 탑승하실 분**을 기다립니다.

https://team.datarize.ai/

감사합니다!