



MISE EN PRODUCTION DE CODE R

RETOUR D'EXPERIENCE

Nicolas Greffard PhD, Data Scientist

Qui ?

Expandium

- 13 ans

Métier

- Monitoring télécom

Solutions

- Software / Hardware => carte d'acquisition réseau + décodage/traitement en temps réel

Clients

- Railway => 9 millions de lignes = 2 semaines
- Opérateurs publics (SFR etc...) => 9 milliards de lignes = 3 jours

Nos OUTILS, NOS DONNÉES

QATS Railway by expandium

Expandium Administrator

Tracing

- ETCS Call
- Transaction
- VGCS / VBS / REC
- HO
- Subscriber Matrix
- HDLC Frame Error

ETCS

- GSM-R
- GPRS
- VGCS-VBS-REC
- Radio
- Roaming
- Handover
- Reference Analytics

Transaction Type: Call
Direction Name: All
IMSI:
MSISDN:
IMEI:
Called Number:
Calling Number:
NID_ENGINE:
NID_OPERATIONAL:

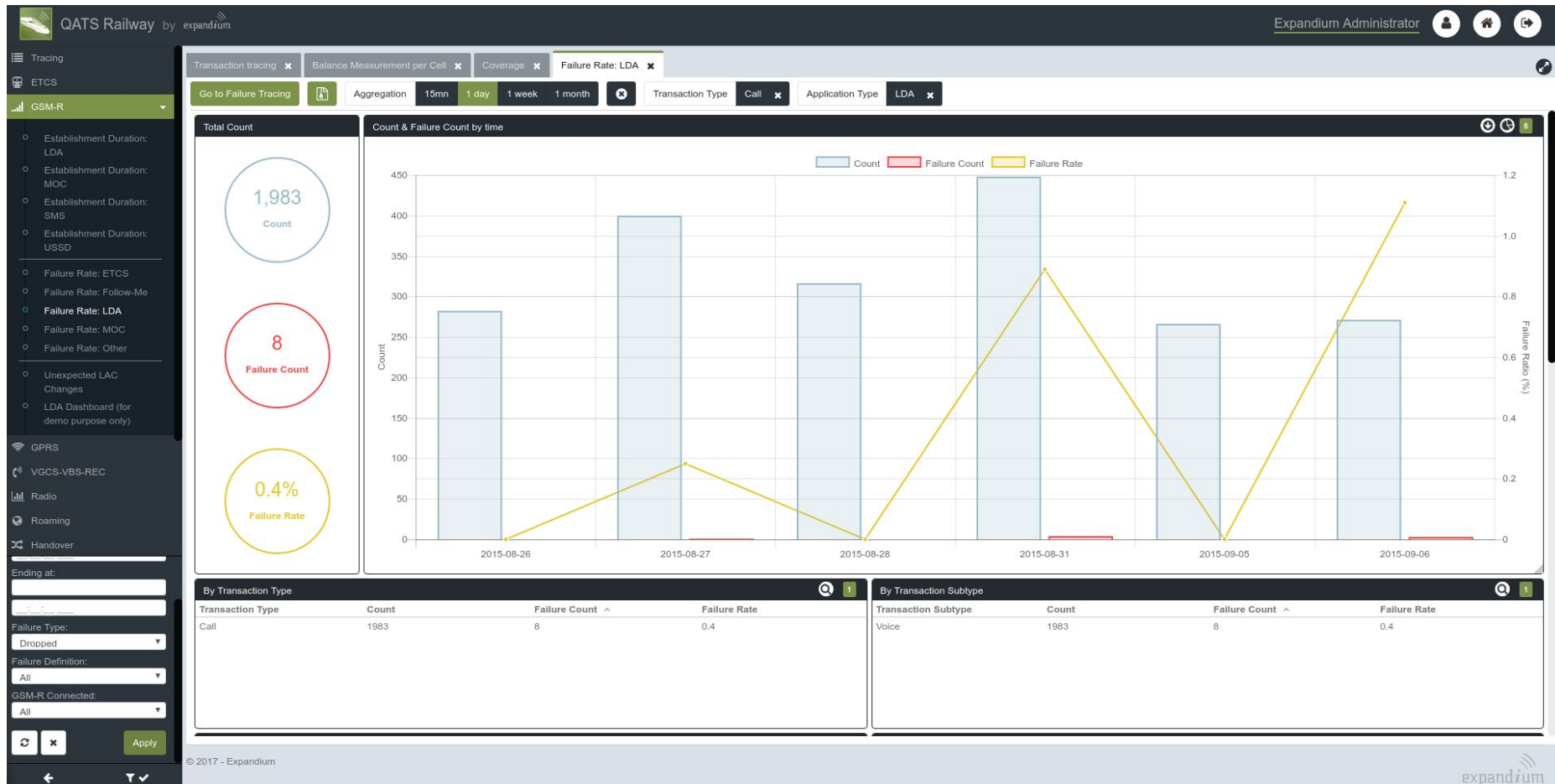
Transaction tracing

Starting at	Ending at	Establishment Delay	Transaction duration	Start LAC	Start CI	Stop LAC	Stop CI	TMSI	Reallocated TMSI	IMSI	MSISDN
2015-09-06 20:31:57.150	2015-09-06 20:32:02.920	4s 58ms	5s 770ms	1	30102	1	30102	412b22b0	412b236d	262103000034865	491835816253
2015-09-06 20:31:43.375	2015-09-06 20:31:49.824	4s 227ms	6s 449ms	1	10111	1	10111	412b18ee		204218000214213	
2015-09-06 20:31:40.234	2015-09-06 20:31:43.820	2s 664ms	3s 586ms	1	11172	1	11172	412b15ef		204218000013311	31840821533
2015-09-06 20:31:30.175	2015-09-06 20:31:34.873	4s 33ms	4s 698ms	1	13571	1	13571	412b2053		204218000116997	31840816825
2015-09-06 20:31:12.417	2015-09-06 20:31:45.333	2s 822ms	32s 916ms	1	15162	1	15162	412ac9b5		204218000119753	31840823303
2015-09-06 20:31:06.642	2015-09-06 20:31:59.332	2s 674ms	52s 690ms	1	16051	1	16051	412b22c2		204218000075590	31840820496
2015-09-06 20:30:56.634	2015-09-06 20:31:21.242	3s 768ms	24s 608ms	1	16061	1	16061	412b2249		204218000013875	31840821815
2015-09-06 20:30:38.857	2015-09-06 20:31:04.605	2s 838ms	25s 748ms	1	13571	1	13571	412b2053		204218000116997	31840816825
2015-09-06 20:30:26.707	2015-09-06 20:31:39.422	3s 919ms	1min 12s 715ms	1	13521	1	13521	412b1f64		204218000012697	31840821326
2015-09-06 20:30:23.082	2015-09-06 20:30:48.837	3s 828ms	25s 755ms	1	11242	1	11242	412b211e		204218000016257	31840822006
2015-09-06 20:30:13.506	2015-09-06 20:31:04.437	5s 356ms	50s 931ms	1	13711	1	13711	412b1a9e		204218000038503	31840820451
2015-09-06 20:29:16.457	2015-09-06 20:29:58.975	6s 685ms	42s 518ms	1	10231	1	10231	412b22f7	412b2306	262103000026628	491835810692
2015-09-06 20:29:12.459	2015-09-06 20:30:12.251	2s 320ms	59s 792ms	1	15071	1	15071	412afa80		204218000214599	
2015-09-06 20:29:04.982	2015-09-06 20:29:40.758	2s 641ms	35s 776ms	1	11142	1	11141	412b15ef		204218000013311	31840821533
2015-09-06 20:28:56.449	2015-09-06 20:29:06.997	3s 598ms	10s 548ms	1	10231	1	10231	412b20ab	412b22f7	262103000026628	491835810692
2015-09-06 20:28:52.589	2015-09-06 20:29:22.989	3s 779ms	30s 400ms	1	16202	1	16202	40f50259		204218000427501	
2015-09-06 20:28:35.786	2015-09-06 20:29:19.244	3s 131ms	43s 458ms	1	10292	1	10292	412b22e6		204218000010456	31840820026
2015-09-06 20:28:31.089	2015-09-06 20:28:55.839	4s 24ms	24s 750ms	1	19351	1	19351	412b0e1a		204218000214833	
2015-09-06 20:28:27.495	2015-09-06 20:29:01.666	3s 263ms	34s 171ms	1	15061	1	15061	412b1a08	412b22e8	262103000035651	491835816456
2015-09-06 20:28:25.003	2015-09-06 20:29:53.421	2s 565ms	1min 28s 418ms	1	17842	1	17842	412aa5db		204218000119387	31840823120
2015-09-06 20:28:20.058	2015-09-06 20:28:53.675	3s 315ms	33s 617ms	1	13542	1	13542	412b1963	412b22e3	206028000804122	
2015-09-06 20:28:13.259	2015-09-06 20:29:14.844	2s 612ms	1min 1s 585ms	1	41431	1	41431	412b00a2		204218000214207	

Total : 5041 rows, 28 columns | Last update: 10:23:27 | Sorted by start time desc

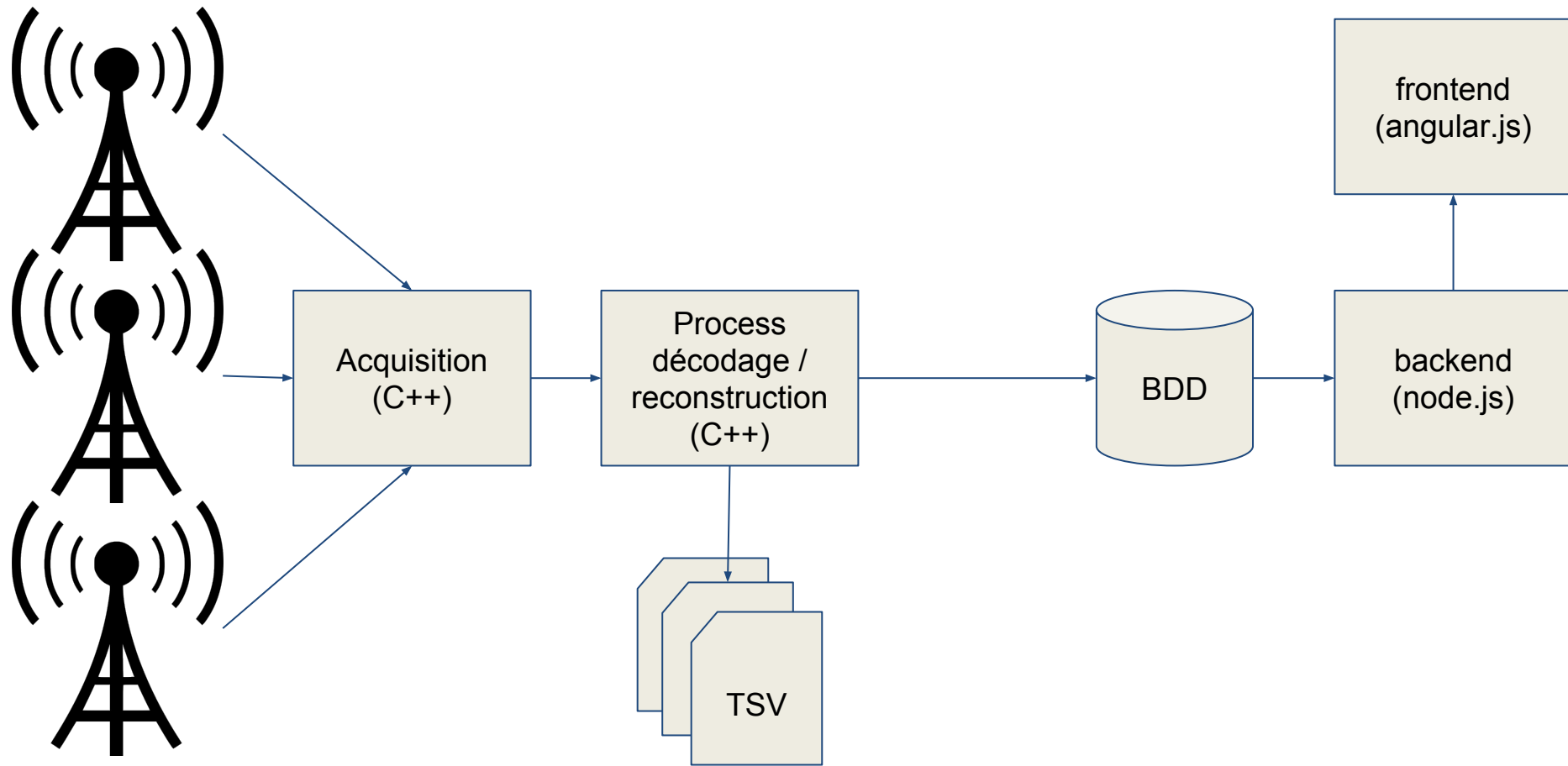
© 2017 - Expandium

Nos OUTILS, NOS DONNÉES





ARCHITECTURE SIMPLIFIÉE

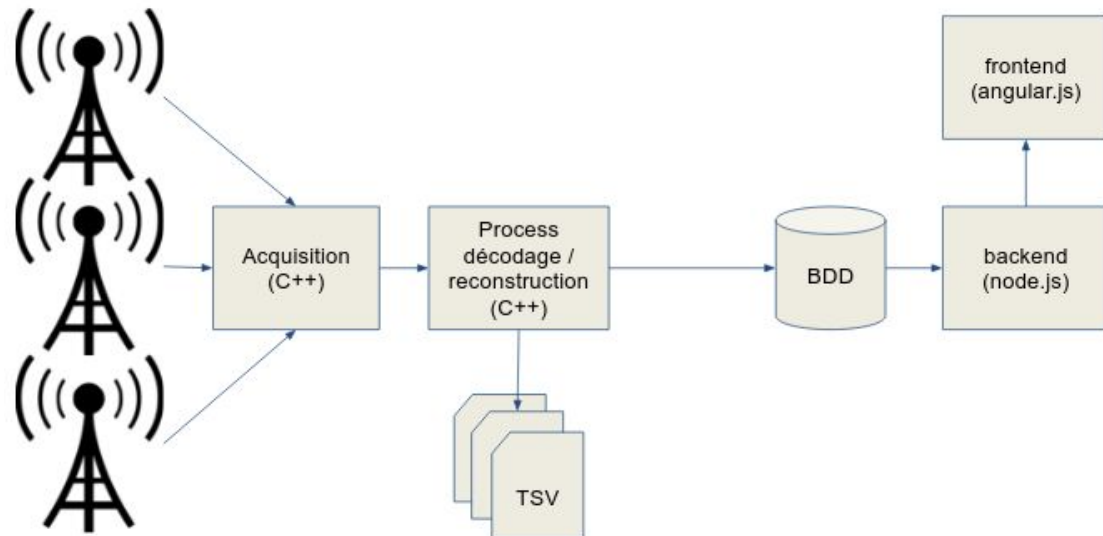


ARCHITECTURE SIMPLIFIÉE

Système Linux Debian

Données sensibles

- Système fermé
- Livraisons sous la forme d'un "CD" (iso) Debian avec nos packages ajoutés au repo



PROJET DATA SCIENCE #1

REFERENCE ANALYTICS

Besoin

Identifier les anomalies dans le signal radio d'appels de trains

Méthode

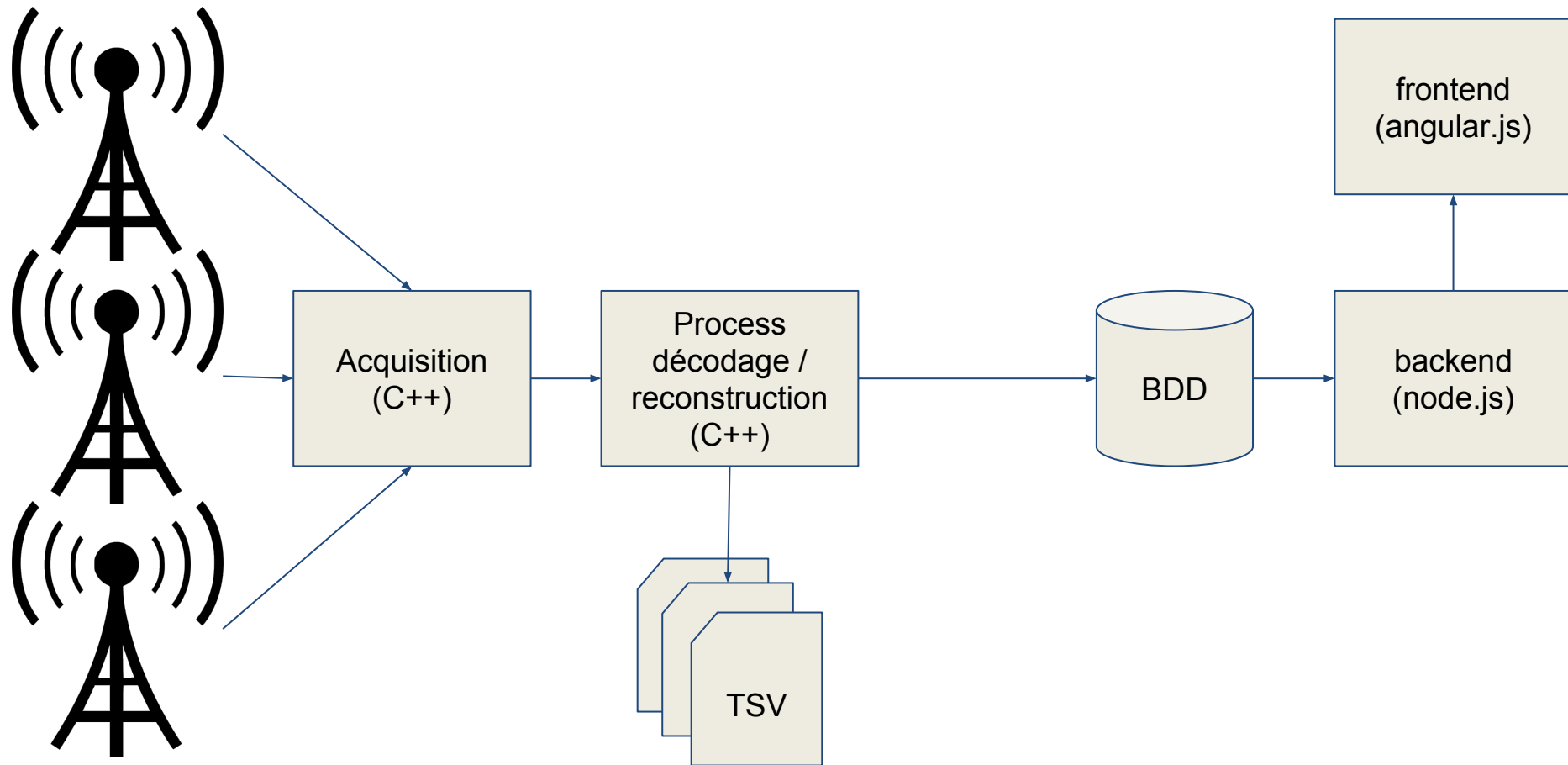
- Extraction d'appels de références (même mobile, même rails, etc..) : **Clustering**
- Comparaison du nouvel appel et de sa référence : **tests statistiques**, détection d'anomalies locales (**analyse de séries temporelles & AutoEncoder**), **analyse de graphe**

Solution en pratique

- Environ 7000 LOC R, 1 dev (=> I/O; pre/post-processing, pipeline d'analyse, persistance, etc..)

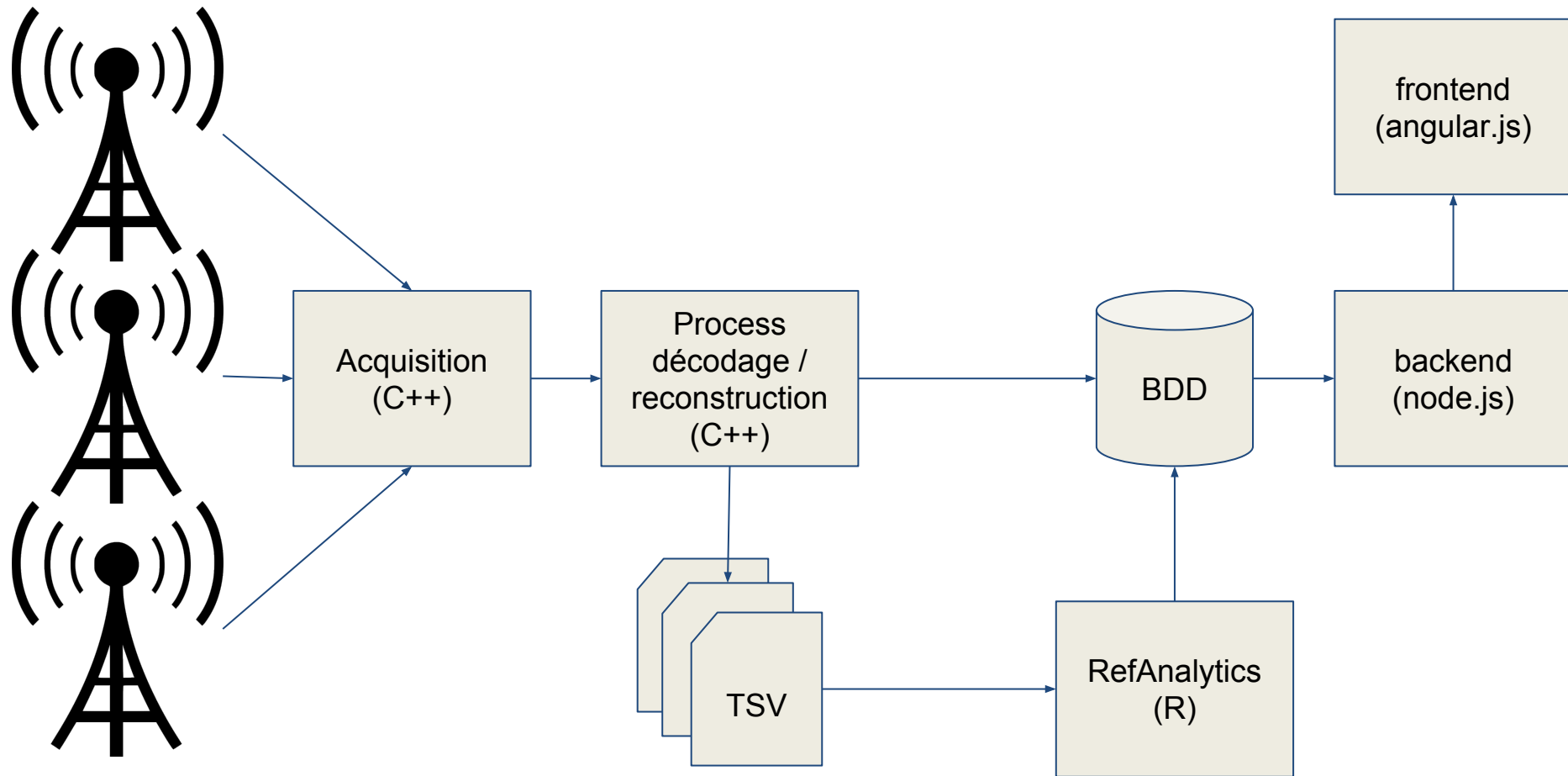
PROJET DATA SCIENCE #1

REFERENCE ANALYTICS



PROJET DATA SCIENCE #1

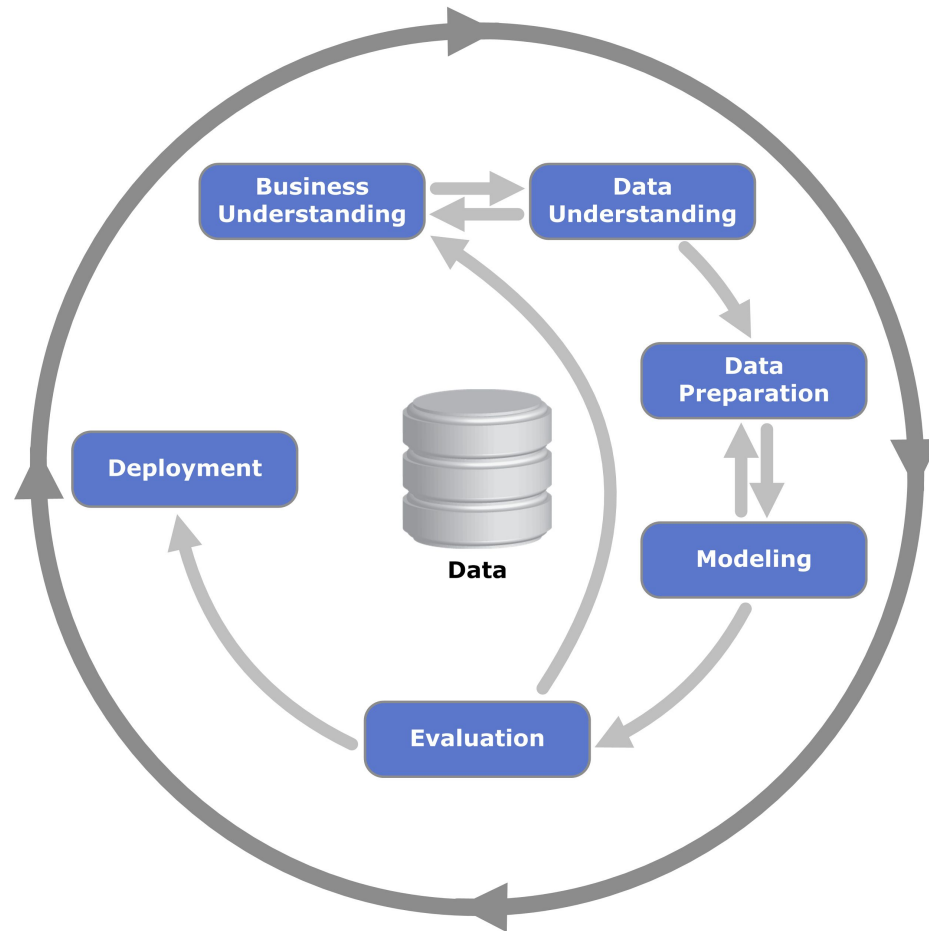
REFERENCE ANALYTICS



PROJET DATA SCIENCE #1

REFERENCE ANALYTICS

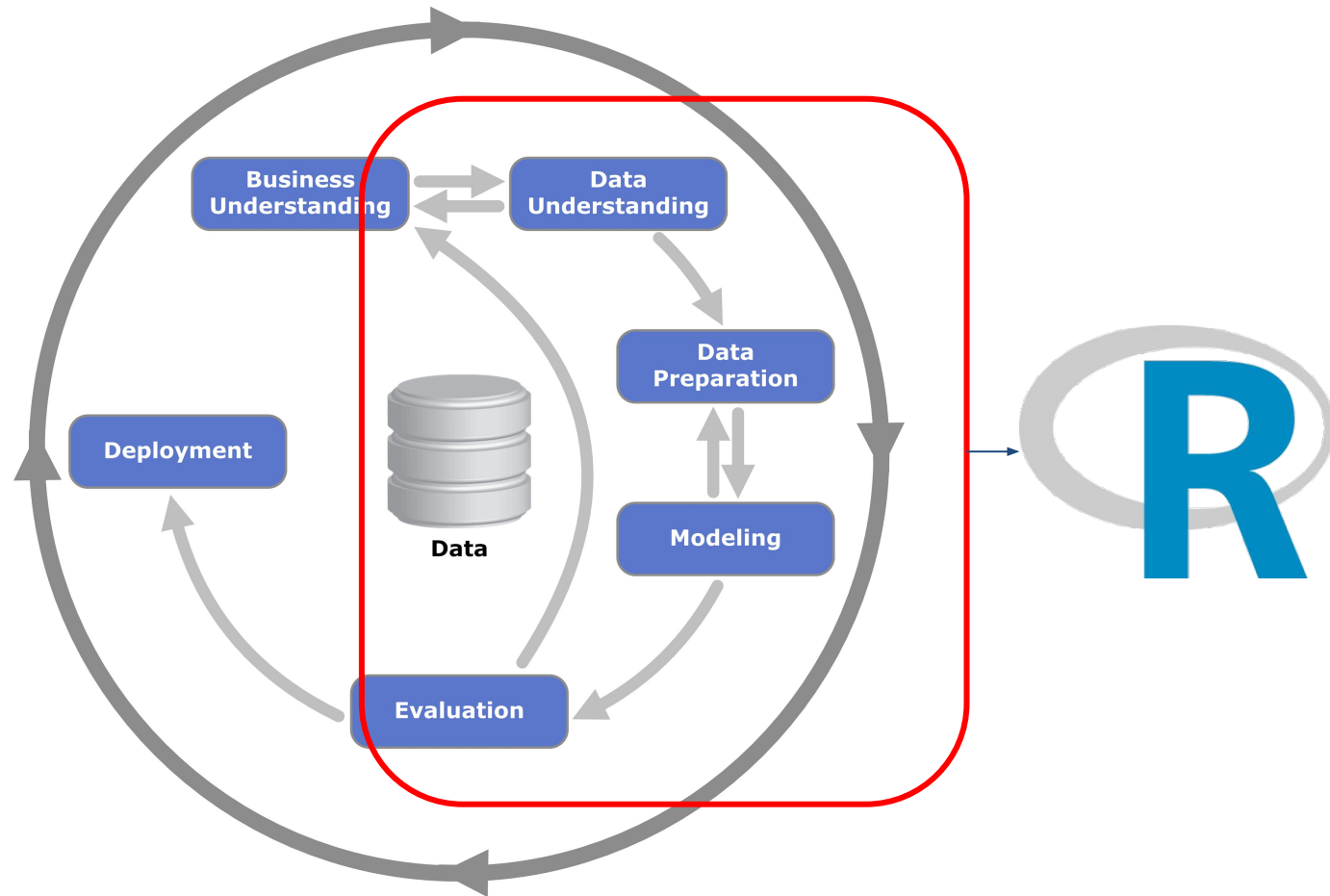
Cycle de vie



PROJET DATA SCIENCE #1

REFERENCE ANALYTICS

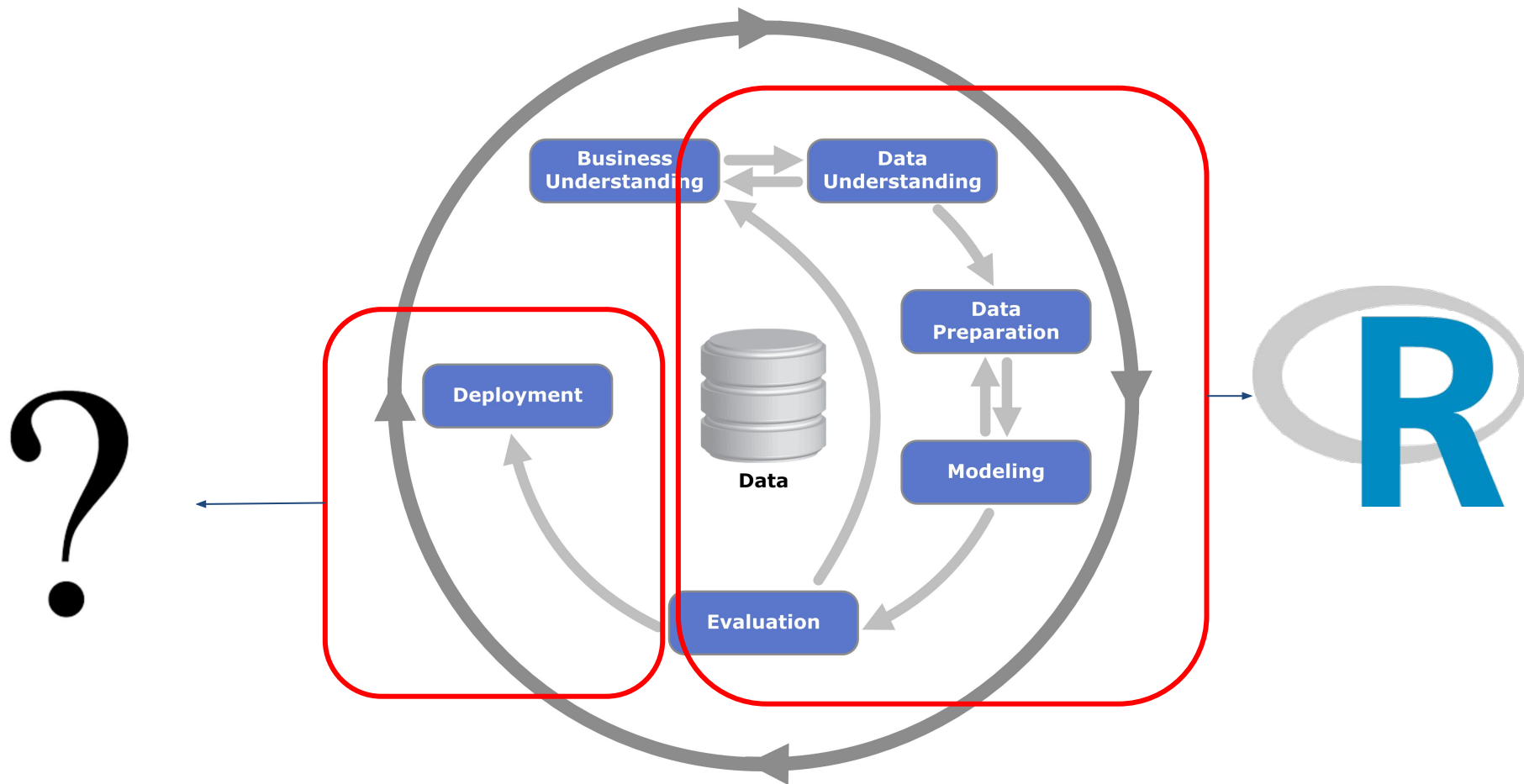
Cycle de vie



PROJET DATA SCIENCE #1

REFERENCE ANALYTICS

Cycle de vie



PROJET DATA SCIENCE #1

REFERENCE ANALYTICS

Production Ready Code industrialisable

Notions subjectives, Kezako ?

PROJET DATA SCIENCE #1

REFERENCE ANALYTICS

Production Ready Code industrialisable

Notions subjectives, Kezako ?

Code qui a passé une code review, qui a des tests unitaires, qui a des tests d'intégrations, dont le fonctionnel a été validé



Pierrick

PROJET DATA SCIENCE #1

REFERENCE ANALYTICS

Production Ready Code industrialisable

Notions subjectives, Kezako ?



Pierrick



Anthony

et qui a été testé dans
un environnement
isoprod ?

PROJET DATA SCIENCE #1

REFERENCE ANALYTICS

Production Ready Code industrialisable

Notions subjectives, Kezako ?

code qui respecte les standards de
la société ? Code lisible,
reprenable par un autre membre
de l'équipe ?



Pierrick



Anthony

PROJET DATA SCIENCE #1

REFERENCE ANALYTICS

Production Ready Code industrialisable

Notions subjectives, Kezako ?

Code -décidé par une ou plusieurs personnes- à mettre à disposition pour d'autres personnes



Pierrick



Anthony



Florent

PROJET DATA SCIENCE #1

REFERENCE ANALYTICS

Production Ready Code industrialisable

Notions subjectives, Kezako ?

c'est production ready quand tu es
prêt à en assumer les
conséquences



Pierrick



Anthony



Florent



Armand

PROJET DATA SCIENCE #1

REFERENCE ANALYTICS

Production Ready Code industrialisable

Notions connexes, qui comprennent (liste non exhaustive):

Production Ready

- Code **relu** par un **tier**, **réutilisable** (par quelqu'un d'autre), dont on **assume** la paternité/maternité
- Code **testé** unitairement, fonctionnellement et en **intégration** (idéalement dans un environnement isoprod == comme chez le client, si il y a un client)

Code industrialisable

- Code qui puisse rentrer dans le processus (idéalement **automatisé**) de **déploiement** et de livraison de l'entreprise

PROJET DATA SCIENCE #1

REFERENCE ANALYTICS

pRoduction, industRialisation, c'est possible ?

En tout cas ce n'est pas impossible; **ce qu'on a fait** :

Relecture / réutilisation / maintenabilité / standards :

- Code **versionné** (git)
- Code **reviews** (même si 1 seul dev R)
- Réunions de **partage** (vulgarisation des algorithmes et méthodes utilisées)
- **Documentation** scientifique (sur notre outil confluence + carnet de recherche)
- **Standards** de style : [Google](#) ou [Hadley Wickhman](#)

PROJET DATA SCIENCE #1

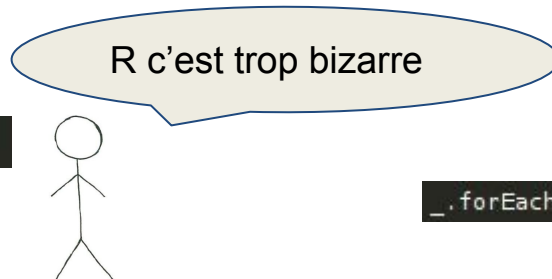
REFERENCE ANALYTICS

pRoduction, industRialisation, c'est possible ?

En tout cas ce n'est pas impossible; **ce qu'on a fait** :

Relecture / réutilisation / maintenabilité / standards :

- Code **versionné** (git)
- Code **reviews** (même si 1 seul dev R)
- Réunions de **partage** (vulgarisation des algorithmes et méthodes utilisées)
- **Documentation** scientifique (sur notre outil confluence + carnet de recherche)
- **Standards** de style : [Google](#) ou [Hadley Wickhman](#)



```
df.var <- x$col[[1]][1]
```

```
_forEach()
```

```
template <typename S1, typename S2 = std::string,  
         typename = typename std::enable_if<!std::is_same<S1, Cust  
         >::value  
         >::type>
```

```
x?.var == true
```

PROJET DATA SCIENCE #1

REFERENCE ANALYTICS

Production, industrialisation, c'est possible ?

En tout cas ce n'est pas impossible; **ce qu'on a fait** :

Tests / Intégration / Outillage / Déploiement :

- Ca devient intéressant
- R est **intégrable** aux processus existant d'intégration continu :
 - RefAnalytics fait de l'analyse statistique en batch, lancé à intervalles réguliers
 - On déploie nos outils sous forme de **packages debian**
 - ⇒ Faisons la même chose pour le code R
 - R propose un système de packaging : utilisons le !

PROJET DATA SCIENCE #1

REFERENCE ANALYTICS

production, industrialisation, c'est possible ?

En tout cas ce n'est pas impossible; ce qu'on a fait :

Tests / Intégration / Outillage / Déploiement :

Description du package R:

```
Package: qatsDME
Type: Package
Title: Data Mining Engine for QATS
Version: 1.1
Date: 2015-10-07
Author: Nicolas Greffard
Maintainer: <nicolas.greffard@expandium.com>
Description: ABIS data DM processing
License: Copyright (C) Expandium - All Rights Reserved
Depends: R (>= 2.0.0), RColorBrewer, mclust, \
         tseries, RSQLite, stringr, jsonlite, \
         doParallel, foreach, data.table, RUnit, XML, bit64, autoencoder, flock, zoo
```


PROJET DATA SCIENCE #1

REFERENCE ANALYTICS

pRoduction, industRialisation, c'est possible ?

En tout cas ce n'est pas impossible; **ce qu'on a fait** :

Tests / Intégration / Outillage / Déploiement :

Script de construction du package R:

```
#!/bin/bash

echo "=== BUILDING R PACKAGE ==="

R -e 'package.skeleton(name="qatsDME", code_files=c("../src/conf_qats_dme.R",
  "../src/processing_pipeline.R",
  "../src/clustering.R",
  ""))'

R CMD build qatsDME
R CMD check qatsDME

echo "=== INSTALLING R PACKAGE LOCALLY ==="

sudo R CMD INSTALL qatsDME
```

PROJET DATA SCIENCE #1

REFERENCE ANALYTICS

pRoduction, industRialisation, c'est possible ?

En tout cas ce n'est pas impossible; **ce qu'on a fait** :

Tests / Intégration / Outillage / Déploiement :

Script de construction du package Debian contenant le package R

```
PKG_BASENAME=qats-dme
PKG_VER=1.5
PKG_REV=2
PKG_SHORTDESC="Expandium Qats-dme"
PKG_LONGDESC="Expandium Qats-dme: data mining engine to process abis etcs data"
PKG_DEPS_Debian="r-base (>= 3.1), r-base-core (>= 3.1), r-base-dev(>= 3.1), \
                gsl-bin (>= 1.16), libgsl0-dev(>= 1.16), libgsl0ldbl(>=1.16),\
                libgsl0-dbg (>= 1.16), libxml2-dev (>= 2.9), libicu-dev (>= 52.1),\
                libcurl4-openssl-dev (>= 7.3)"
```

PROJET DATA SCIENCE #1

REFERENCE ANALYTICS

production, industrialisation, c'est possible ?

En tout cas ce n'est pas impossible; ce qu'on a fait :

Tests / Intégration / Outillage / Déploiement :

Contenu du package Debian contenant le package R

- bin: les exécutables
- Debian: les scripts de configurations (install/remove)
- var/lib... : les dépendances.....

```
qats-dme
├── bin
│   ├── dme_learn_network
│   ├── dme_make_dependencies
│   └── dme_mine_data
├── Debian
│   ├── qats-dme.configure
│   └── qats-dme.remove
├── etc
│   └── conf_qats_dme.R
├── usr
│   └── share
│       ├── doc
│       │   ├── qats-dme
│       │   ├── changelog.gz
│       │   └── copyright
│       └── man
│           └── man1
│               └── dme_learn_network.1.gz
└── var
    ├── lib
    │   └── expand
    │       └── dme
    │           └── src
    │               └── contrib
    │                   ├── autoencoder_1.1.tar.gz
    │                   ├── PACKAGES
    │                   ├── qatsDME_1.1.tar.gz
    │                   ├── Rcpp_0.12.12.tar.gz
    │                   ├── Rcpp_0.12.13.tar.gz
    │                   ├── Rcpp_0.12.3.tar.gz
    │                   ├── Rcpp_0.12.4.tar.gz
    │                   ├── Rcpp_0.12.5.tar.gz
    │                   ├── RcppCCTZ_0.2.3.tar.gz
    │                   ├── RcppRoll_0.2.2.tar.gz
    │                   ├── zoo_1.7-12.tar.gz
    │                   ├── zoo_1.7-13.tar.gz
    │                   └── zoo_1.8-0.tar.gz
```



PROJET DATA SCIENCE #1

REFERENCE ANALYTICS

pRoduction, industRialisation, c'est possible ?

En tout cas ce n'est pas impossible; **ce qu'on a fait** :

Tests / Intégration / Outillage / Déploiement :

Création d'un repo local R :

```
library("miniCRAN")

pkgs <- c(
  "autoencoder",
  "tseries",
  "stringr",
  "...")

pth <- "../lib"

deps <- pkgDep(pkgs, enhances=TRUE)

makeRepo( c(pkgs,deps),
  path=pth,
  download=TRUE)
```

PROJET DATA SCIENCE #1

REFERENCE ANALYTICS

pRoduction, industRialisation, c'est possible ?

En tout cas ce n'est pas impossible; **ce qu'on a fait** :

Tests / Intégration / Outillage / Déploiement :

Création d'un repo local R :

```
library("miniCRAN")

pkgs <- c(
  "autoencoder",
  "tseries",
  "stringr",
  "...")

pth <- "../lib"

deps <- pkgDep(pkgs, enhances=TRUE)

makeRepo( c(pkgs,deps),
  path=pth,
  download=TRUE)
```

Lors de l'installation du package Debian :

```
#!/usr/bin/env Rscript

install.packages("qatsDME",
  repos = "file:///var/lib/qats-dme/expand/dme/",
  type = "source");
```

PROJET DATA SCIENCE #1

REFERENCE ANALYTICS

pRoduction, industRialisation, c'est possible ?

En tout cas ce n'est pas impossible; **ce qu'on a fait** :

Tests / Intégration / Outillage / Déploiement :

- RUnit
- Définition des tests

```
1 test.functionLearning1 <- function(){
2   myData = read.table("...")
3   result = myFunction1(myData)
4   checkEquals(current = result,
5               target = 42,
6               msg = "Checking the return value of myFunction1")
7 }
8
9 test.functionValidation1 <- function(){
10  [...]
11 }
12
13 [...]
14
```

En tout cas ce n'est pas impossible; **ce qu'on a fait :**

- RUnit
- Définition des tests

- Lancement des tests



PROJET DATA SCIENCE #1

REFERENCE ANALYTICS

pRoduction, industRialisation, c'est possible ?

En tout cas ce n'est pas impossible; **ce qu'on a fait** :

Tests / Intégration / Outillage / Déploiement :

- package R + package Debian (+scripts)
 - `dpkg -i qats-dme / apt-get install qats-dme`
- Tests
- ⇒ On peut utiliser **Jenkins** !

“Jenkins is an open source automation server which enables developers around the world to reliably build, test, and deploy their software”

PROJET DATA SCIENCE #1

REFERENCE ANALYTICS

pRoduction, industRialisation, c'est possible ?

En tout cas ce n'est pas impossible; **ce qu'on a fait** :


Tests / Intégration / Outillage / Déploiement :

- package R + package Debian (+scripts)
 - `dpkg -i qats-dme / apt-get install qats-dme`
- Tests
- ⇒ On peut utiliser **Jenkins** !
 - Job de build : lance les scripts de packaging
 - Job de test : installe les packages + les autres briques logicielles + lance les tests

"Jenkins is an open source automation server which enables developers around the world to reliably build, test, and deploy their software"

PROJET DATA SCIENCE #1


REFERENCE ANALYTICS


 **Jenkins**


rechercher


Nicolas Greffard | se déconnecter


Rafraîchissement automatique


 Retour au tableau de bord


 État


 Modifications


 Répertoire de travail


 Lancer un build avec des paramètres


 Supprimer Projet


 Configurer

 Build Graph

 TAP Extended Test Results

 Failure Cause Management

 Failure Scan Options

 Historique des builds

tendance

find

#920

17 nov. 2017 04:19

#919

16 nov. 2017 10:02

#918

16 nov. 2017 09:03

#917

16 nov. 2017 03:59

#916

15 nov. 2017 15:15

#915

15 nov. 2017 03:53

#914

14 nov. 2017 15:16

#913

14 nov. 2017 14:48

#912


14 nov. 2017 04:19


#911

13 nov. 2017 10:36

#910


13 nov. 2017 02:38


 RSS des builds


 RSS des échecs

Projet develop-e2e-dme

This installs the C++, the EDS, and a full featured Quats Railway. It is supposed to run at midnight.

 Espace de travail

 Changements récents

 Derniers résultats des tests (aucune erreur)

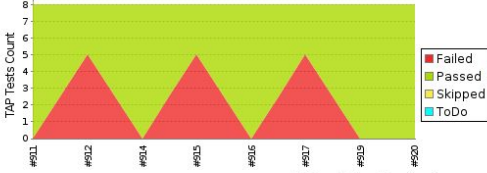
Projets en amont

- develop-00-launch-all-tests
- develop-e2e-00-launch-all-tests-v3


Liens permanents


- Dernier build (#920), il y a 12 h
- Dernier build stable (#920), il y a 12 h
- Dernier build avec succès (#920), il y a 12 h
- Dernier build en échec (#917), il y a 1 j 12 h
- Dernier build non réussi (#918), il y a 1 j 7 h
- Last completed build (#920), il y a 12 h

TAP Tests



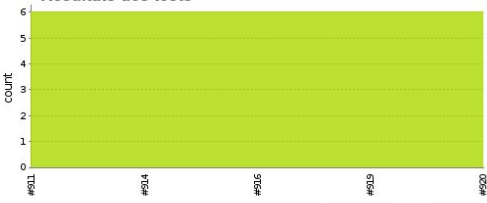
Build	Failed	Passed	Skipped	ToDo
#911	0	8	0	0
#912	5	3	0	0
#914	0	8	0	0
#915	5	3	0	0
#916	0	8	0	0
#917	5	3	0	0
#919	0	8	0	0
#920	0	8	0	0

 Chuck Norris' preferred IDE is hexedit.


 modifier la description


Désactiver le projet

Résultats des tests



Build	Failed	Passed	Skipped	ToDo
#911	0	8	0	0
#912	5	3	0	0
#914	0	8	0	0
#915	5	3	0	0
#916	0	8	0	0
#917	5	3	0	0
#919	0	8	0	0
#920	0	8	0	0

 (montrer les échecs seulement) agrandir



expandium

35

PROJET DATA SCIENCE #1

REFERENCE ANALYTICS

pRoduction, industRialisation, c'est possible ?

En tout cas ce n'est pas impossible; **ce qu'on a fait** :

Tests / Intégration / Outillage / Déploiement :

- Tests d'intégrations lancés par Runit depuis Jenkins
 - Chaîne de traitement donc data in => data out
 - On peut lancer toutes les briques, traiter les données et comparer les résultats à des résultats validés à la main (si match => pas de régression)
 - Les algos n'ont pas une grosse composante stochastique, donc le traitement est reproductible. Dans le cas contraire; possibilité d'appliquer d'autres mesures que la stricte égalité (seuils/intervalles de confiance etc..)

PROJET DATA SCIENCE #1

REFERENCE ANALYTICS

Enseignements

- Packaging Debian & R : **standard** mais **contraignant**
- Obligation de stocker un repo local => **Dependencies HELL**
 - RSQLite: méthode deprecated entre deux livraisons
 - Deps systèmes installées en “sous-marin” (libicu-dev)
 - Package inutilisable car nécessitant une lib système incompatible avec des librairies utilisés par d'autres produits Expandium
- Mais on a du **R en prod** !

PROJET DATA SCIENCE #2

TRAIN CLUSTERING

Besoin

[Confidentiel] : Classification **semi-supervisée** pour reconstruire des trajets de train à partir de données télécom

Beaucoup de **manipulation de données** : R apparaît comme un bon candidat

Méthode

- Classification ascendante hiérarchique + beaucoup de customization

Solution en pratique

- Environ 2000 LOC R, 1 dev
- Architecture similaire à RefAnalytics : data in (TSV) => data out (TSV résultats)

PROJET DATA SCIENCE #2

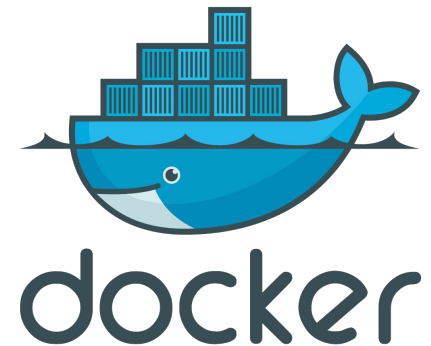
TRAIN CLUSTERING

Déploiement continu

- **Validation** de l'algorithme entre 2 partis (Expandium et notre client)
- Besoin de le mettre à jour très **régulièrement**

Solution : Docker

- *“Docker est un outil qui peut empaqueter une application et ses dépendances dans un conteneur isolé, qui pourra être exécuté sur n'importe quel serveur”*
- On peut voir ça comme une machine virtuelle allégée
 - Facilite le **déploiement continu** (on build l'image et on la déploie)
 - Permet d'encapsuler tout le code R et ses dépendances facilement



PROJET DATA SCIENCE #2

TRAIN CLUSTERING

Docker

- Une image docker est spécifiée par un Dockerfile

```
FROM rocker/r-base

MAINTAINER Nicolas Greffard "nicolas.greffard@expandium.com"
ENV DEBIAN_FRONTEND noninteractive
RUN apt-get update && apt-get install -y strace libxml2-dev

RUN R -e "install.packages(c('data.table', 'dplyr', 'RSQLite', \
                             'stringr', 'bit64', 'XML', 'xml2', 'RUnit'))"

COPY src/* src/
WORKDIR /src
CMD R -f start.R >>/tmp/tmp_tc.log
```

- Construction de l'image: `docker build -t nom_image:TAG` .
- Lancement: `docker run -v /path/to/data/host:/path/to/data nom_image:TAG`

Notes : on peut “versionner” les images dans un repo docker

PROJET DATA SCIENCE #2

TRAIN CLUSTERING

Tests

- On ne lance plus un script installé par un package Debian mais une image Docker
- On peut “injecter” les tests unitaires lors de la construction de l'image :
 - `ADD tests/* & cat “defineTestSuite(…)” >> start.R`
- Dans le job Jenkins de build :
 - On crée une image (minor version impaire eg: 1.3) de test
 - S'ils passent on crée et on publie sur le repo une image finale (paire : 1.4)

PROJET DATA SCIENCE #2

TRAIN CLUSTERING

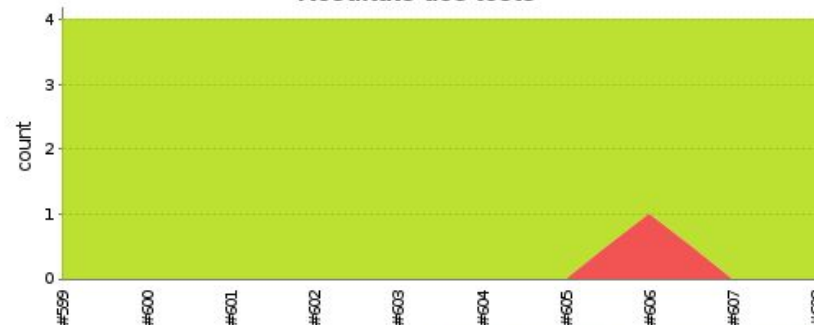
Tests



[Recent Changes](#)

Stage View

Résultats des tests



[\(montrer les échecs seulement\)](#) [agrandir](#)

		cleaning up workspace	generate train-clustering image	process train-clustering image	test train-clustering image	publish train-clustering image
	Average stage times: (Average full run time: ~6min 24s)	354ms	12s	1min 5s	62ms	2s
#608	Jan 12 02:43 No Changes	387ms	13s	1min 2s	92ms	1s
#607	Jan 11 15:36 1 commits	388ms	9s	1min 11s	37ms	5s
#606	Jan 11 11:38 1 commits	380ms	12s	1min 10s	102ms failed	

PROJET DATA SCIENCE #2

TRAIN CLUSTERING

Déploiement

- On installe plus un package mais une image Docker
- Idéalement : orchestrateur type Kubernetes
- A minima : reverse proxy + docker pull nom_image:latest depuis le serveur client
⇒ compliqué à cause des restrictions de l'accès distant VPN
- Bricolage :
 - `docker save docker-registry:4242/mon_image:TAG | bzip2 > img.tar.bz`
 - `rsync / sftp / etc`
 - `cat img.tar.bz | bunzip | docker load`

PROJET DATA SCIENCE #2

TRAIN CLUSTERING

Enseignements

- Docker facilite le déploiement et le “build” du code R : plus **propre** et plus **facile** à modifier / tester
- Dependencies hell \Rightarrow les deps sont incluses dans le Dockerfile
 - Renforce la **reproductibilité** du code / des résultats
 - Permet de développer/tester dans un environnement **ISOprod**
- Problèmes
 - Chaîne de traitement à partir de données TSVs \Rightarrow ne facilite pas l'interopérabilité et l'interactivité du code R
 - R pêche dans l'aspect Tuyauterie/**plomberie** autour du code d'analyse
 - Services / Daemons
 - APIs, accès systèmes, accès concurrentiels

PROJET DATA SCIENCE #3

SIMULATEUR GSM-R

Besoin

Etre capable de prédire le niveau de champ (qualité du signal radio) d'un mobile / train en fonction de variables dépendantes hétérogènes (antenne, type antenne, météo, géolocalisation du train, congestion réseau, congestion ferroviaire, etc..)

⇒ Problème de **régression**

Méthode

- Deep learning / GBMs / Random Forests

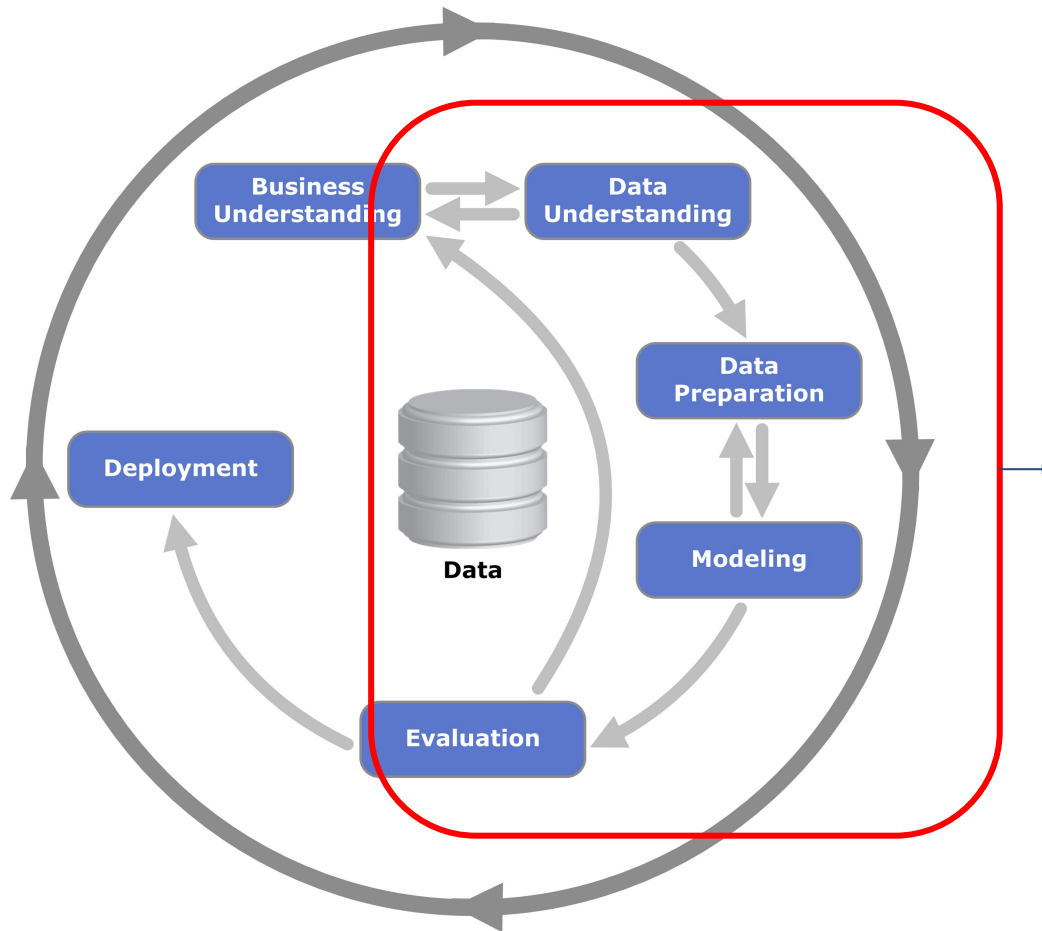
Solution en pratique

- Environ 1300 LOC R, 1 dev (puis 2)

PROJET DATA SCIENCE #3

SIMULATEUR GSM-R

Cycle de vie



- Data preparation / Modeling / Evaluation sous Rstudio
- Preprocessing: `preproc.R`
- Training: `params_grid_search.R`
- Testing: `predict.R`

PROJET DATA SCIENCE #3

SIMULATEUR GSM-R

Comment rendre le modèle utilisable par l'extérieur ?

En parlant de plomberie :

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

⇒ Ou comment transformer son code R en **API** http Rest très simplement

```
## @post /predict
predict <- function(dat = NULL){
  if(is.null(dat)){
    stop("Need some data !")
  }
  dat <- preProcessing(dat)
  d_frame <- as.h2o(dat, destination_frame = "api_frame")

  pred <- as.data.frame(h2o.predict(model_dl = getNewestModel(), d_frame))
  pred$predict <- floor(pred$predict)

  return(pred)
}
```

Décorations `## @get` ou `## @post/route/<params>`

Les arguments de la fonction `predict(dat)` viennent directement de la requête http (soit dans son body soit via les sub-routes)

PROJET DATA SCIENCE #3

SIMULATEUR GSM-R

Comment rendre le modèle utilisable par l'extérieur ?

En parlant de plomberie :

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

⇒ Ou comment transformer son code R en **API** http Rest très simplement

```
## @post /predict
predict <- function(dat = NULL){
  if(is.null(dat)){
    stop("Need some data !")
  }
  dat <- preProcessing(dat)
  d_frame <- as.h2o(dat, destination_frame = "api_frame")

  pred <- as.data.frame(h2o.predict(model_dl = getNewestModel(), d_frame))
  pred$predict <- floor(pred$predict)

  return(pred)
}
```

```
library("plumber")
app <- plumb("api.R")
app$run(host="0.0.0.0", port = 42)
```

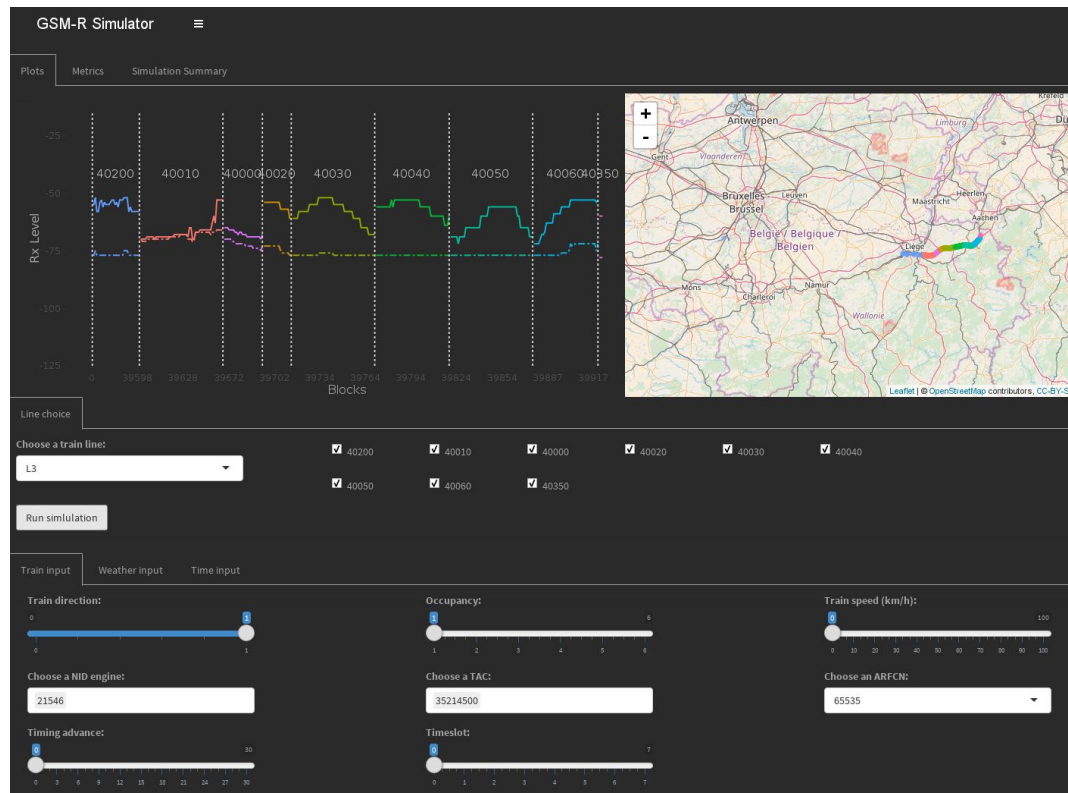
plumber va automatiquement parser le fichier api.R et exposer les fonctions décorées

PROJET DATA SCIENCE #3

SIMULATEUR GSM-R

Interface prototype

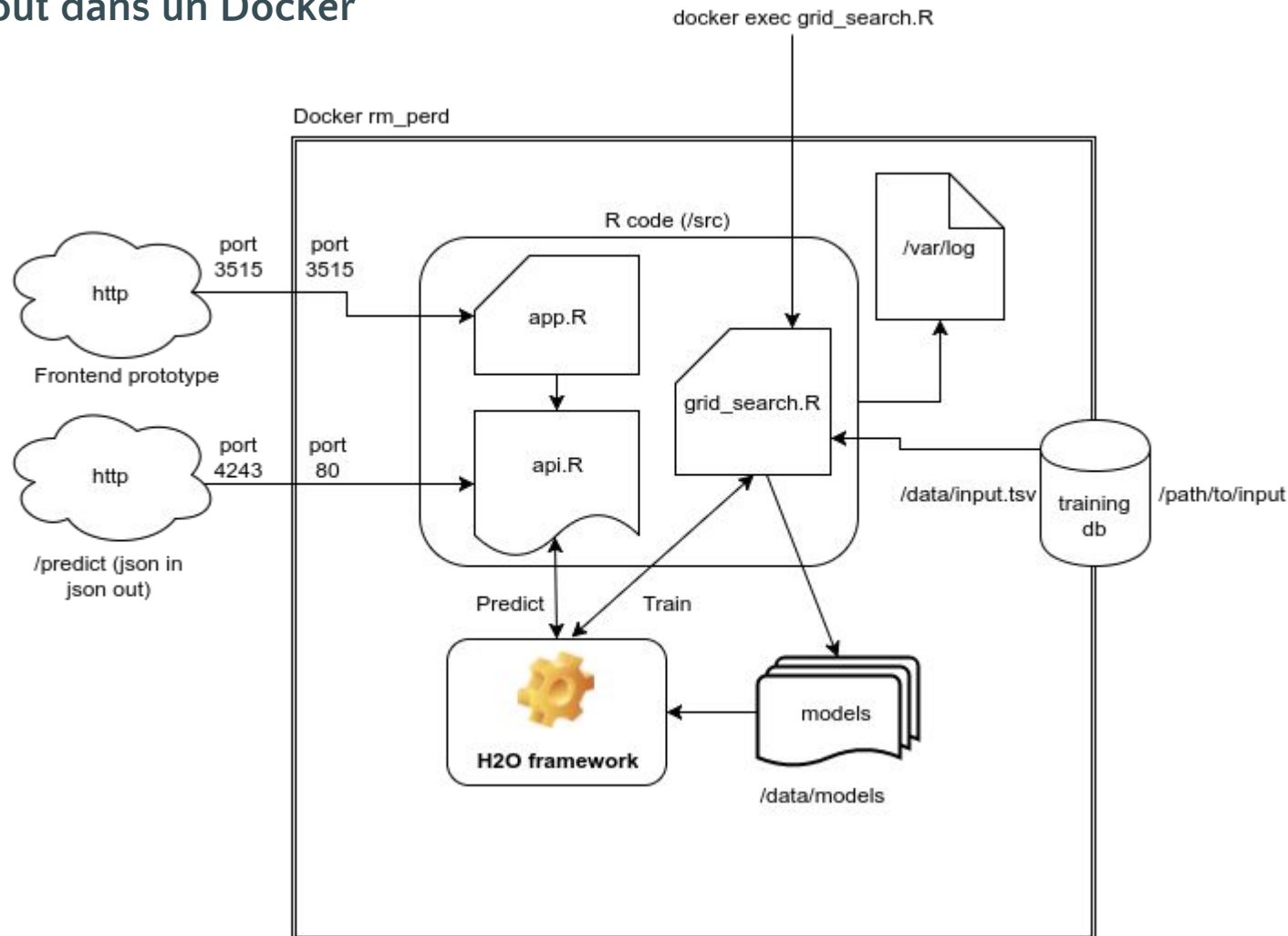
La logique de training/testing (predictions) étant en R: on peut directement intégrer d'autres composants R, eg : Shiny/ShinyDashboard.



PROJET DATA SCIENCE #3

SIMULATEUR GSM-R

Le tout dans un Docker



PROJET DATA SCIENCE #3

SIMULATEUR GSM-R

Le tout dans un Docker

Deps
Système

Deps
R

Ports
exposés

Sources &
modèles

Config &
lancement
supervisord

```
FROM rocker/r-base

MAINTAINER Nicolas Greffard "nicolas.greffard@expandium.com"
ENV DEBIAN_FRONTEND noninteractive

RUN apt-get -y -q update && \
    apt-get -y install \
        libcurl3 \
        libssl-dev \
        libcurl4-openssl-dev \
        libicu-dev \
        libxml2-dev \
        sudo \
        nano \
        openjdk-8-jre \
        supervisor

RUN R -e "install.packages(c('h2o', \
    'plumber', 'stringr', 'rvest', 'weatherData', 'shiny', \
    'ggplot2', 'plotly', 'rwwwunderground', 'rvest'))"

EXPOSE 80
EXPOSE 3515

COPY src/* /src/
COPY data/models /data/models
COPY data/input.tsv /data/
COPY data/meas_sncb.dat /data/

WORKDIR /src

COPY supervisord.conf /etc/

CMD /usr/bin/supervisord
```

PROJET DATA SCIENCE #3

SIMULATEUR GSM-R

R code as a service

- Supervisor : programme Python permettant de lancer d'autres commandes/programmes et de les maintenir online (ie: restart si crash etc..)

```
[supervisord]
nodaemon=true
```

```
[program:rmp_frontend]
command=R -e "source('app.R')"
autorestart = true
stdout_logfile=/var/log/rmp_frontend
stderr_logfile=/var/log/rmp_frontend
```

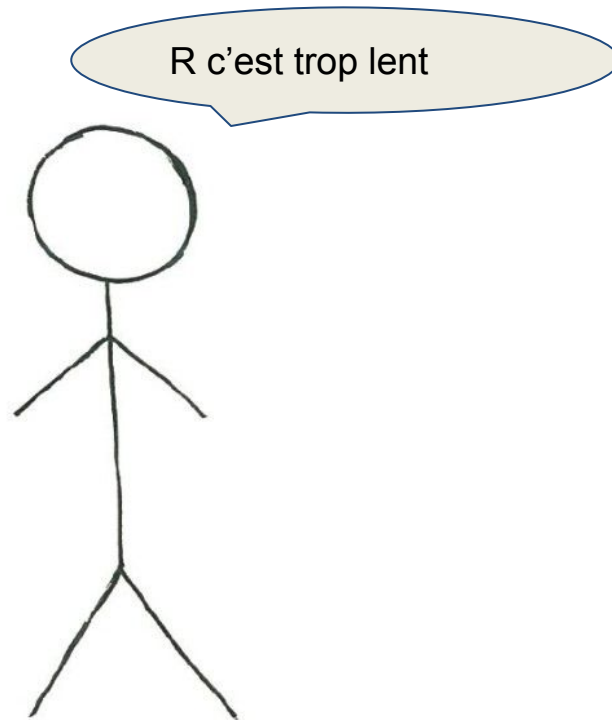
Code Shiny

```
[program:rmp_api]
command=R -e "source('start_server.R')"
autorestart = true
stdout_logfile=/var/log/rmp_api
stderr_logfile=/var/log/rmp_api
```

Code Plumber
app\$run(plumb("api.R"))

PROJET DATA SCIENCE #3

SIMULATEUR GSM-R



PROJET DATA SCIENCE #3

SIMULATEUR GSM-R

Scaling

- h2o : **backend** d'apprentissage qui tourne sur un cluster **hadoop** (ou en standalone sur une machine)
- Pour la partie R (API HTTP par exemple) on peut scale up de la même manière en utilisant les outils de l'univers Docker
 - Docker-compose
 - Load balancers

PROJET DATA SCIENCE #3

SIMULATEUR GSM-R

Scaling

- h2o : **backend** d'apprentissage qui tourne sur un cluster **hadoop** (ou en standalone sur une machine)
- Pour la partie R (API HTTP par exemple) on peut scale up de la même manière en utilisant les outils de l'univers Docker
 - Compose file

```
version: '2'
services:
  rm_pred:
    build: .
  lb:
    image: 'dockercloud/haproxy:1.2.1'
    links:
      - rm_pred
    environment:
      - TIMEOUT=connect 10000, client 500000, server 500000
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
    ports:
      - 4242:80
```

Notre image de simulateur

Une image de load balancing

PROJET DATA SCIENCE #3

SIMULATEUR GSM-R

Scaling

- h2o : **backend** d'apprentissage qui tourne sur un cluster **hadoop** (ou en standalone sur une machine)
- Pour la partie R (API HTTP par exemple) on peut scale up de la même manière en utilisant les outils de l'univers Docker
 - Docker-compose
 - Load balancers

`docker-compose build`

`docker-compose up`

`docker-compose scale rm_pred=5`

PROJET DATA SCIENCE #3

SIMULATEUR GSM-R

Monitoring

- Autant tirer parti de l'existant
 - Chez expandium : Zabbix et ELK (Kibana)
 - Zabbix : export de résultats/métriques dans une base Sqlite lue par Zabbix
 - Kibana (ELK) : Fonction de LOG home made qui peut output du human readable ou du json pour être parsé par logstash

PROJET DATA SCIENCE #3

SIMULATEUR GSM-R

Enseignements et conclusion

- Les nouvelles technologies autour des containers nous offrent des outils qui viennent combler les faiblesses de R
- Elles nous aident à faire du R dans un environnement technique type micro-service
- Dans ce contexte, Plumber permet de passer très rapidement d'un POC à un produit
- De plus en plus de frameworks "high performance" proposent des interfaces R
 - <https://tensorflow.rstudio.com/>
 - <http://docs.h2o.ai/>
 - <https://azure.microsoft.com/en-us/blog/doazureparallel/>
- Pour tout le reste, il y a Rcpp ;)

RÉFÉRENCES

http://blog.sellorm.com/files/R_is_Production-Safe.pdf

Allianz • What: Insurance Claim scoring • How: API's in R • Using: Rserve, Java

Worldpay • What: Call Centre CRM tool and x-sell pipeline • How: Web app written in R • Using: Shiny

ONS • What: Migrate from SAS • How: Migrate SAS based services to R • Using: Plumber & command line

Hedge Fund • What: Batch based scoring system • How: Sophisticated command line application • Using: R and various databases

Merci

Ps: si ces problématiques vous intéressent et que vous voulez faire parler votre R-FU et vos skills data science, Expandium Recrute

Pour plus d'information : www.expandium.com

