A cloud-based pipeline for analysis of FHIR and long-read data

Tim Dunn University of Michigan **Erdal Cosgun** Microsoft Biomedical Platforms and Genomics

Introduction

As genome sequencing becomes cheaper and more accurate, it is becoming increasingly viable to merge this data with electronic health information to inform clinical decisions. In this work, we demonstrate a full pipeline for working with PacBio sequencing data and clinical FHIR data, from initial data to tertiary analysis. The electronic health records are stored in FHIR – Fast Healthcare Interoperability Resource – format, the current leading standard for health care data exchange. For the genomic data, we perform variant calling on long read PacBio HiFi data using Cromwell on Azure. Both data formats are parsed, processed, and merged in a single scalable pipeline which securely performs tertiary analyses using cloud-based Jupyter notebooks. We include three example applications: exporting patient information to a database, clustering patients, and performing a simple pharmacogenomic study.

```
##fileformat=VCFv4.3
##fileDate=20210115
##source=VariantCallerScript
##reference=file:///references/HG38.fasta
##contig=<ID=20,length=62435964,assembly=B36,species="Homo sapiens">
##phasing=partial
##INFO=<ID=DP,Number=1,Type=Integer,Description="Total Depth">
##INFO=<ID=AF,Number=A,Type=Float,Description="Allele Frequency">
##INFO=<ID=AA,Number=1,Type=String,Description="Ancestral Allele">
##INFO=<ID=DB,Number=0,Type=Flag,Description="dbSNP membership, build 129">
##FILTER=<ID=q10,Description="Quality below 10">
##FILTER=<ID=s50,Description="Less than 50% of samples have data">
##FORMAT=<ID=GT,Number=1,Type=String,Description="Genotype">
##FORMAT=<ID=DP,Number=1,Type=Integer,Description="Read Depth">
##FORMAT=<ID=HQ,Number=2,Type=Integer,Description="Haplotype Quality">
#CHROM POS     ID  REF ALT    QUAL FILTER INFO                              FORMAT      NA00001        NA00002
20    14370   rs6 G   A      29   PASS   DP=14;DB;                          GT:DP:HQ 0|0:1:51,51    1|0:8:51,51
20    17330   .   T   A      3    q10    DP=11;AF=0.017 GT:DP:HQ 0|0:3:58,50 0|1:5:65,3
20    1110696 rs5 A   G,T    67   PASS   DP=10;AA=T;DB   GT:DP:HQ 1|2:6:23,27 2|1:0:18,2
20    1230237 .   T   .      47   PASS   DP=13;AA=T      GT:DP:HQ 0|0:7:56,60 0|0:4:51,51
20    1234567 ms1 GTC G,GTCT 50   PASS   DP=9;AA=G       GT:DP    0/1:4       0/2:2
```

Fig. 1: Example VCF file, including the file header and variant call data.

```
'resource': {'resourceType': 'Patient',
   'id': 'b5f1da11-3826-4821-bb84-dd72294c9a4c',
   'meta': {'versionId': '1',
    'lastUpdated': '2022-07-13T19:23:23.981+00:00',
    'profile': ['http://hl7.org/fhir/us/core/StructureDefinition/us-core-patient']},
   'text': {'status': 'generated',
    'div': '<div xmlns="http://www.w3.org/1999/xhtml">Generated by Synthea</div>'},
   'extension': [{'extension': [{'url': 'ombCategory',
       'valueCoding': {'system': 'urn:oid:2.16.840.1.113883.6.238',
        'code': '2106-3',
        'display': 'White'}},
      {'url': 'text', 'valueString': 'White'}],
     'url': 'http://hl7.org/fhir/us/core/StructureDefinition/us-core-race'},
    {'extension': [{'url': 'ombCategory',
       'valueCoding': {'system': 'urn:oid:2.16.840.1.113883.6.238',
        'code': '2135-2',
        'display': 'Hispanic or Latino'}},
      {'url': 'text', 'valueString': 'Hispanic or Latino'}],
     'url': 'http://hl7.org/fhir/us/core/StructureDefinition/us-core-ethnicity'},
    {'url': 'http://hl7.org/fhir/StructureDefinition/patient-mothersMaidenName',
     'valueString': 'Julia241 Luna60'},
```

Fig. 2: Example synthetic FHIR data, generated using Synthea [1].

Overview

Our cloud-based pipeline contains the following steps:

1. **Variant Calling:** We first run a scalable cloud-based variant calling pipeline using "Cromwell on Azure" [2].
2. **FHIR Data Generation:** Synthetic FHIR patient data is generated using Synthea [1].
3. **Deploy FHIR Server:** A FHIR server is deployed to securely host patient data which can be accessed using an Azure API.
4. **Upload FHIR Data:** A script is run to upload all synthetic generated patient data to the FHIR server.
5. **Deploy Sync Agent to Download FHIR Data:** The "Sync Agent" [3] function app downloads data from the FHIR server and stores it in Parquet format.
6. **Extract FHIR Data:** A script extracts the desired clinical information from the Parquet files into a data frame.
7. **Merge VCF Files:** Variant calls for all patients are merged into a single file for efficient processing.
8. **Parse VCF Data:** A script extracts the desired variant call information from the merged VCF file into a data frame.
9. **Merge FHIR and VCF Data:** The data frames containing patient clinical and genomic data are merged into a single data frame.
10. **Analysis:** Several potential downstream applications are demonstrated.

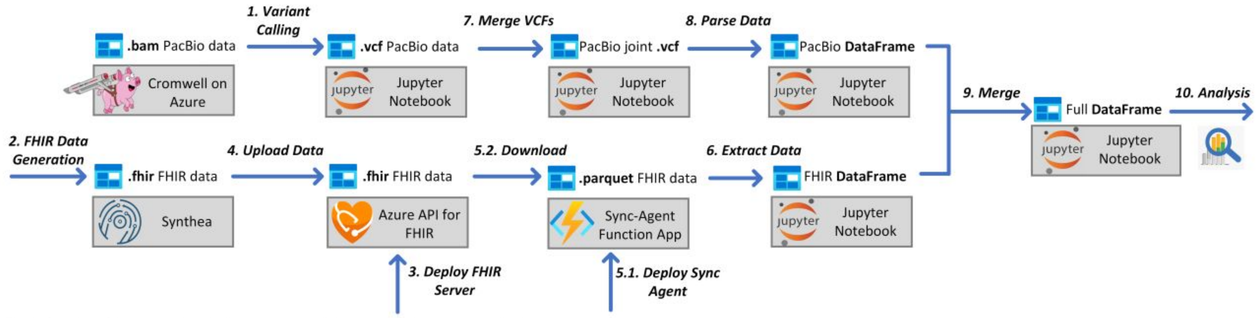

Fig. 3: Overview of data processing pipeline, common to all demonstrated downstream applications.

Methodology

Cromwell is a workflow management system for running genomics analysis scripts at various scales, from a local machine or computing cluster to larger cloud instances. Cromwell was originally developed by the Broad Institute, and is used in the Genome Analysis Toolkit's (GATK) recommended "Best Practices" pipeline for genome analysis. "Cromwell on Azure" is a project developed by Microsoft that configures all the Azure resources necessary to run Cromwell workflows on the Azure cloud. The first stage of our pipeline uses Cromwell on Azure to perform variant calling on long read PacBio HiFi reads. Although this whole genome sequencing pipeline also calls structural variants, at the moment only small germline variants are used in downstream analyses. An overview of how Cromwell on Azure works is shown below, in Figure 4.

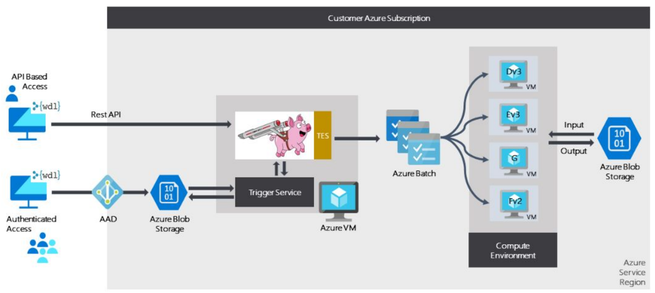

Fig. 4: High-level overview of Microsoft's "Cromwell on Azure" service, used to perform variant calling [2].

In order to ensure that all data processing is secure, our pipeline works within a Jupyter notebook hosted on an Azure "Confidential Compute" virtual machine. Results are then made available on a local machine using SSH tunneling. This setup can be seen below, in Figure 6. Confidential virtual machines have security features such as secure boot, a virtual trusted platform module (vTPM), boot integrity monitoring, and virtualization-based security. Together, these features protect against persistent or advanced threats such as rootkits. Virtualization provides further security by isolating memory address spaces to remove any possibility of memory cross-contamination.

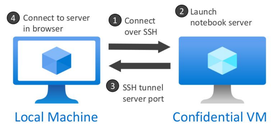

Fig. 6: Confidential computing configuration.

Ensuring the confidentiality of patient health information is paramount. In order to securely upload FHIR data to a server and later query this data, we must have some means of performing authentication. Azure Active Directory is a cloud-based identity and access management service which we selected to use for authentication purposes. After deploying a FHIR server, we added permissions for our notebook script to write data to the server, registering it with Azure Active Directory and providing it with a secret key (generated by Azure AD) as proof-of-identity. In order to upload data to the FHIR server, the notebook requests a secret token from Azure AD using its secret key, and then uses that token to POST resources to the FHIR server.

Fig. 7: Secure deployment process for an "Azure API for FHIR" server.

Synthea is a widely-used open source tool for generating realistic (but synthetic) patient data in FHIR format [1]. This enables researchers to work with realistic clinical datasets without worrying about any of the legal, ethical, or security concerns that would accompany working with real patient data. Synthea works by first using general census demographic data in combination with user-specified configuration information to generate a synthetic world population. Once this population has been generated, disease incidence and prevalence statistics are used in tandem with disease-specific models to simulate patients contracting a given disease and all subsequent interactions with the healthcare system that will be stored in clinical records. This information can then be exported in FHIR format.

In addition to Synthea's default disease modules, for our third example pipeline application (see "Results") we created a custom Synthea module to model a simplified breast cancer patient cohort. A graphical depiction of this module is shown below, in Figure 5. For this cohort, patients had a default survival rate of 50%. Based on the combination of their genetics (the presence of a particular variant) and their prescribed medication (Epirubicin or Doxorubicin), their survival rate either remained the same or increased to 90%. This models the fact that medication effectiveness can depend upon the genetics of the patient involved.

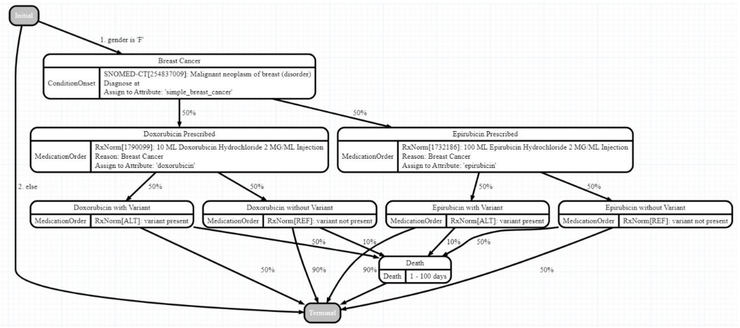

Fig. 5: Custom Synthea module which models breast cancer patients' survival rates as a function of their medications and genetics.

In order to perform data science applications with FHIR data, we first need to convert the hierarchical JSON data into tabular format. An existing open source tool called the "FHIR to Synapse Sync Agent" solves this problem by downloading FHIR data from the server and converting it to Parquet format. Figure 8 shows an overview of how the Sync Agent works. Parquet files store the same information as ordinary CSV ("comma-separated values") or TSV ("tab-separated values") files, but in a more efficient compressed manner. Data is stored in column-major order, which allows compression algorithms such as run length encoding, dictionary encoding, or delta encoding to be applied per column depending upon each column's format and values.

Once downloaded, we parse the Parquet FHIR data to retrieve relevant information and load it into a data frame. Although Parquet is already a tabular format, the converted data contains extraneous information which must be discarded. Furthermore, FHIR data pertaining to a single patient is stored in multiple resource types. These files (such as Patient, Medication, or Condition resources) must be parsed separately and the records associated with one another using the patient's unique Medical Record Number (MRN). In the end, each row stores data for a single patient, and the numerous columns contain all desired information about that patient, extracted from the Parquet files. At this point the FHIR and PacBio data can be merged together, as shown in Figure 9.

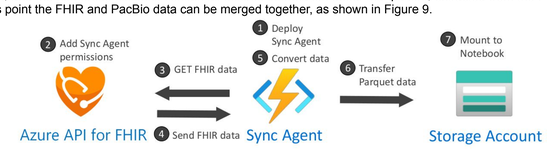

Fig. 8: Deployment process for a "Sync Agent" which downloads data from the FHIR server [3].

Results

Application #1 – Data Exportation: The first and simplest application we demonstrated was exporting the merged FHIR and PacBio data to a database for further analyses. We selected Azure Synapse Analytics as our database platform because it ensures scalability and reliability.

	city	state	country	gender	dead	age	newest_med_code	...	49:GT	49:AF_0	49:AF_1	49:PL_0/0	49:PL_0/1	49:PL_1/1	49:DP
0	Boston	MA	US	female	False	22.009037	748856	...	0/0	1	0	0	8	8	7
1	Amesbury	MA	US	male	False	64.588092	310798	...	0/0	1	0	0	3	3	7
2	Boston	MA	US	female	False	17.316361	1367439	...	0/1	0.285714	0.714286	29	0	23	7
3	Yarmouth	MA	US	male	True	69.918688	896209	...	0/0	1	0	0	4	4	5
4	Attleboro	MA	US	female	False	25.182206	751905	...	0/0	1	0	0	9	9	15

Fig. 9: Final data frame of merged clinical FHIR data and PacBio variant calling data.

Application #2 – Patient Clustering: The second application we demonstrated was an exploratory clustering of our patient dataset using three different clustering methods: K-Means++, DBSCAN, and Spectral Clustering. There is no ground truth available, so this is only useful for discovering similarities between patients.

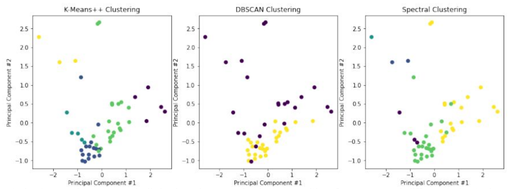

Fig. 10: Patient clusters using merged clinical and genomic data, resulting from several different clustering algorithms.

Application #3 – Pharmacogenomic Study: The final application we demonstrated was a basic pharmacogenomic study to determine the effect of two different medication treatments on breast cancer patients' survival rates. First, a custom Synthea module was designed which modelled a patient population where outcomes were only improved beyond the base 50% survival rate for patients who were prescribed Epirubicin and had a particular genetic variant, and for patients who were prescribed Doxorubicin and did not have the genetic variant. This model is depicted in Figure 5, and uses both clinical and genomic data. Below, Figure 11 shows a breakdown of patient variant presence, treatment type, and survival.

Fig. 11: Sankey diagram showing distribution of patient treatments and outcomes.

More Information

References:

[1] Jason Walonoski et al. "Synthea: An approach, method, and software mechanism for generating synthetic patients and the synthetic electronic health care record." Journal of the American Medical Informatics Association, 25(3):230–238, 2018.
[2] Microsoft. Cromwell on Azure. https://github.com/microsoft/CromwellOnAzure.
[3] Microsoft. FHIR to Synapse Sync Agent. https://github.com/microsoft/FHIR-Analytics-Pipelines.

Acknowledgements: The authors would like to thank Venkat Malladi and Olesya Melnichenko for their support throughout this project.
.
Funding: Microsoft's Biomedical Platforms and Genomics Team supported this work by hosting Tim Dunn through the 2022 Microsoft Research Intern Program. Tim Dunn was also partially supported by the NSF under NSF Graduate Research Fellowship 1841052.

Github Code: https://github.com/microsoft/genomicsnotebook