Sequence Analysis

htsget: a protocol for securely streaming genomic data

Jerome Kelleher1, Mike Lin2, CH Albach3, Ewan Birney4, Robert Davies5, Marina Gourtovaia5, David Glazer5, Cristina Y. Gonzalez6, Andrew Kemp6, John Marshall5, Andrew Nowak5, Alexander Senf4, Jaime M. Tovar-Corona5, Alexander Vikhorev4, Thomas M. Keane4* and GA4GH Streaming Task Team8

1Big Data Institute, Li Ka Shing Centre for Health Information and Discovery, University of Oxford, Oxford, UK, 2DNAneXus Inc, 3Verily Inc, 4European Bioinformatics Institute, Wellcome Genome Campus, Hinxton, CB10 1SD, UK 5Wellcome Sanger Institute, Wellcome Genome Campus, Hinxton, CB10 1SA, UK 6Google Inc, 7Wolfson Wohl Cancer Research Centre, Institute of Cancer Sciences, University of Glasgow, Glasgow G61 1QH, UK 8Global Alliance for Genomics and Health (ga4gh.org)

Joint first authors *To whom correspondence should be addressed.

Abstract
Summary: Standardised interfaces for efficiently accessing high-throughput sequencing data are a fundamental requirement for large-scale genomic data sharing. We have developed htsget, a protocol for secure, efficient and reliable access to sequencing read and variation data. We demonstrate four independent client and server implementations, and the results of a comprehensive interoperability demonstration.

Availability and implementation: http://samtools.github.io/hts-specs/htsget.html
Contact: tk2@ebi.ac.uk
Supplementary information: Supplementary data are available at Bioinformatics online.

1 Introduction
We are witnessing a transition of sequencing technologies from being primarily a research platform to a routine clinical healthcare assay. In the next five years, it is predicted that the vast majority of the human genomes sequenced will be from healthcare (Birney, Vamathevan, and Goodhand 2017; https://doi.org/10.1101/203554). The sheer volume of human genetic data offers new opportunities for the discovery of new disease associations for common and rare disease. Most human genetic data is subject to participant consent agreements which define how the data can be shared. In research, The European Genome-phenome Archive (EGA), The database of Genotypes and Phenotypes (dbGaP), and the Japanese Genome-phenome Archive (JGA) have provided centralised repositories for sharing controlled access human data. As consented human data is increasingly subject to jurisdictional restrictions of data storage and rapidly increasing in scale, we are moving away from a few centralised repositories to a more distributed network of nodes (e.g. biobanks, national or regional healthcare providers, commercial clouds) housing large cohorts. In 2013, the Global Alliance for Genomics and Health was formed as a policy-framing and technical standards setting organisation, akin to the W3C, to support the development of interoperable standards for discovering, sharing, accessing, and storing genomic data (Birney, Vamathevan, and Goodhand 2017).

Standardisation of next-generation sequencing file formats such as SAM/BAM for read data (H. Li et al. 2009) and VCF/BCF for variants (Danecek et al. 2011) was one of the key technical achievements from the 1000 Genomes Project. This has enabled a global ecosystem of interoperable sequence analysis tools, and production scale pipelines to be built around these formats. Sending genomic data over networks has largely been achieved by transporting files in these well-established formats using network protocols such as FTP/HTTP(s), Globus (Allicock et al.; https://doi.org/10.1109/SC.2005.72), and proprietary protocols such as Aspera (http://asperasoft.com/). Commonly used tools such as Samtools (H. Li et al. 2009), GATK (McKenna et al. 2010), bedtools (Quinlan and Hall 2010), and IGV (Robinson et al. 2011) all support obtaining data over FTP/HTTP(s). Utilising open and widely implemented transfer protocols has ensured easy integration with common networking software libraries, reduced dependency on proprietary software or frameworks, and interoperability across network infrastructures.

This file-centric approach to defining and delivering data has substantial drawbacks, however, and there is an increasing consensus that more fine-grained and flexible data access methods are needed. Several such methods of storing and serving genomic data, such as GEMINI (Paila et al. 2013), BGT (Heng Li 2016), GenAp (Kozanitis and Patterson 2016), and Hail (Ganna et al. 2016) have been developed. To avoid fragmentation and to promote a diverse ecosystem of interoperable storage and presentation technologies for genomic data, we need to develop standardised APIs for the discovery and delivery of read and variant data.

In this paper, we introduce a new open standard for real-time secure streaming of genomic data. This protocol, htsget, allows a client to retrieve data overlapping a specific genomic interval and uses existing community standards such as SAM/BAM/CRAM/VCF/BCF as the on-the-wire format. The protocol operates by a simple indirection layer, which provides many benefits over direct HTTP/FTP access to the data by clients. Secure access to data is supported by the industry-standard OAuth 2.0 protocol (https://tools.ietf.org/html/rfc6749).

© The Author(s) 2018. Published by Oxford University Press.
This is an Open Access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted reuse, distribution, and reproduction in any medium, provided the original work is properly cited.
The fundamental goal of htsget is to introduce a standardised interface for requesting and delivering genomic data that is not bound by file semantics. The protocol does not attempt to provide an end-to-end solution for managing genomic data. Issues around the organisation of metadata and data discovery are outside the scope of this protocol. Future work will address these important issues; we envisage htsget as part of a family of loosely coupled protocols enabling efficient and secure discovery and exchange of genomic data.

2 Results

2.1 Schematic of protocol
The key mechanic of the protocol is that the client provides a URL (determined via another discovery service), a preferred format, and an optional genomic range via a HTTP(s) GET request (Figure 1). The server returns a small JSON block with a list of URLs. The client downloads the data from the URLs, concatenates the downloaded data in the order provided by the server to produce the full result of their query.

![Schematic of htsget protocol](image)

**Figure 1:** Schematic of htsget protocol.

2.2 Security
We envision that htsget will be used to stream both open genetic data and data that is subject to increased security and authorisation requirements (e.g. consented human data). Sensitive information transmitted on public networks, such as access tokens and human genomic data, must be protected using Transport Level Security (TLS). Htsget can be integrated into existing authorisation and authentication infrastructures (AAI) that use OAuth2.0 tokens to authorise data requests.

2.2 Initial implementations and interoperability
Htsget has been implemented by a set of research and commercial providers of human data, and community tool maintainers to develop a diverse set of client and server implementations. These include resources such as the European Genome-phenome Archive (EGA), Google Cloud Platform, SAMtools/HTSlib, DNAnexus, and read browsers such as Integrated Genome Viewer (IGV) and Biodalliance (see Supplementary material for full details).

To demonstrate interoperability, each server loaded the 1000 Genomes/Hapmap CEU trio in both BAM (NCBI37) and CRAM format (GRCh38), as well as RNASeq and ChIP-seq data. We developed a test application that runs a mixture of random and edge-case queries for a given client/server combination, checking the integrity of the returned data against a local file. We then used this application to run an extensive suite of tests on a total of 25 different client-server combinations. Full details are given in the supplementary material.

3 Discussion

Files have long been the fundamental unit of bioinformatics, providing a simple abstraction for data organisation and mapping naturally to Unix workflows. However, the explosive growth in the volume of genomic data and the increasingly distributed nature of computing resources are making this classical file-centric approach unsustainable. From a user perspective (i.e. the data consumer), downloading many terabytes of data simply to access a small subset of interest is clearly not satisfactory. Direct access to genomic loci in read or variant data over HTTP is a major improvement, but this is achieved by the means of auxiliary index files and requires that the server expose file semantics for all the data that it serves. This severely limits the flexibility of server providers in how data may be dynamically generated and organised, and also may require that several copies of near-identical data must be stored (if, for example, they wish to support different formats).

We have introduced htsget, an API for requesting and delivering genomic data. It builds on existing well-supported file formats for transport, while giving service providers freedom to use alternative internal data storage models. Using existing formats means that tools and data processing pipelines are only required to make minimal modifications to support htsget, exemplified by the list of initial implementations. There is a single layer of indirection which enables parallelism and robustness for service providers and clients. Htsget is intended to transport both open and controlled access data, appropriate security (TLS, HTTPS, and OAuth2.0) are essential aspects of the specification.

4 Acknowledgements
Htsget was the product of the GA4GH Streaming API task team. We acknowledge the support of the GA4GH secretariat. We acknowledge advice and input from Dixie Baker, Vadim Zalzman, Angel Pizarro, Richard Durbin, Mark Diekhans, Edmon Begoli, Ila Tulchinsky, Heng Li, Rishi Nag, Stephen Keenan, Ilkka Lappalainen, and Jim Robinson.

5 Funding
This work was supported by the Wellcome Trust [100956/Z/13/Z] [201535/Z/16/Z].

6 References


