CSCI 385 - First Deliverable

Kris Selvidge

10/8/2020

Introduction

The domain of this project is network data and how we can use R to visualize performance metrics. For this first deliverable I am looking specifically at HTTP traffic to public websites. Future explorations will include other protocols, routing and other network performance experiments.

I grew up using computers that were not on networks (Commodore 64, IBM Tandy) and watching the rise of computer network has fascinated me. Not only is the network telecommunications industry valued at nearly $3 trillion dollars, almost every industry includes some form of information system dependent on network technology.

I am currently taking EECE 446 Networks course at CSU Chico and in one of our assignments we created an application in C running on Ubuntu 18. The application opens a socket connection to a host, sends a request for the root file from the host and then reads the results returned from the host. Files returned to the socket are read by the source application in a series of chunks of data in bytes. The performance is tested for chunks between size 4 and 1000.

A brief understanding of how my experiment was setup is indicated in the data set section. Background information on wikipedia and Beej's Guide to Networking are also good references.

Data Set

The dataset is the output of a project I programmed for a Networking course at CSU Chico. I collected the data as a requirement for my Networking class. Once I saw the potential for what the application could do I expanded on it to include more fields for data collection.

The data is artificially designed through my own amateur programming. It is a very limited capture of traffic created by a C algorithm. There may be flaws in my programming so this is a known limitation on the data presented. The application is currently only running on my local machine so the results are dependent on my computers performance and location in relation to the destinations collected from.

```
netdat <- read_csv('D:\\fall2020\\TTH\\CSCI385 Data Science\\data1\\d4.csv')
```

```
## Parsed with column specification:
## cols(
##   source = col_character(),
##   destination = col_character(),
##   protocol = col_character(),
##   chunkSize = col_double(),
##   tCount = col_double(),
##   fileSize = col_double(),
```

```
##   retransmits = col_double(),
##   connectTime = col_double(),
##   requestTime = col_double(),
##   receiveTime = col_double()
## )
```

```
head(netdat)
```

```
## # A tibble: 6 x 10
##   source destination protocol chunkSize tCount fileSize retransmits connectTime
##   <chr>  <chr>       <chr>        <dbl>  <dbl>    <dbl>       <dbl>       <dbl>
## 1 192.1~ www.google~ HTTP/1.0         4    199    48408           3    0.000588
## 2 192.1~ www.google~ HTTP/1.1         4    207    48611           4    0.000474
## 3 192.1~ www.google~ HTTP/1.0         5    218    48407           2    0.000411
## 4 192.1~ www.google~ HTTP/1.1         5    216    48609           2    0.000445
## 5 192.1~ www.google~ HTTP/1.0         6    216    48432           6    0.00041
## 6 192.1~ www.google~ HTTP/1.1         6    228    48672           9    0.000414
## # ... with 2 more variables: requestTime <dbl>, receiveTime <dbl>
```

Categorical Variables:

(Note all categorical variables were setup by the experiement. Observational variables are listed below in the continuos variable section.)

- 'source' - 'character' - Origin location requesting root file from destination. Note: In this phase of the experiement there is only one location (localhost) requesting data. This computer is my local HP Omen 17 laptop running Ubuntu 18.
- 'destination' - 'character' - Web server sending root file. These hosts are ten top and popular websites.
- 'protocol' - 'character' - Transfer protocol version HTTP (Hypertext Transfer Protocol) 1.0 or 1.1. Note that twitter.com does not respond to HTTP 1.1 requests.

Continuous Variables

- 'chunkSize' - 'double' - Size of each chunk of packet data in bytes. This value is the only continuous variable adjusted during the experiment. Values range from 4 to 1000.

(Note all variable values below were observed as a part of the experiment.)

- 'tCount' - 'double' - Number of completed HTML markup tags parsed within chunks. Tags markup content within documents and are opened with character '<' and closed with character '>'.
- 'fileSize' - 'double' - Total root file size in bytes.
- 'retransmits' - 'double' - Number of packets smaller than chunkSize indicating lost bytes that required a retransmit of data.
- 'connectTime' - 'double' - Time in seconds to establish connection from source to destination.
- 'requestTime' - 'double' - Time in seconds to submit request from source to destination.
- 'receiveTime' - 'double' - Time in seconds to receive root file from destination to source.

Exploratory Data Analysis

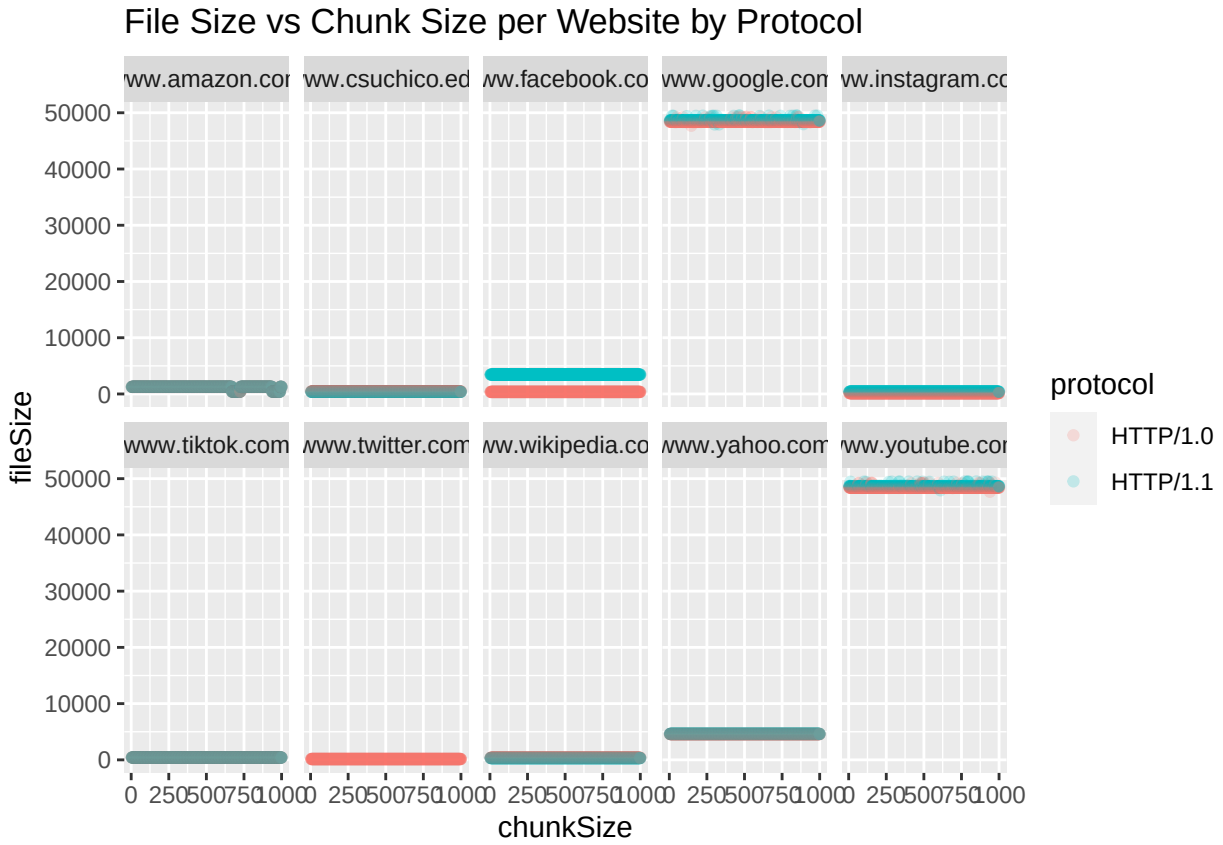
```
summary(netdat)
```

```
##     source          destination          protocol           chunkSize
##  Length:18943       Length:18943       Length:18943       Min.   :   4
##  Class :character   Class :character   Class :character   1st Qu.: 253
##  Mode  :character   Mode  :character   Mode  :character   Median : 502
##                                                           Mean   : 502
##                                                           3rd Qu.: 751
##                                                           Max.   :1000
##      tCount          fileSize      retransmits      connectTime
##  Min.   :  0.00   Min.   :  150   Min.   : 1.000   Min.   :0.000383
##  1st Qu.: 11.00   1st Qu.:  366   1st Qu.: 2.000   1st Qu.:0.000444
##  Median : 15.00   Median :  448   Median : 2.000   Median :0.000470
##  Mean   : 78.87   Mean   :11191   Mean   : 4.881   Mean   :0.000480
##  3rd Qu.: 60.00   3rd Qu.: 4612   3rd Qu.: 2.000   3rd Qu.:0.000504
##  Max.   :311.00   Max.   :49515   Max.   :29.000   Max.   :0.001922
##   requestTime         receiveTime
##  Min.   :2.300e-05   Min.   :0.0000050
##  1st Qu.:4.100e-05   1st Qu.:0.0000090
##  Median :4.600e-05   Median :0.0000230
##  Mean   :4.757e-05   Mean   :0.0001213
##  3rd Qu.:5.000e-05   3rd Qu.:0.0000750
##  Max.   :5.660e-04   Max.   :0.0067130
```

Summary Observations: This experiment collected 18943 total observations. chunkSize was incremented from a minimum of 4 to a maximum of 1000. 4 was chosen as an arbitrary minimum to establish a pattern for tag counting. In some instances there were no tags returned, indicating an index file request may have resulted in a plain text response rather than HTML. Every request resulted in a response as the minimum file size returned is 150 bytes.

File Size:

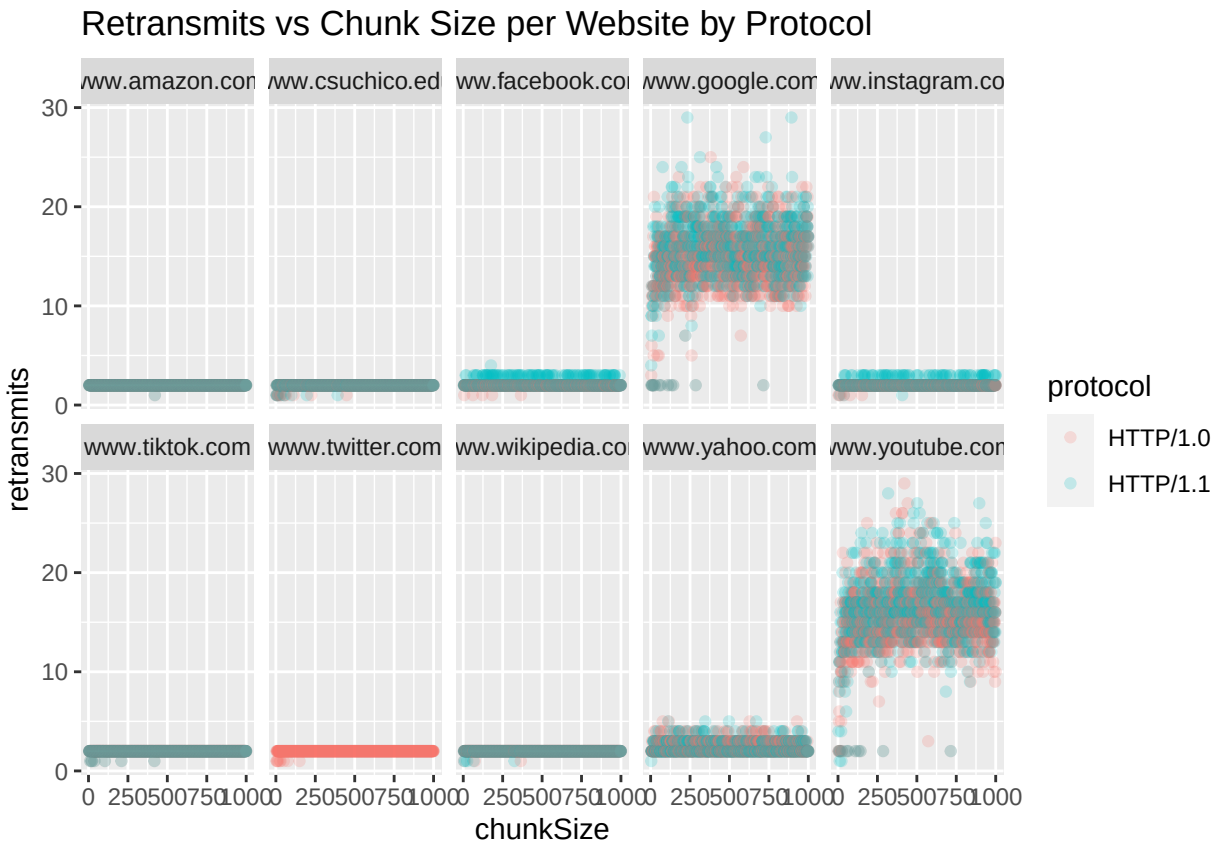
```
ggplot(data = netdat) +
  geom_point(mapping = aes(x = chunkSize, y = fileSize, color = protocol), alpha = 0.2) +
  facet_wrap(~ destination, nrow = 2) +
  ggtitle("File Size vs Chunk Size per Website by Protocol")
```



File Size Observations: The majority of hosts (destinations) appear to almost transmit no information at all. While the values are mostly above zero, the very small size indicates a request for root file results in a message return rather than an actual file. Google, Yahoo and Youtube reply with a significant amount of information.

Retransmits:

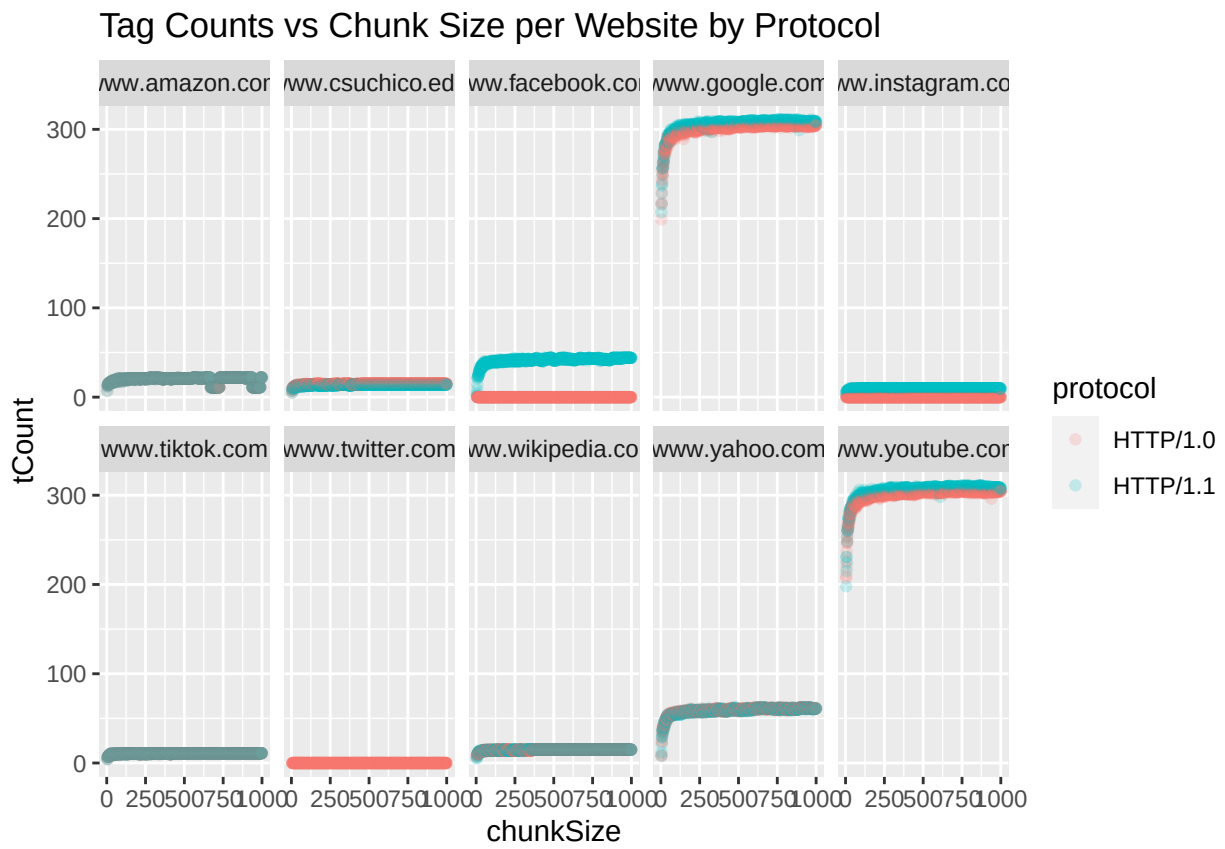
```
ggplot(data = netdat) +
  geom_point(mapping = aes(x = chunkSize, y = retransmits, color = protocol), alpha = 0.2) +
  facet_wrap(~ destination, nrow = 2) +
  ggtitle("Retransmits vs Chunk Size per Website by Protocol")
```



Retransmits Observations: From this series of graphs there does not appear to be a pattern for chunksize impact on retransmits for all sites. On half of the sites we do see a drop to almost no retransmits with smaller chunk sizes.

Tag Counts:

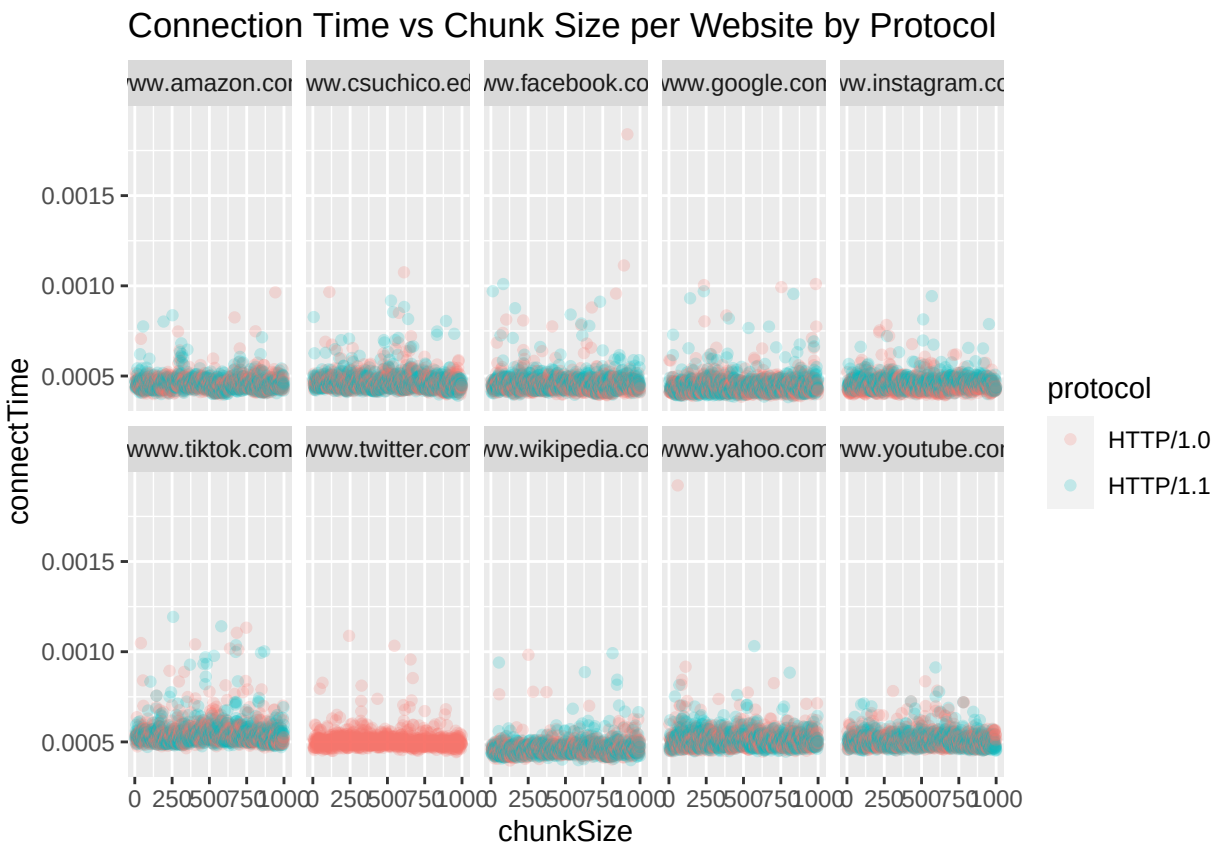
```
ggplot(data = netdat) +
  geom_point(mapping = aes(x = chunkSize, y = tCount, color = protocol), alpha = 0.2) +
  facet_wrap(~ destination, nrow = 2) +
  ggtitle("Tag Counts vs Chunk Size per Website by Protocol")
```



Tag Counts Observations: In general where filesizes are significant a smaller chunk size will result in fewer tag counts. This is probably due to the cutoff for counts in between chunks. This seems to clear out after a chunk size of 100. In the case of Youtube and Google there is a marginarlly larger tag count using HTTP/1.1. For Facebook and Instagram when using HTTP/1.0 there does not appear to be any tags returned, indicating a plain text response in comparison to a marked down response with HTTP/1.1.

Connection Time:

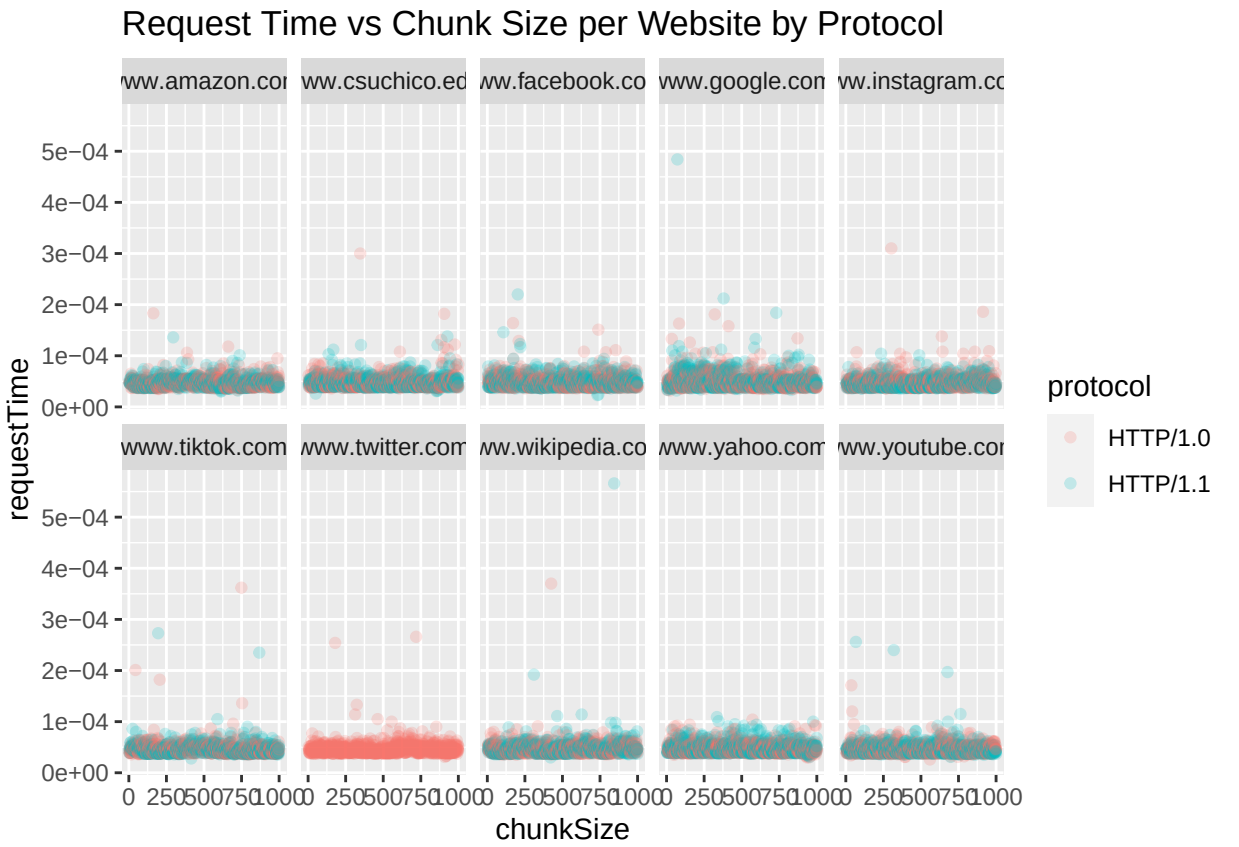
```
ggplot(data = netdat) +
  geom_point(mapping = aes(x = chunkSize, y = connectTime, color = protocol), alpha = 0.2) +
  facet_wrap(~ destination, nrow = 2) +
  ggtitle("Connection Time vs Chunk Size per Website by Protocol")
```



Connection Time Observations: There are very few deviations for connection time in all sites regardless of chunk size or protocol use.

Request Time:

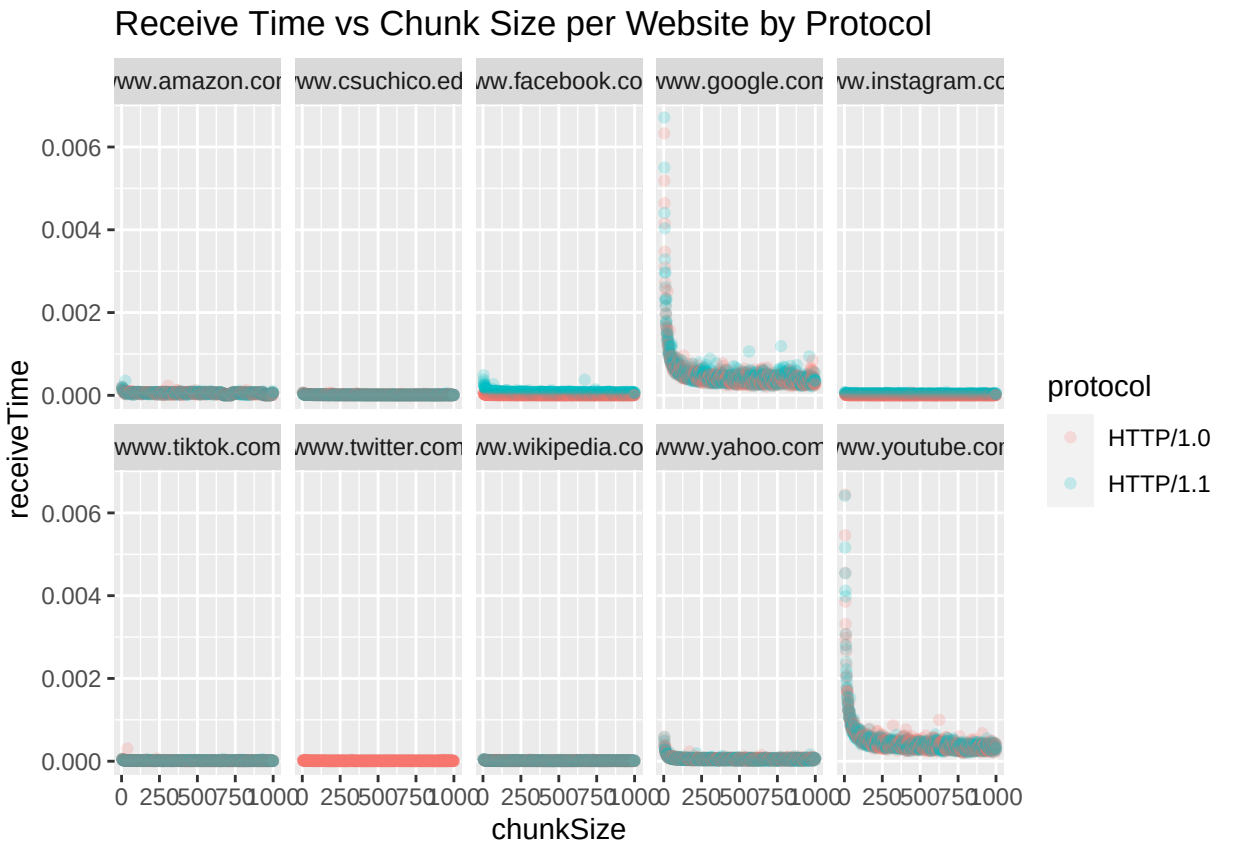
```
ggplot(data = netdat) +
  geom_point(mapping = aes(x = chunkSize, y = requestTime, color = protocol), alpha = 0.2) +
  facet_wrap(~ destination, nrow = 2) +
  ggtitle("Request Time vs Chunk Size per Website by Protocol")
```



Request Time Observations: There are very few deviations for request time in all sites regardless of chunk size or protocol use.

Receive Time:

```
ggplot(data = netdat) +
  geom_point(mapping = aes(x = chunkSize, y = receiveTime, color = protocol), alpha = 0.2) +
  facet_wrap(~ destination, nrow = 2) +
  ggtitle("Receive Time vs Chunk Size per Website by Protocol")
```



Receive Time Observations: For sites returning significant data, there is a clear spiked increase in receiving time up until the chunk size is greater than 100 bytes.

Data Science Questions

Which websites contain larger root files? Among the sites tested, Google, Youtube and Yahoo returned significantly larger root files when a simple GET request was sent.

Is there a performance difference between HTTP 1.0 and HTTP 1.1? In the data set collected from this experiment, there did not appear to be a noticeable difference between either protocol.

How does the size of chunk parsing impact response time and tag count? Response time decreases until chunks are parse at more than 100 bytes with no observed further improvement past this point. Tags were parsed less often up to 100 bytes. The only advantage to smaller chunk usage was a very minor decrease in the frequency of retransmits.

Predictive Conclusion: To maximum speed of receiving HTTP/1.0 and HTTP/1.1 files on all websites, do not set chunk size parsing to less than 100 bytes.

Ethical Implications

Opacity: While the data collection is relatively straightforward and my code is available for inspection to recreate my experiment, an understanding of C programming is required for scrutiny. The learning curve is rudimentary however.

Scale: Although my experiment can be reran to include other protocols and a much larger number of websites, the R modeling here would need to be adjusted and reconsidered for such a larger scale.

Damage: Benchmarks results could be misinterpreted if compared to newer versions of protocols for web file transfer. HTTP/2 was introduced in 2015 and is currently used by 7% of web browsers. Unlike HTTP/1.0 and 1.1, this new protocol allows synchronous communication through web sockets for two way simulanteous communication. HTTP/3 is an upcoming standard that is currently only in use by Safari 14 introduced in September 2020, though over 10 million websites have implemented support for this newer standard. Unlike HTTP/2 and HTTP 1.0 and 1.1, HTTP/3 uses a new transport protocol called QUIC (general purpose transfer protocol) instead of the previous TCP (transfer control protocol). Future experiments might include HTTP/2, HTTP/3 and other protocol comparisons.