Variant Data Comparison Between Websites with Machine Learning Analysis

Northeastern University

Author: Joshua Conte

Class: DA5030 - Introduction to Data Mining and Machine Learning

Instructor: Sara Arunagiri, Ph.D.

Date: December 11, 2017

Contents

1 Abstract

The final project for the class DA5030 Introduction to Data Mining and Machine Learning is a data mining project that scrapes data from the HGMD website and compares it to the ExAC website and reports the duplicate and unique variants. Then the unique data is analyzed using unsupervised learning algorithms.

2 Background

There is a website called, HGMD, which has a list of all published mutations in every gene. The user enters the specific gene and the website shows a list of all of the published mutations, which contains the variant and position. However, some of these publications are from 20 years ago where they might not be considered mutations today, now that we have more information. Genetic information is constantly being updated as more research is being conducted. To check if the published mutation is still considered a mutation, there is another website, ExAC, which has a list of all known variants for all known genes that are not mutations. The user can enter the specific gene and get a list of all the variants and the position. Then the variants and positions are compared from each website, if the two match, then the variant is no longer considered a mutation. Currently, geneticists need to perform this analysis manually by visually comparing the websites. The reason I want to do this project is because my wife and her colleges need to do this manually and her bioinformatics department is too busy at this time to find a solution. My hope is to find a solution and present it to her team and eventually get a job in their bioinformatics group.

To perform this project I plan to have the user enter the URL's with the gene info into R. Then R will scrape the websites and put the data into data frames. I will compare the data frames and print the differences. I am not sure of the details now, I will need to research the best method to do this (use R and compare or make a database with the info and use SQL). I want to also see if it is possible to enter the gene into R, as a text input, and have R search the database for the gene info then scrape the website, I am not sure if it is possible and I will need to research it.

This project is dependent on if it is possible to use R to scrape the information from the websites, I am not sure if it is possible with R and I am currently trying to figure it out. Also, I will need permission to use the HGMD website, it is very expensive to get a licenses and I will ask the company for a 2 month trial for a graduate research project. If they give me a trial then I can do it, if not, it is not feasible to pursue this project.

3 Data sources

Variant data used will be scraped from ExAC and HGMD websites.

4 Algorithms and Data Sets

4.1 Data Sets

The r packages used for this assignemt are:

For scraping and shaping the data

- rvest[1]
- RSelenium[2]
- plyr[3]

For analysis

- ggplot2[4]
- cluster[5]
- amap[6]
- useful[7]

4.2 Algorithms

For analysis I will use unsupervised learning algorithms.

The goal for unsupervised learning is to model the underlying structure or distribution in the data in order to learn more about the data. Algorithms are left to their own devises to discover and present the interesting structure in the data[8].

4.2.1 Types of Clustering

Clustering (e.g., k-means, mixture models, hierarchical clustering). Cluster analysis or clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense or another) to each other than to those in other groups (clusters)[9]. There are various types of cluster analysis.

- Partitioning-based clustering (K-means and its variants)
- Hierarchical clustering

4.2.1.1 Partitioning-based clustering

4.2.1.1.1 K-means

k-means clustering is a method of vector quantization, originally from signal processing, that is popular for cluster analysis in data mining. k-means clustering aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster. This results in a partitioning of the data space into Voronoi cells[10].

4.2.1.1.2 K-medoids (PAM)

The K-medoids or Partitioning Around Medoids (PAM) algorithm (Kaufman & Rousseeuw'87) is related to the k-means algorithm and but uses medoid shifts rather than reassigning points based on Euclidean distance. Each cluster is represented by one of the objects (i.e. points) in the cluster A medoid is a point in a cluster whose dissimilarity to all the points in the cluster is minimal. Medoids are similar in concept to means or centroids, but medoids are always members of the data set. That is, in 2D Cartesian space a centroid can be any valid x.y coordinate. Whereas a medoid must be one of the data points[11].

Pseduocode for the k-medoid clustering (Partitioning Around Medoids (PAM)) algorithm:

```
Initialize: randomly select[citation needed] (without replacement) k of the n data points as the medoids

Associate each data point to the closest medoid.

While the cost of the configuration decreases:
  For each medoid m, for each non-medoid data point o:
    Swap m and o, recompute the cost

  If the total cost of the configuration increased in the previous step, undo   the swap.
```

4.2.1.2 Hierarchical clustering

In hierarchical clustering the idea is to group data objects (i.e. points) into a tree of clusters. That is, hierarchical clustering is a method of cluster analysis which seeks to build a hierarchy of clusters[12].

These trees (hierarchies) generally fall into two types:

4.2.1.2.1 Agglomerative hierarchical clustering

Initially each data object (i.e. point) in its own cluster. Iteratively the clusters are merged together from the "bottom-up." The two most similar/closest objects are aggreated in to the same cluster/data object. Then the next two, until there is just one cluster/data object. This agglomerative approach result in straggly (long and thin) clusters due to a chaining effect. It is also sensitive to noise[11].

4.2.1.2.2 Divisive hierarchical clustering

In divisive hierarchical clustering all data objects (i.e. points) are initially in one cluster. These clusters are successively divided recursivley in a "top-down" manner. The cluster is broken in to two clusters that are most dissimilar. Then each of those clusters is broken in to two cluster that are most dissimilar. This continues until each clsuter is a single data object (i.e. point)[11].

5 Scraping Websites

Below is a walkthrough without showing the code. I did this because I needed to use if statements and the first chunk of code is over 250 lines. This section will show what the program does and the next section will show the code.

In order to scrape the ExAC website I need to use RSelenium and rvest. This is because the table was constructed using javascript, specifically jQuery. When a website makes use of JavaScript to display data, the rvest and XML packages miss the required functionality.

Selenium is a web automation tool that literally "drives" the browser, so it can see anything you see when you right click and inspect element in Chrome or Firefox, making it possible to scrape the information with rvest. This vastly widens the universe of content that can be extracted from automation, but can be slow as all content must be rendered in the browser.

The great thing about RSelenium is that the user does not need to download and install a Selenium server, RSelenium will download it and run it automatically. Below is the R code.

5.1 Scraping the ExAC Website

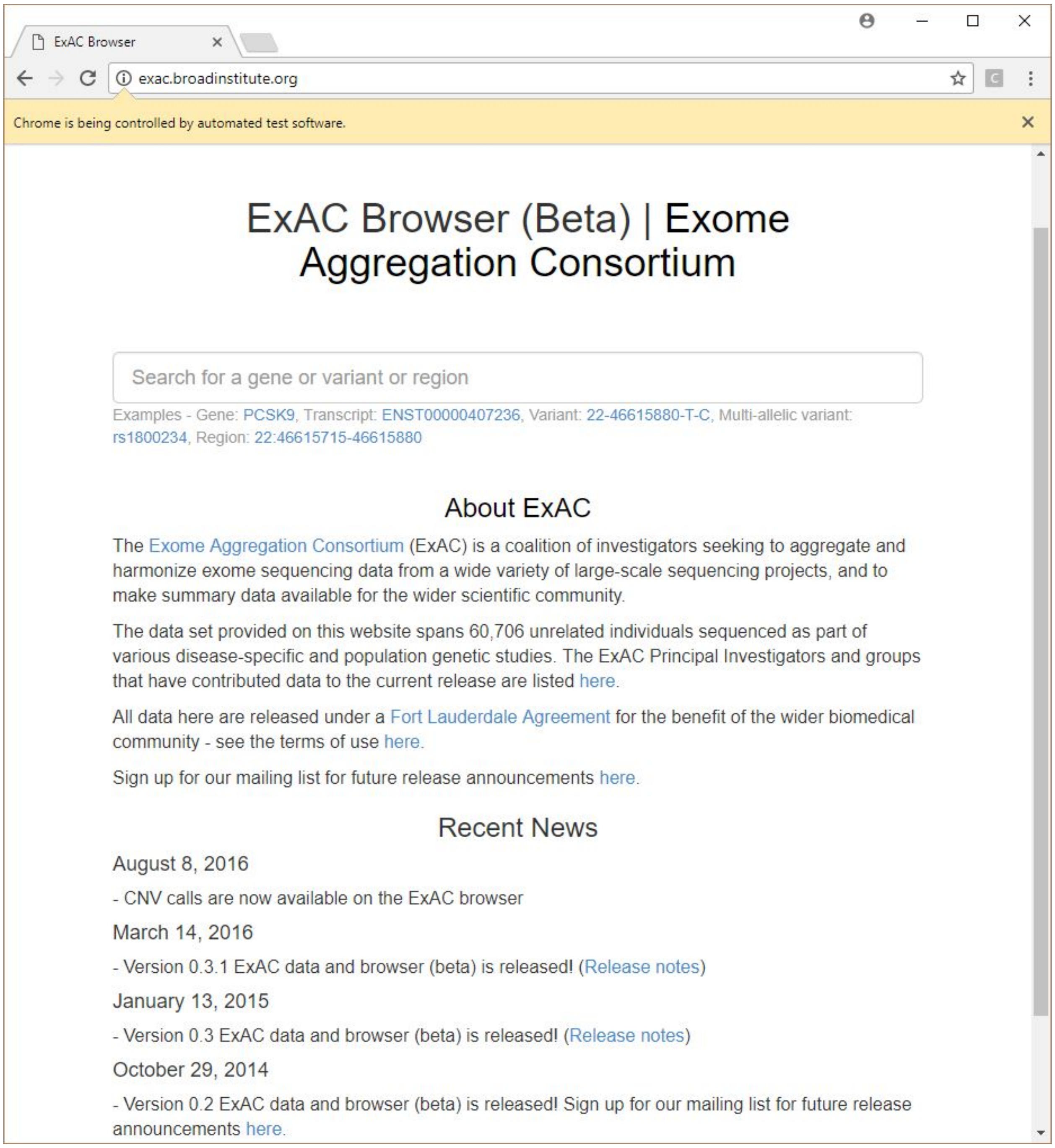
Once RSelenium is up and running, the website is opened through RSelenium as defined in the code, figure 1 shows an example of what it looks like.

Then the gene name is automatically entered (this is an input in the code) and RSelenium searches for the gene and scrapes the data and puts it into a data frame. Figure 2 shows an image of the website.

5.2 Scraping the HGMD Website

Now that the ExAC data has been collected it is time to move to HGMD. I began by opening the website, see figure 3 as reference.

Then the username and password were entered automatically, which are defined in the code.



Chrome is being controlled by automated test software.

ExAC Browser (Beta) | Exome Aggregation Consortium

Search for a gene or variant or region

Examples - Gene: PCSK9, Transcript: ENST00000407236, Variant: 22-46615880-T-C, Multi-allelic variant: rs1800234, Region: 22:46615715-46615880

About ExAC

The Exome Aggregation Consortium (ExAC) is a coalition of investigators seeking to aggregate and harmonize exome sequencing data from a wide variety of large-scale sequencing projects, and to make summary data available for the wider scientific community.

The data set provided on this website spans 60,706 unrelated individuals sequenced as part of various disease-specific and population genetic studies. The ExAC Principal Investigators and groups that have contributed data to the current release are listed here.

All data here are released under a Fort Lauderdale Agreement for the benefit of the wider biomedical community - see the terms of use here.

Sign up for our mailing list for future release announcements here.

Recent News

August 8, 2016

- CNV calls are now available on the ExAC browser

March 14, 2016

- Version 0.3.1 ExAC data and browser (beta) is released! (Release notes)

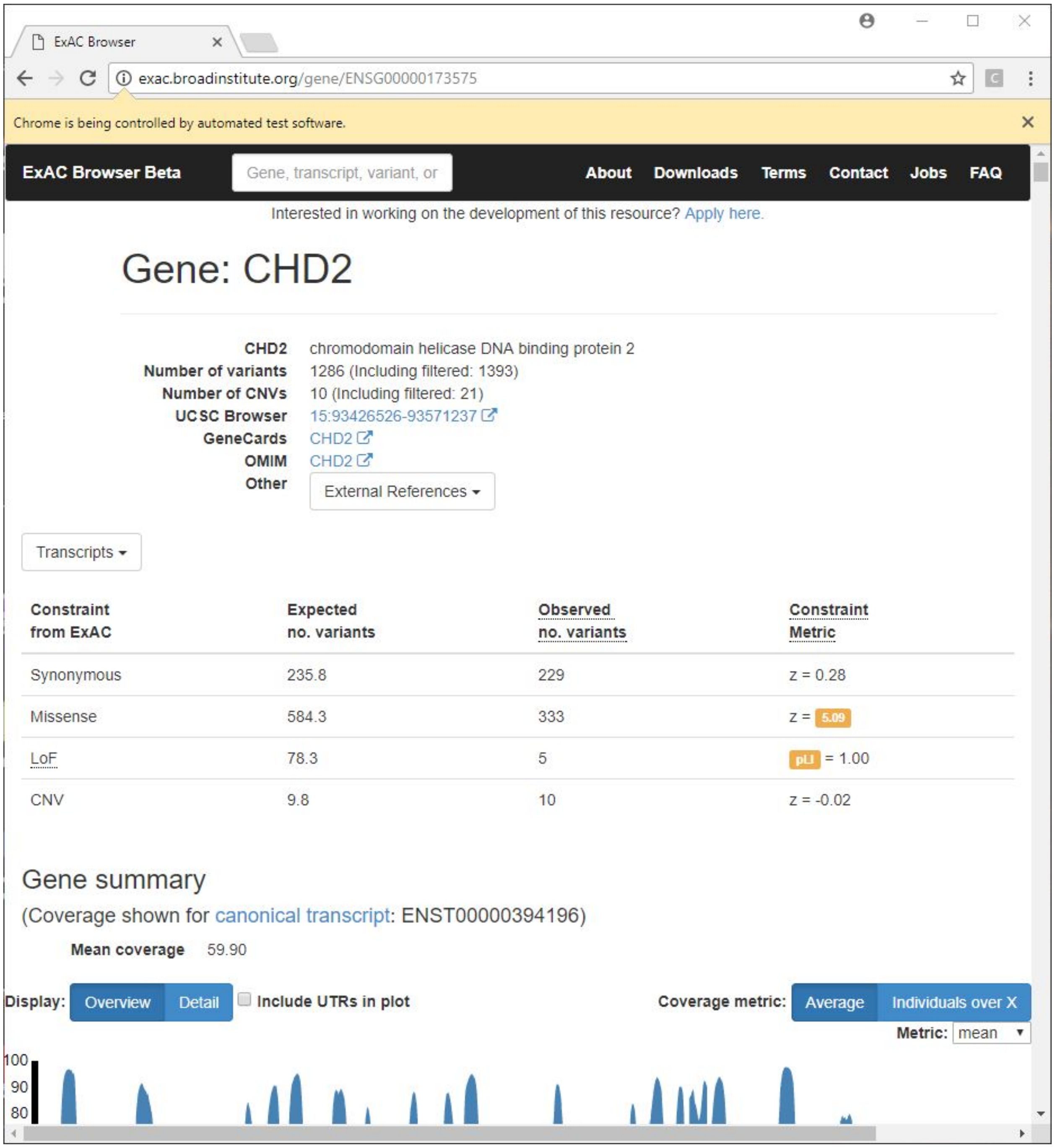
January 13, 2015

- Version 0.3 ExAC data and browser (beta) is released! (Release notes)

October 29, 2014

- Version 0.2 ExAC data and browser (beta) is released! Sign up for our mailing list for future release announcements here.

Figure 1: ExAC Webpage



ExAC Browser

exac.broadinstitute.org/gene/ENSG00000173575

Chrome is being controlled by automated test software.

ExAC Browser Beta — About — Downloads — Terms — Contact — Jobs — FAQ

Interested in working on the development of this resource? Apply here.

Gene: CHD2

CHD2	chromodomain helicase DNA binding protein 2
Number of variants	1286 (Including filtered: 1393)
Number of CNVs	10 (Including filtered: 21)
UCSC Browser	15:93426526-93571237
GeneCards	CHD2
OMIM	CHD2
Other	External References

Transcripts

Constraint from ExAC	Expected no. variants	Observed no. variants	Constraint Metric
Synonymous	235.8	229	z = 0.28
Missense	584.3	333	z = 5.09
LoF	78.3	5	pLI = 1.00
CNV	9.8	10	z = -0.02

Gene summary

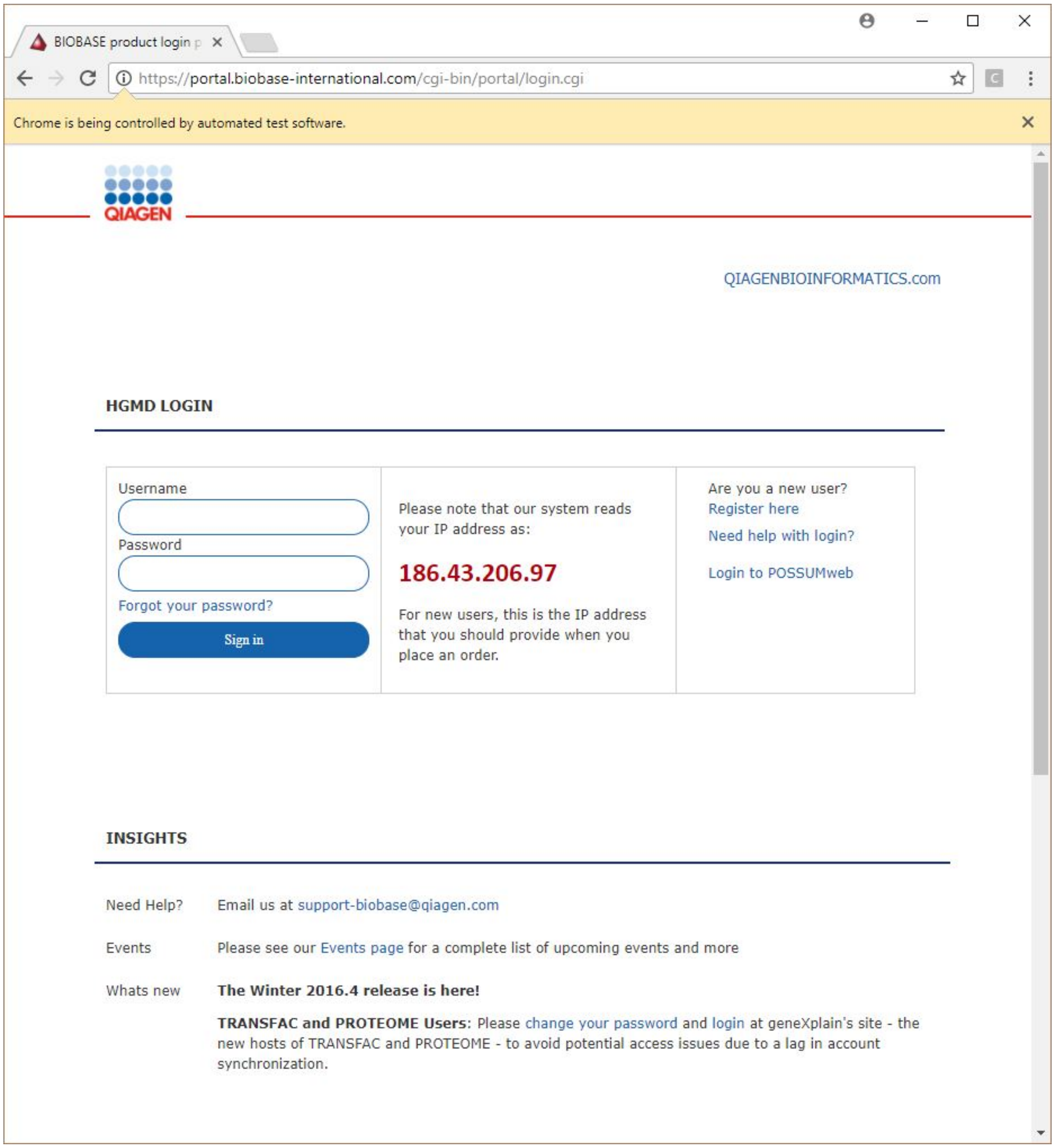
(Coverage shown for canonical transcript: ENST00000394196)

Mean coverage 59.90

Display: Overview Detail Include UTRs in plot — Coverage metric: Average Individuals over X

Metric: mean

Figure 2: ExAC CHD2 Summary



Chrome is being controlled by automated test software.

HGMD LOGIN

Are you a new user?

Please note that our system reads your IP address as:

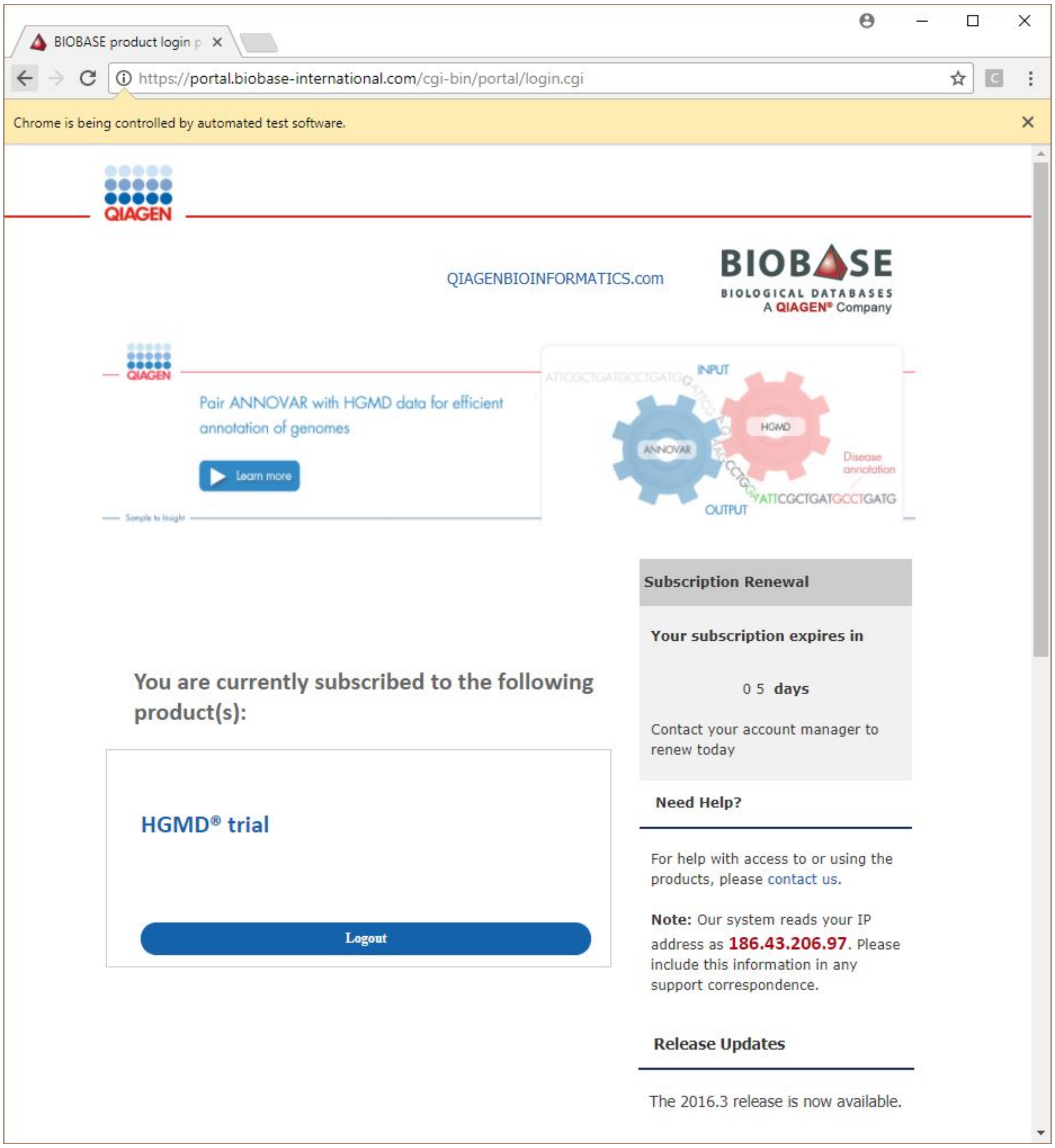
186.43.206.97

For new users, this is the IP address that you should provide when you place an order.

INSIGHTS

Need Help?	Email us at support-biobase@qiagen.com
Events	Please see our Events page for a complete list of upcoming events and more
Whats new	**The Winter 2016.4 release is here!**
	TRANSFAC and PROTEOME Users: Please change your password and login at geneXplain's site - the new hosts of TRANSFAC and PROTEOME - to avoid potential access issues due to a lag in account synchronization.

Figure 3: HGMD Webpage



Chrome is being controlled by automated test software.

You are currently subscribed to the following product(s):

HGMD® trial

Subscription Renewal

Your subscription expires in

0 5 days

Contact your account manager to renew today

Need Help?

For help with access to or using the products, please contact us.

Note: Our system reads your IP address as 186.43.206.97. Please include this information in any support correspondence.

Release Updates

The 2016.3 release is now available.

Figure 4: HGMD Webpage After Password

Figure 4 is what the website looks like after the username and password are entered.

Now the link needs to be "clicked". Figure 5 shows the website after the linked has been "clicked".

Then an authentication popup window appears where the username and password need to be entered. RSelenium deals with this by putting the username and password into the URL. The code chunk is shown below:

```
# An authentication window pops up, to get around this
    # I need to switch to the main window
    # get all windows
    windows <- remDr$getWindowHandles()

    # Swith to main window
    remDr$switchToWindow(windows[[1]])

    # This enters the username and password into the url
    HGMD_url2 <-
      paste(
        "http://",
        username,
        ":",
        password,
        "@hgmdtrial.biobase-international.com/hgmd/pro/start.php",
        sep = ""
      )

    # This goes to the HGMD website
    remDr$navigate(HGMD_url2)

    # Swith to other window
    remDr$switchToWindow(windows[[2]])

    # Close the window
    remDr$closeWindow()

    # Swith to main window
    remDr$switchToWindow(windows[[1]])
```

Figure 6 is what the website looks like after the authentication information is entered.

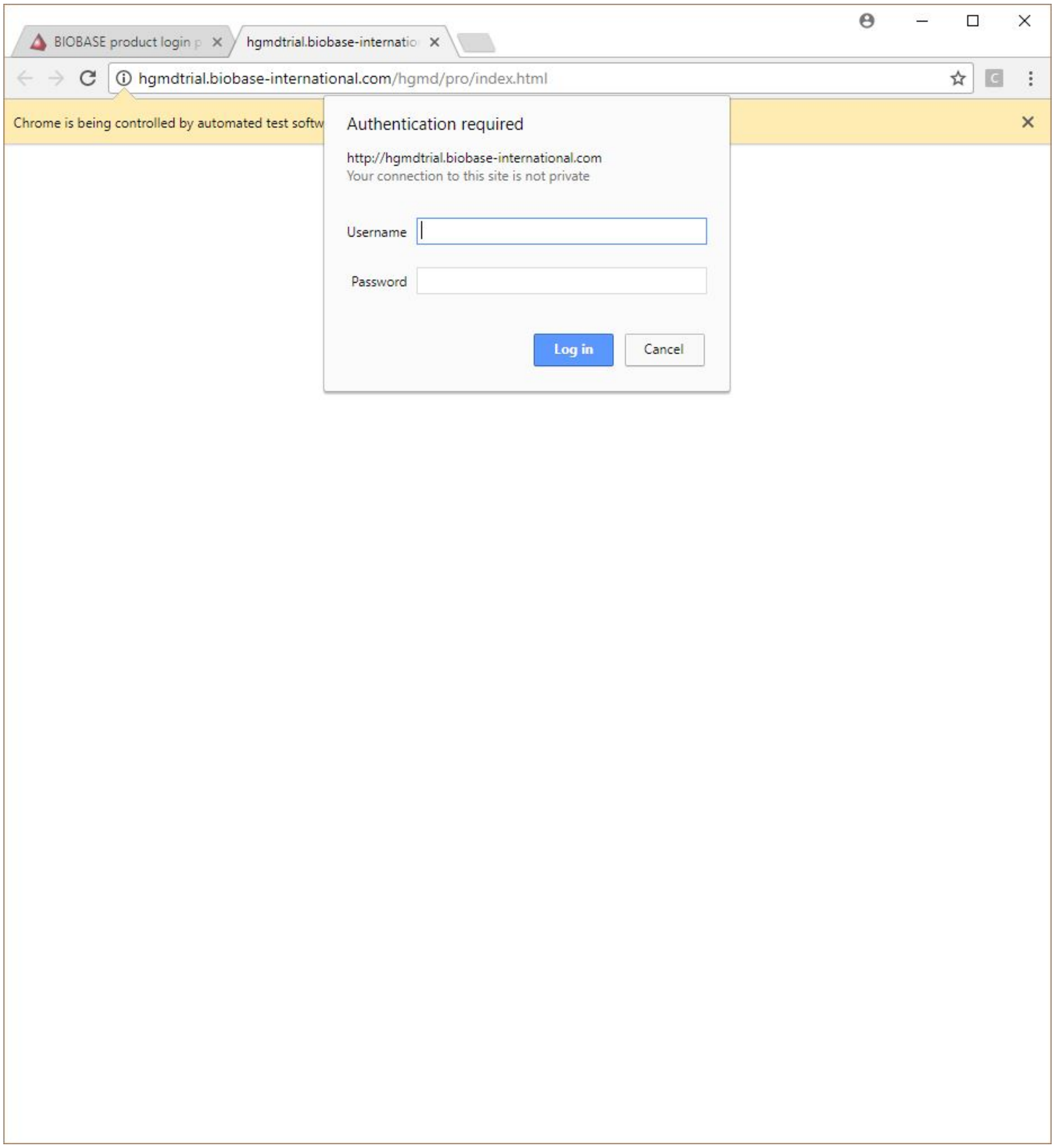
Next, RSelenium "clicks" the gene tab in the upper left corner to search for the gene. Figure 7 shows the result and below id the code.

Next, the gene is entered in the textbox and "enter" is pressed. Then a webpage is displayed with a summary of the gene, as shown in figure 8.

Next, the "Get all mutations" link is pressed and this opens a webpage with all of the mutations in four different categories: Missense, splicing, deletions, and insertion. These are the four tables that are needed to be scraped. The webpage are shown in the images below.

6 Coding with R

R studio was configured with the following parameters before beginning the project:



BIOBASE product login p

hgmdtrial.biobase-internatio

hgmdtrial.biobase-international.com/hgmd/pro/index.html

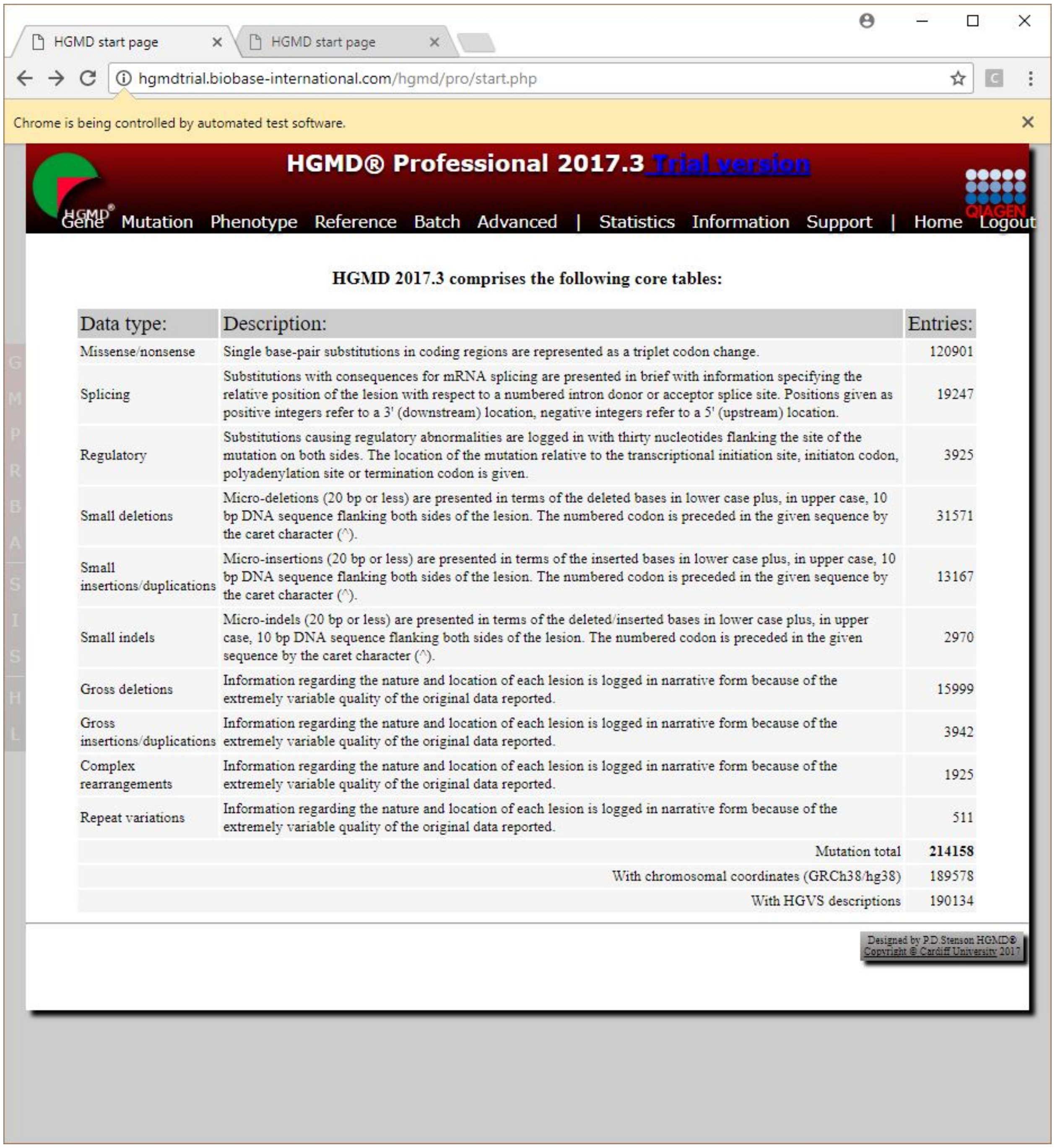
Chrome is being controlled by automated test softw

Authentication required

http://hgmdtrial.biobase-international.com

Your connection to this site is not private

Figure 5: HGMD Webpage After Click

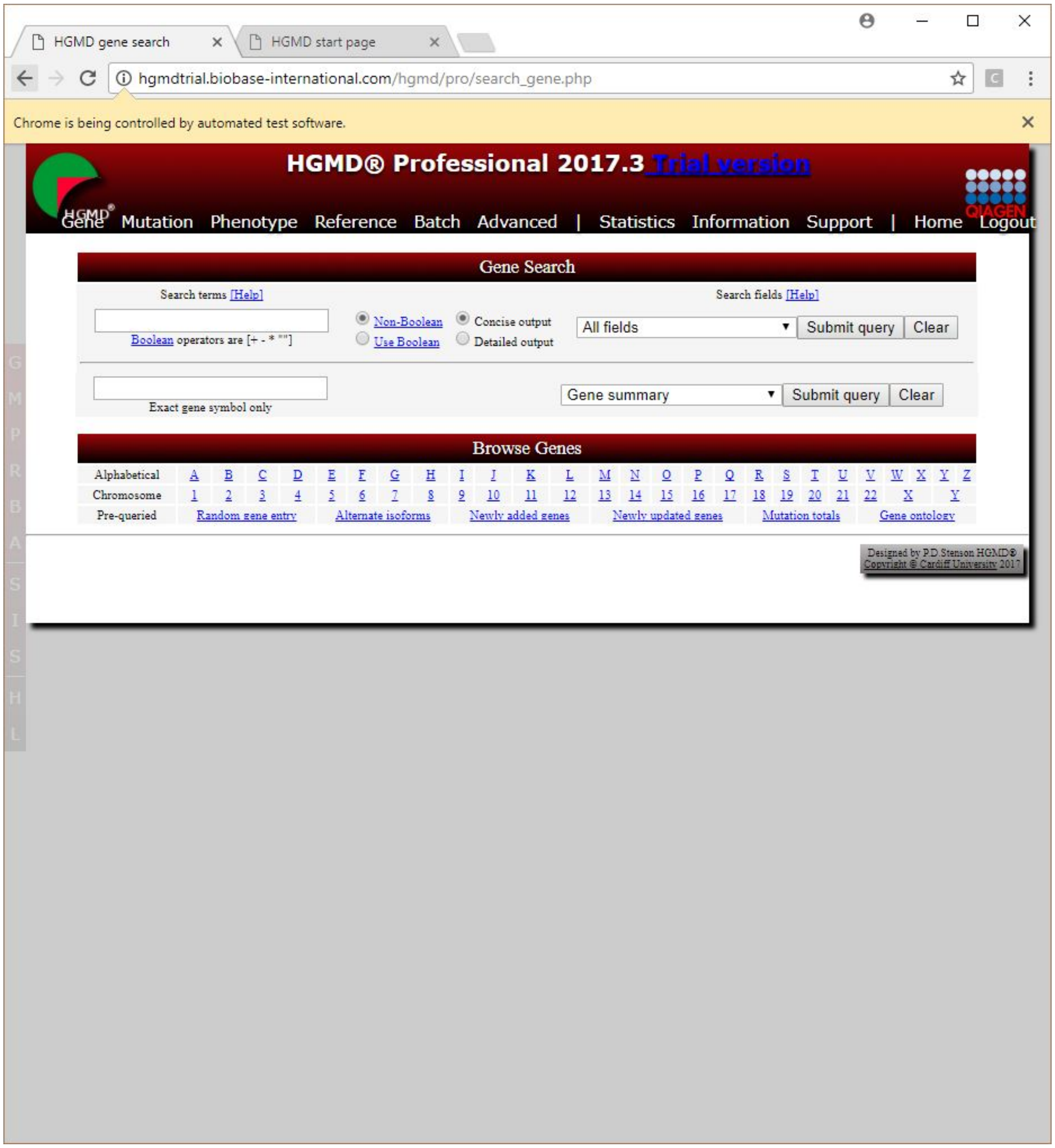

HGMD start page
HGMD start page
hgmdtrial.biobase-international.com/hgmd/pro/start.php
Chrome is being controlled by automated test software.
HGMD® Professional 2017.3 Trial version
Gene Mutation Phenotype Reference Batch Advanced | Statistics Information Support | Home Logout
QIAGEN

HGMD 2017.3 comprises the following core tables:

Data type:	Description:	Entries:
Missense/nonsense	Single base-pair substitutions in coding regions are represented as a triplet codon change.	120901
Splicing	Substitutions with consequences for mRNA splicing are presented in brief with information specifying the relative position of the lesion with respect to a numbered intron donor or acceptor splice site. Positions given as positive integers refer to a 3' (downstream) location, negative integers refer to a 5' (upstream) location.	19247
Regulatory	Substitutions causing regulatory abnormalities are logged in with thirty nucleotides flanking the site of the mutation on both sides. The location of the mutation relative to the transcriptional initiation site, initiaton codon, polyadenylation site or termination codon is given.	3925
Small deletions	Micro-deletions (20 bp or less) are presented in terms of the deleted bases in lower case plus, in upper case, 10 bp DNA sequence flanking both sides of the lesion. The numbered codon is preceded in the given sequence by the caret character (^).	31571
Small insertions/duplications	Micro-insertions (20 bp or less) are presented in terms of the inserted bases in lower case plus, in upper case, 10 bp DNA sequence flanking both sides of the lesion. The numbered codon is preceded in the given sequence by the caret character (^).	13167
Small indels	Micro-indels (20 bp or less) are presented in terms of the deleted/inserted bases in lower case plus, in upper case, 10 bp DNA sequence flanking both sides of the lesion. The numbered codon is preceded in the given sequence by the caret character (^).	2970
Gross deletions	Information regarding the nature and location of each lesion is logged in narrative form because of the extremely variable quality of the original data reported.	15999
Gross insertions/duplications	Information regarding the nature and location of each lesion is logged in narrative form because of the extremely variable quality of the original data reported.	3942
Complex rearrangements	Information regarding the nature and location of each lesion is logged in narrative form because of the extremely variable quality of the original data reported.	1925
Repeat variations	Information regarding the nature and location of each lesion is logged in narrative form because of the extremely variable quality of the original data reported.	511
	Mutation total	**214158**
	With chromosomal coordinates (GRCh38/hg38)	189578
	With HGVS descriptions	190134

Designed by P.D.Stenson HGMD®
Copyright © Cardiff University 2017

Figure 6: HGMD Webpage Database


HGMD gene search
HGMD start page
hgmdtrial.biobase-international.com/hgmd/pro/search_gene.php
HGMD
QIAGEN

Chrome is being controlled by automated test software.

HGMD® Professional 2017.3 Trial version

Gene Mutation Phenotype Reference Batch Advanced | Statistics Information Support | Home Logout

Gene Search

Search terms [Help] Search fields [Help]

Boolean operators are [+ - * ""]

[✓ Non-Boolean] Use Boolean [✓ Concise output] Detailed output

All fields ▼ Submit query Clear

Exact gene symbol only

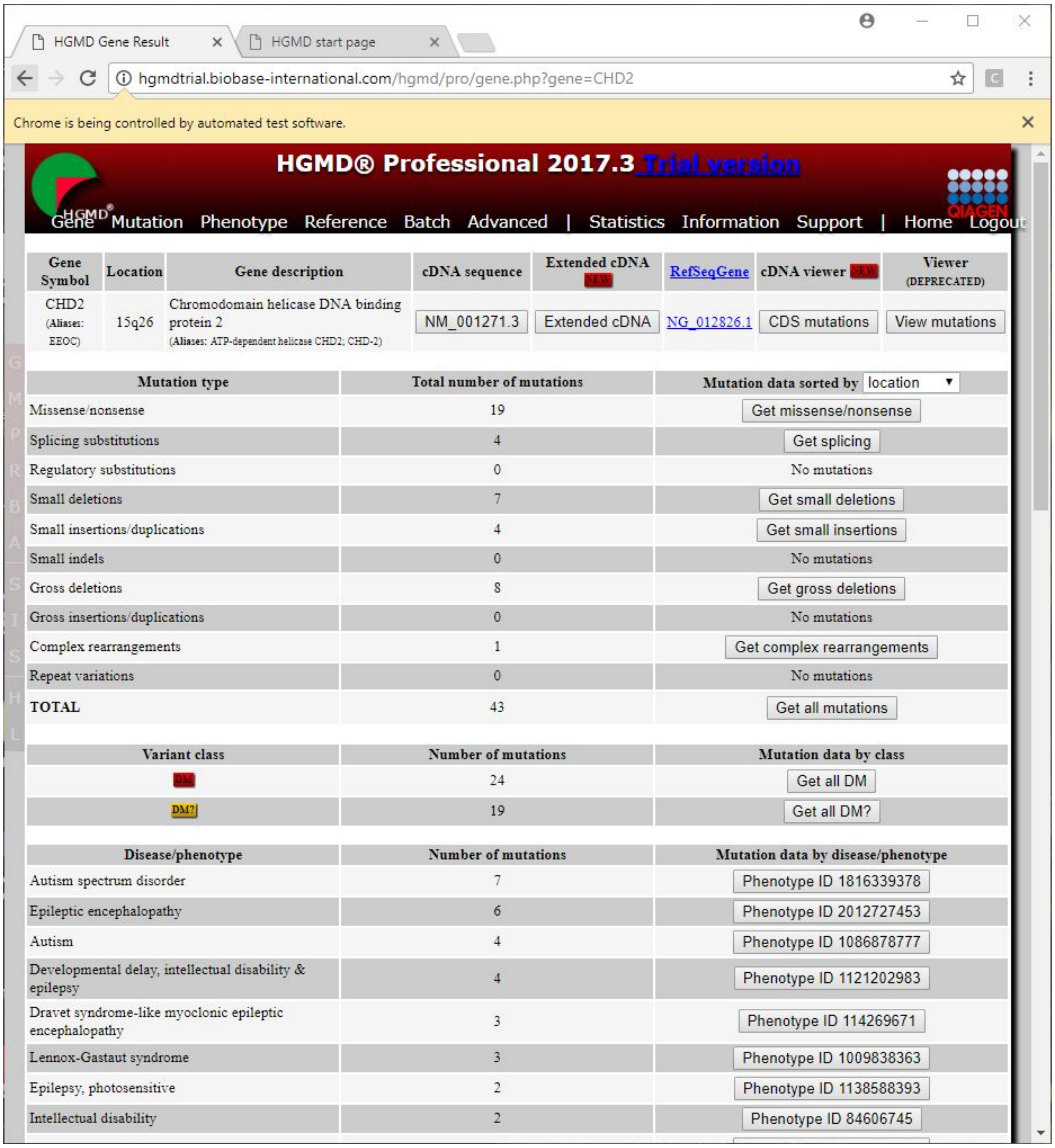
Gene summary ▼ Submit query Clear

Browse Genes

Alphabetical	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Chromosome	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	X	Y		
Pre-queried	Random gene entry	Alternate isoforms	Newly added genes	Newly updated genes	Mutation totals	Gene ontology																				

Designed by P.D.Stenson HGMD®
Copyright © Cardiff University 2017

Figure 7: HGMD Webpage for Genes



HGMD Gene Result
HGMD start page
hgmdtrial.biobase-international.com/hgmd/pro/gene.php?gene=CHD2
Chrome is being controlled by automated test software.
HGMD® Professional 2017.3 Trial version
Gene Mutation Phenotype Reference Batch Advanced | Statistics Information Support | Home Logout

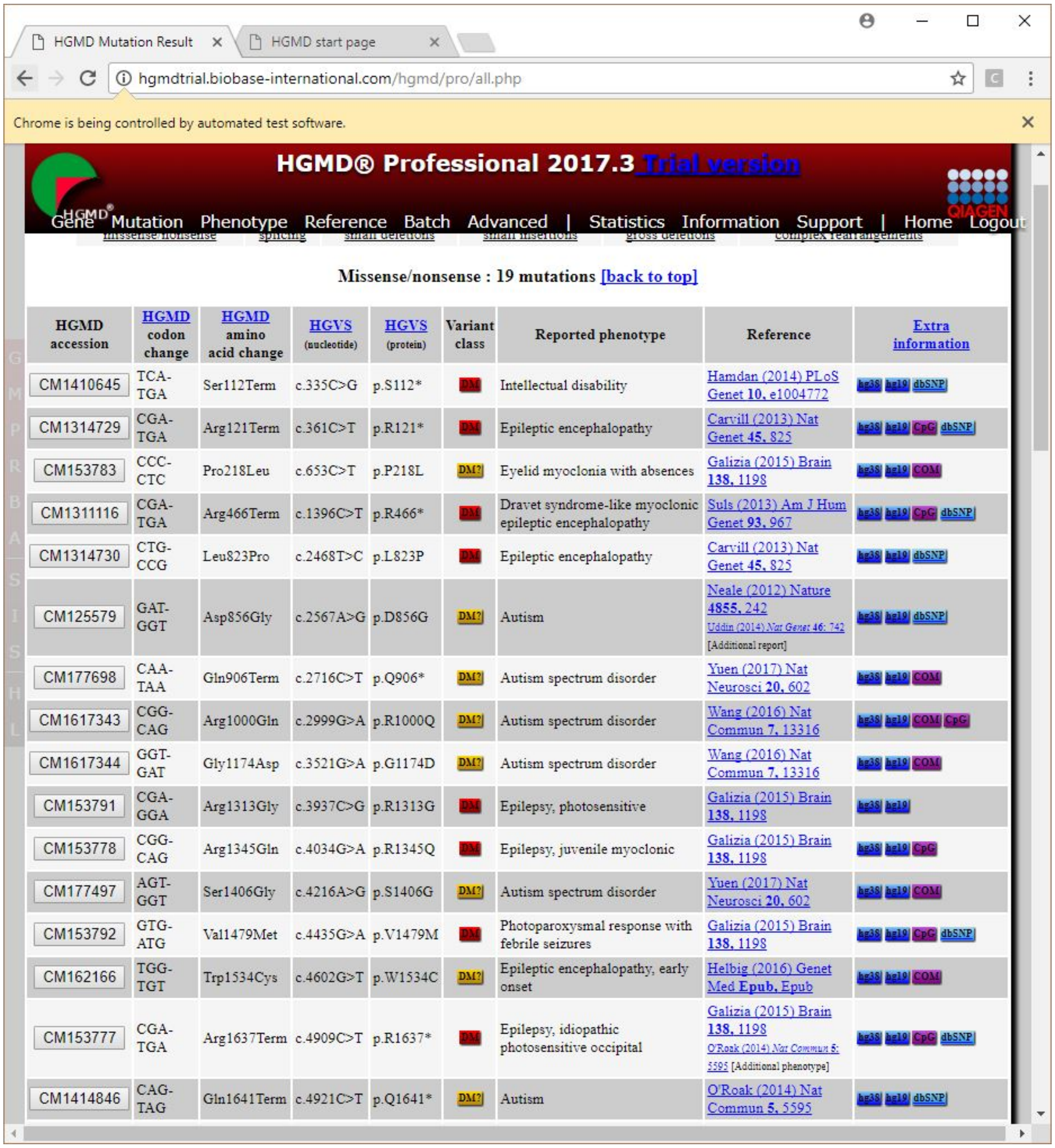
Gene Symbol	Location	Gene description	cDNA sequence	Extended cDNA NEW	RefSeqGene	cDNA viewer NEW	Viewer (DEPRECATED)
CHD2 (Aliases: EEOC)	15q26	Chromodomain helicase DNA binding protein 2 (Aliases: ATP-dependent helicase CHD2; CHD-2)	NM_001271.3	Extended cDNA	NG_012826.1	CDS mutations	View mutations

Mutation type	Total number of mutations	Mutation data sorted by location
Missense/nonsense	19	Get missense/nonsense
Splicing substitutions	4	Get splicing
Regulatory substitutions	0	No mutations
Small deletions	7	Get small deletions
Small insertions/duplications	4	Get small insertions
Small indels	0	No mutations
Gross deletions	8	Get gross deletions
Gross insertions/duplications	0	No mutations
Complex rearrangements	1	Get complex rearrangements
Repeat variations	0	No mutations
TOTAL	43	Get all mutations

Variant class	Number of mutations	Mutation data by class
DM	24	Get all DM
DM?	19	Get all DM?

Disease/phenotype	Number of mutations	Mutation data by disease/phenotype
Autism spectrum disorder	7	Phenotype ID 1816339378
Epileptic encephalopathy	6	Phenotype ID 2012727453
Autism	4	Phenotype ID 1086878777
Developmental delay, intellectual disability & epilepsy	4	Phenotype ID 1121202983
Dravet syndrome-like myoclonic epileptic encephalopathy	3	Phenotype ID 114269671
Lennox-Gastaut syndrome	3	Phenotype ID 1009838363
Epilepsy, photosensitive	2	Phenotype ID 1138588393
Intellectual disability	2	Phenotype ID 84606745

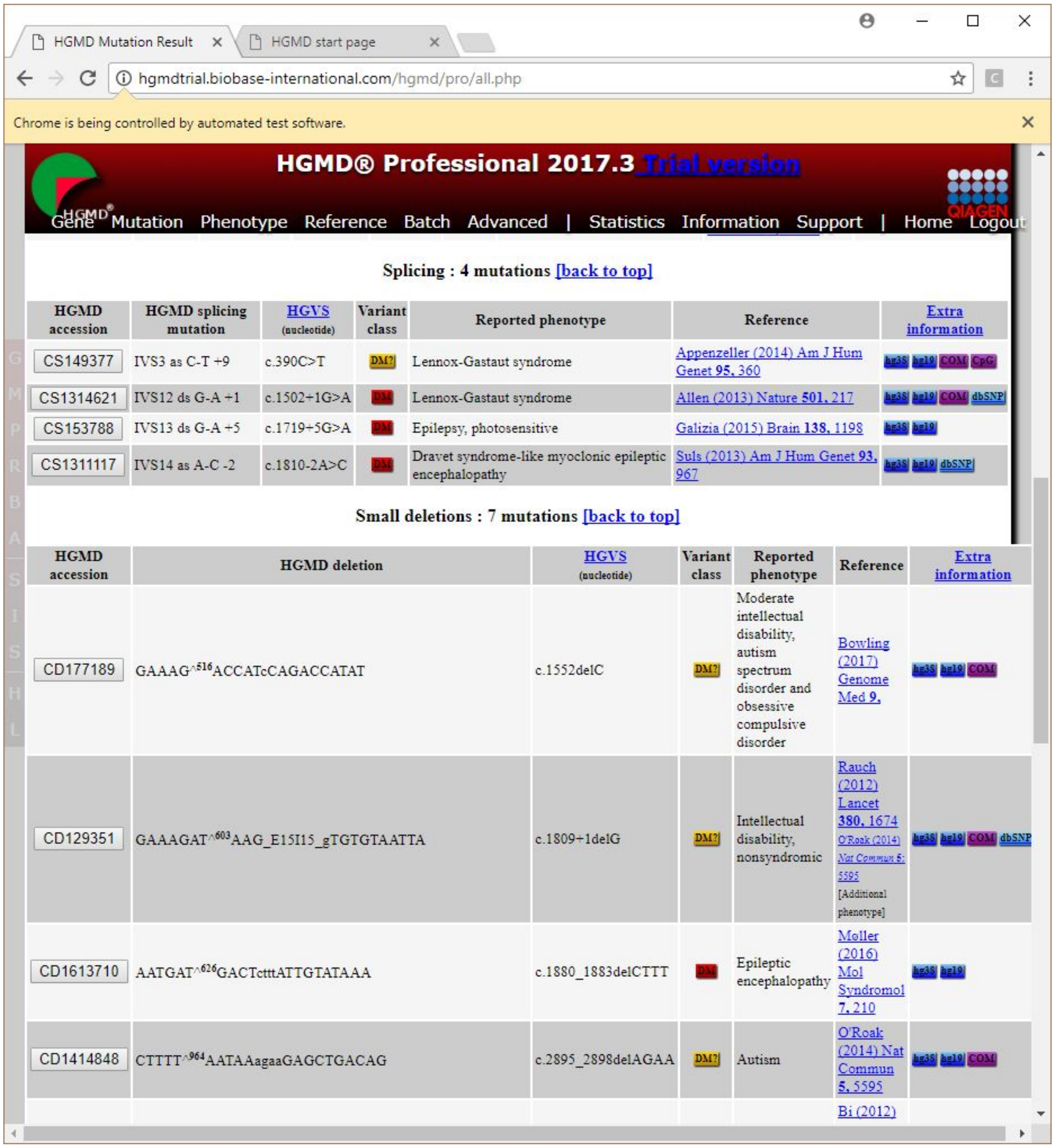
Figure 8: HGMD Webpage for Gene Summary


HGMD Mutation Result
HGMD start page
hgmdtrial.biobase-international.com/hgmd/pro/all.php
Chrome is being controlled by automated test software.
HGMD® Professional 2017.3 Trial version
Gene Mutation Phenotype Reference Batch Advanced | Statistics Information Support | Home Logout
missense nonsense splicing small deletions small insertions gross deletions complex rearrangements

Missense/nonsense : 19 mutations [back to top]

HGMD accession	HGMD codon change	HGMD amino acid change	HGVS (nucleotide)	HGVS (protein)	Variant class	Reported phenotype	Reference	Extra information
CM1410645	TCA-TGA	Ser112Term	c.335C>G	p.S112*	DM	Intellectual disability	Hamdan (2014) PLoS Genet **10**, e1004772	hg38 hg19 dbSNP
CM1314729	CGA-TGA	Arg121Term	c.361C>T	p.R121*	DM	Epileptic encephalopathy	Carvill (2013) Nat Genet **45**, 825	hg38 hg19 CpG dbSNP
CM153783	CCC-CTC	Pro218Leu	c.653C>T	p.P218L	DM?	Eyelid myoclonia with absences	Galizia (2015) Brain **138**, 1198	hg38 hg19 COM
CM1311116	CGA-TGA	Arg466Term	c.1396C>T	p.R466*	DM	Dravet syndrome-like myoclonic epileptic encephalopathy	Suls (2013) Am J Hum Genet **93**, 967	hg38 hg19 CpG dbSNP
CM1314730	CTG-CCG	Leu823Pro	c.2468T>C	p.L823P	DM	Epileptic encephalopathy	Carvill (2013) Nat Genet **45**, 825	hg38 hg19 dbSNP
CM125579	GAT-GGT	Asp856Gly	c.2567A>G	p.D856G	DM?	Autism	Neale (2012) Nature **4855**, 242 Uddin (2014) *Nat Genet* **46**: 742 [Additional report]	hg38 hg19 dbSNP
CM177698	CAA-TAA	Gln906Term	c.2716C>T	p.Q906*	DM?	Autism spectrum disorder	Yuen (2017) Nat Neurosci **20**, 602	hg38 hg19 COM
CM1617343	CGG-CAG	Arg1000Gln	c.2999G>A	p.R1000Q	DM?	Autism spectrum disorder	Wang (2016) Nat Commun **7**, 13316	hg38 hg19 COM CpG
CM1617344	GGT-GAT	Gly1174Asp	c.3521G>A	p.G1174D	DM?	Autism spectrum disorder	Wang (2016) Nat Commun **7**, 13316	hg38 hg19 COM
CM153791	CGA-GGA	Arg1313Gly	c.3937C>G	p.R1313G	DM	Epilepsy, photosensitive	Galizia (2015) Brain **138**, 1198	hg38 hg19
CM153778	CGG-CAG	Arg1345Gln	c.4034G>A	p.R1345Q	DM	Epilepsy, juvenile myoclonic	Galizia (2015) Brain **138**, 1198	hg38 hg19 CpG
CM177497	AGT-GGT	Ser1406Gly	c.4216A>G	p.S1406G	DM?	Autism spectrum disorder	Yuen (2017) Nat Neurosci **20**, 602	hg38 hg19 COM
CM153792	GTG-ATG	Val1479Met	c.4435G>A	p.V1479M	DM	Photoparoxysmal response with febrile seizures	Galizia (2015) Brain **138**, 1198	hg38 hg19 CpG dbSNP
CM162166	TGG-TGT	Trp1534Cys	c.4602G>T	p.W1534C	DM?	Epileptic encephalopathy, early onset	Helbig (2016) Genet Med **Epub**, Epub	hg38 hg19 COM
CM153777	CGA-TGA	Arg1637Term	c.4909C>T	p.R1637*	DM	Epilepsy, idiopathic photosensitive occipital	Galizia (2015) Brain **138**, 1198 O'Roak (2014) *Nat Commun* **5**: 5595 [Additional phenotype]	hg38 hg19 CpG dbSNP
CM1414846	CAG-TAG	Gln1641Term	c.4921C>T	p.Q1641*	DM?	Autism	O'Roak (2014) Nat Commun **5**, 5595	hg38 hg19 dbSNP

Figure 9: HGMD Webpage for Missense



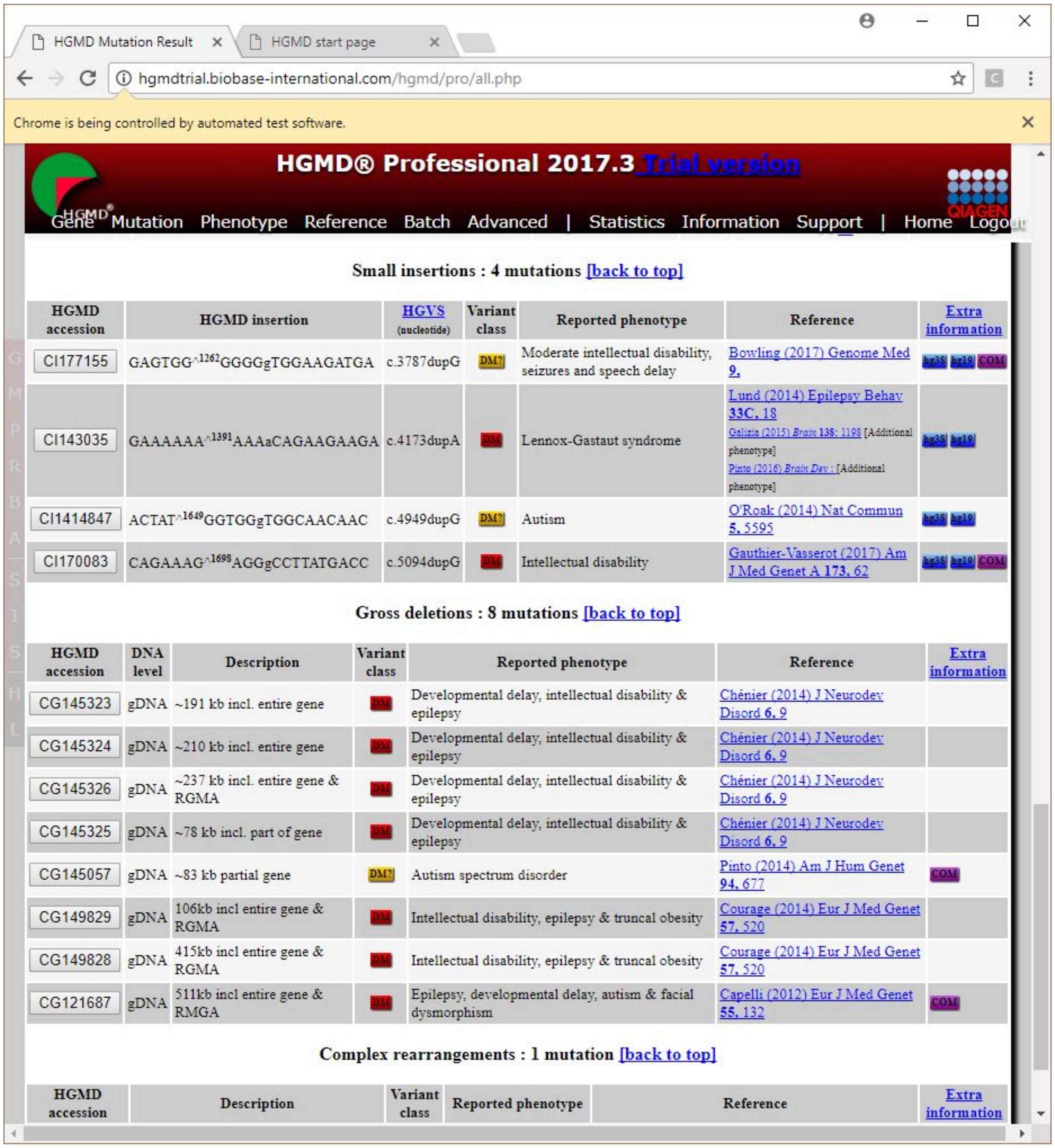
Splicing : 4 mutations [back to top]

HGMD accession	HGMD splicing mutation	HGVS (nucleotide)	Variant class	Reported phenotype	Reference	Extra information
CS149377	IVS3 as C-T +9	c.390C>T	DM?	Lennox-Gastaut syndrome	Appenzeller (2014) Am J Hum Genet **95**, 360	hg38 hg19 COM CpG
CS1314621	IVS12 ds G-A +1	c.1502+1G>A	DM	Lennox-Gastaut syndrome	Allen (2013) Nature **501**, 217	hg38 hg19 COM dbSNP
CS153788	IVS13 ds G-A +5	c.1719+5G>A	DM	Epilepsy, photosensitive	Galizia (2015) Brain **138**, 1198	hg38 hg19
CS1311117	IVS14 as A-C -2	c.1810-2A>C	DM	Dravet syndrome-like myoclonic epileptic encephalopathy	Suls (2013) Am J Hum Genet **93**, 967	hg38 hg19 dbSNP

Small deletions : 7 mutations [back to top]

HGMD accession	HGMD deletion	HGVS (nucleotide)	Variant class	Reported phenotype	Reference	Extra information
CD177189	GAAAG^^{516}ACCATcCAGACCATAT	c.1552delC	DM?	Moderate intellectual disability, autism spectrum disorder and obsessive compulsive disorder	Bowling (2017) Genome Med **9**,	hg38 hg19 COM
CD129351	GAAAGAT^^{603}AAG_E15I15_gTGTGTAATTA	c.1809+1delG	DM?	Intellectual disability, nonsyndromic	Rauch (2012) Lancet **380**, 1674 *O'Roak (2014) Nat Commun 5: 5595* [Additional phenotype]	hg38 hg19 COM dbSNP
CD1613710	AATGAT^^{626}GACTctttATTGTATAAA	c.1880_1883delCTTT	DM	Epileptic encephalopathy	Møller (2016) Mol Syndromol **7**, 210	hg38 hg19
CD1414848	CTTTT^^{964}AATAAagaaGAGCTGACAG	c.2895_2898delAGAA	DM?	Autism	O'Roak (2014) Nat Commun **5**, 5595	hg38 hg19 COM
					Bi (2012)	

Figure 10: HGMD Webpage for Splicing and Deletions



HGMD Mutation Result
HGMD start page
hgmdtrial.biobase-international.com/hgmd/pro/all.php
Chrome is being controlled by automated test software.
HGMD® Professional 2017.3 Trial version
Gene Mutation Phenotype Reference Batch Advanced | Statistics Information Support | Home Logout

Small insertions : 4 mutations [back to top]

HGMD accession	HGMD insertion	HGVS (nucleotide)	Variant class	Reported phenotype	Reference	Extra information
CI177155	GAGTGG^^{1262}GGGGgTGGAAGATGA	c.3787dupG	DM?	Moderate intellectual disability, seizures and speech delay	Bowling (2017) Genome Med 9,	hg38 hg19 COM
CI143035	GAAAAAA^^{1391}AAAaCAGAAGAAGA	c.4173dupA	DM	Lennox-Gastaut syndrome	Lund (2014) Epilepsy Behav 33C, 18 Galizia (2015) Brain 138: 1198 [Additional phenotype] Pinto (2016) Brain Dev : [Additional phenotype]	hg38 hg19
CI1414847	ACTAT^^{1649}GGTGGgTGGCAACAAC	c.4949dupG	DM?	Autism	O'Roak (2014) Nat Commun 5, 5595	hg38 hg19
CI170083	CAGAAAG^^{1698}AGGgCCTTATGACC	c.5094dupG	DM	Intellectual disability	Gauthier-Vasserot (2017) Am J Med Genet A 173, 62	hg38 hg19 COM

Gross deletions : 8 mutations [back to top]

HGMD accession	DNA level	Description	Variant class	Reported phenotype	Reference	Extra information
CG145323	gDNA	~191 kb incl. entire gene	DM	Developmental delay, intellectual disability & epilepsy	Chénier (2014) J Neurodev Disord 6, 9	
CG145324	gDNA	~210 kb incl. entire gene	DM	Developmental delay, intellectual disability & epilepsy	Chénier (2014) J Neurodev Disord 6, 9	
CG145326	gDNA	~237 kb incl. entire gene & RGMA	DM	Developmental delay, intellectual disability & epilepsy	Chénier (2014) J Neurodev Disord 6, 9	
CG145325	gDNA	~78 kb incl. part of gene	DM	Developmental delay, intellectual disability & epilepsy	Chénier (2014) J Neurodev Disord 6, 9	
CG145057	gDNA	~83 kb partial gene	DM?	Autism spectrum disorder	Pinto (2014) Am J Hum Genet 94, 677	COM
CG149829	gDNA	106kb incl entire gene & RGMA	DM	Intellectual disability, epilepsy & truncal obesity	Courage (2014) Eur J Med Genet 57, 520	
CG149828	gDNA	415kb incl entire gene & RGMA	DM	Intellectual disability, epilepsy & truncal obesity	Courage (2014) Eur J Med Genet 57, 520	
CG121687	gDNA	511kb incl entire gene & RMGA	DM	Epilepsy, developmental delay, autism & facial dysmorphism	Capelli (2012) Eur J Med Genet 55, 132	COM

Complex rearrangements : 1 mutation [back to top]

HGMD accession	Description	Variant class	Reported phenotype	Reference	Extra information

Figure 11: HGMD Webpage for Insertions

```r
# clears the console in RStudio
cat("\014")
```

```r
# clears environment
rm(list = ls())

# Load required packages
# For scraping and shaping the data
library(rvest)
library(RSelenium)
library(plyr)

# For analysis
require(ggplot2)
require(cluster)
require(amap)
require(useful)
```

6.1 Scraping Websites with RSelenium

This section shows the code that was used to scrape the websites. Unfortunately, I only had a week trial to HGMD website so my username and password will no longer work. To work around this, I scraped and saved the HGMD tables as CSV files and this code will open those up instead of scraping the website.

To start, the user needs to enter a gene to search, since I no longer have access to HGMD, this code will work with genes CHD2 and HCN2 with the CSV files provide:

```r
# Enter the gene to search
gene <- "CHD2"
```

Once the gene is entered, RSelenium will start. I want to add that there are a couple of 'if' statements that are in the code. The first checks of the gene entered is an actual gene, if it is not it lets the user know that it is not a valid gene and stops the program. The next if statement checks to see if the user name and password are valid for the HGMD website, if they are not, the code will load csv files that are provided with the code, if the username and password are good, RSelenium will work as intended.

```r
# From https://stackoverflow.com/questions/42468831/how-to-set-up-rselenium-for-r
# This runs a selenium server with chrome browser, wait for necessary files to
# download
rD <- rsDriver()
```

```
## checking Selenium Server versions:

## BEGIN: PREDOWNLOAD

## BEGIN: DOWNLOAD

## BEGIN: POSTDOWNLOAD

## checking chromedriver versions:

## BEGIN: PREDOWNLOAD

## BEGIN: DOWNLOAD

## BEGIN: POSTDOWNLOAD

## checking geckodriver versions:
```

```
## BEGIN: PREDOWNLOAD

## BEGIN: DOWNLOAD

## BEGIN: POSTDOWNLOAD

## checking phantomjs versions:

## BEGIN: PREDOWNLOAD

## BEGIN: DOWNLOAD

## BEGIN: POSTDOWNLOAD

## [1] "Connecting to remote server"
## $applicationCacheEnabled
## [1] FALSE
##
## $rotatable
## [1] FALSE
##
## $mobileEmulationEnabled
## [1] FALSE
##
## $networkConnectionEnabled
## [1] FALSE
##
## $chrome
## $chrome$chromedriverVersion
## [1] "2.33.506120 (e3e53437346286c0bc2d2dc9aa4915ba81d9023f)"
##
## $chrome$userDataDir
## [1] "C:\\Users\\Josh\\AppData\\Local\\Temp\\scoped_dir25860_21034"
##
##
## $takesHeapSnapshot
## [1] TRUE
##
## $pageLoadStrategy
## [1] "normal"
##
## $databaseEnabled
## [1] FALSE
##
## $handlesAlerts
## [1] TRUE
##
## $hasTouchScreen
## [1] FALSE
##
## $version
## [1] "62.0.3202.94"
##
## $platform
## [1] "Windows NT"
##
## $browserConnectionEnabled
## [1] FALSE
##
## $nativeEvents
## [1] TRUE
##
## $acceptSslCerts
## [1] TRUE
##
## $locationContextEnabled
## [1] TRUE
##
## $webStorageEnabled
## [1] TRUE
##
## $browserName
## [1] "chrome"
##
## $takesScreenshot
## [1] TRUE
##
## $javascriptEnabled
## [1] TRUE
##
## $cssSelectorsEnabled
## [1] TRUE
##
## $setWindowRect
## [1] TRUE
##
## $unexpectedAlertBehaviour
## [1] ""
##
## $id
## [1] "1062828490c610cd82e86c61daa9671b"
```

```
# puts a variable to the session
remDr <- rD$client

# Search ExAC

# URL for ExAC
exac_url <- "http://exac.broadinstitute.org/"

# This goes to the exac website
remDr$navigate(exac_url)

# This enters the gene and hits enter. This takes the user to the gene site
# \uE007 simulates "enter".
webElem <-
  remDr$findElement(using = 'css selector', "#home-searchbox-input")
webElem$sendKeysToElement(list(gene, "\uE007"))

# This extracts the contents of the website
exac_html <- read_html(remDr$getPageSource()[[1]])
```

```r
# This uses rvest to get the table from Selenium
exac <- exac_html %>%
  html_nodes("#variant_table") %>%
  html_table
dfExac = as.data.frame(exac)

# This makes sure a valid entry was made using an if statement.
# If a valid gene is not entered it ends the program and lets the
# user know that it was not a valid gene.
if (length(dfExac) == 0) {
  # If a valid name was not entered then the program stops with this message.
  paste("This is not a valid gene, please try again")

  # close Selenium
  remDr$close()

} else {
  # If a valid gene was entered, then the program will move to the HGMD website

  # Search HGMD

  # Username for HGMD
  username <- "joshua_husk"

  # Password for HGMD
  password <- "trial"

  # URL for HGMD
  HGMD_url <-
    "https://portal.biobase-international.com/cgi-bin/portal/login.cgi"

  # This goes to the HGMD website
  remDr$navigate(HGMD_url)

  # This enters the Username
  webElem <-
    remDr$findElement(
      using = 'css selector',
      "#login_form > table:nth-child(1) > tbody:nth-child(1) > tr:nth-child(2) > td:nth-child(1) > label
    )
  webElem$sendKeysToElement(list(username))

  # This enters the password and hits enter
  webElem <-
    remDr$findElement(
      using = 'css selector',
      "#login_form > table:nth-child(1) > tbody:nth-child(1) > tr:nth-child(4) > td:nth-child(1) > label
    )
  webElem$sendKeysToElement(list(password, "\uE007"))

  exac_html <- read_html(remDr$getPageSource()[[1]])
```

```r
# This section of the code checks to see if the username and password are correct

# This will give 'Error! No subscription' if the subscription is expired
checkOutput <- exac_html %>%
  html_nodes("#login_form > table > tbody > tr:nth-child(8) > td > span") %>%
  html_text()

# This will give 'Error: A wrong username or password was entered.' Or 'Error:
# Subscription has expired. 'if the wrong info was entered
checkOutput2 <- exac_html %>%
  html_nodes("#login_form > table > tbody > tr:nth-child(8) > td > span > font") %>%
  html_text()

# If password and user name are bad this will load the excel files
# If the password and user name are good, then it will scrape the website

if (checkOutput == "Error! No subscription" |
    checkOutput2 == "Error: A wrong username or password was entered. " |
    checkOutput2 == "Error: Subscription has expired.") {
  # close Selenium
  remDr$close()

  # This will open the CSV files after it checks if it is empty, if the files
  # are empty then it sets it to zero

  if (file.info(paste("dfHGMDMissense_", gene, ".csv", sep = ""))$size > 5) {
    dfHGMDMissense <- read.csv2(
      paste("dfHGMDMissense_", gene, ".csv", sep = ""),
      sep = ",",
      stringsAsFactors = FALSE
    )
  } else {
    dfHGMDMissense <- 0
  }

  if (file.info(paste("dfHGMDSplice_", gene, ".csv", sep = ""))$size > 5) {
    dfHGMDSplice <- read.csv2(
      paste("dfHGMDSplice_", gene, ".csv", sep = ""),
      sep = ",",
      stringsAsFactors = FALSE
    )
  } else {
    dfHGMDSplice <- 0
  }


  if (file.info(paste("dfHGMDDelete_", gene, ".csv", sep = ""))$size > 5) {
    dfHGMDDelete <- read.csv2(
      paste("dfHGMDDelete_", gene, ".csv", sep = ""),
      sep = ",",
      stringsAsFactors = FALSE
    )
  } else {
    dfHGMDDelete <- 0
  }


  if (file.info(paste("dfHGMDInsert_", gene, ".csv", sep = ""))$size > 5) {
    dfHGMDInsert <- read.csv2(
      paste("dfHGMDInsert_", gene, ".csv", sep = ""),
      sep = ",",
      stringsAsFactors = FALSE
    )
  } else {
    dfHGMDInsert <- 0
  }

} else {
  # This will scrape the website
  # This finds the link to the database
  webElem <-
    remDr$findElement(
      using = 'css selector',
      "#form-template > tbody > tr:nth-child(2) > td > div > table > tbody > tr:nth-child(2) > td > a
    )
  # click the search link
  webElem$clickElement()

  # An authentication window pops up, to get around this
  # I need to switch to the main window
  # get all windows
  windows <- remDr$getWindowHandles()

  # Swith to main window
  remDr$switchToWindow(windows[[1]])

  # This enters the username and password into the url
  HGMD_url2 <-
    paste(
      "http://",
      username,
      ":",
      password,
      "@hgmdtrial.biobase-international.com/hgmd/pro/start.php",
      sep = ""
    )

  # This goes to the HGMD website
  remDr$navigate(HGMD_url2)

  # Swith to other window
  remDr$switchToWindow(windows[[2]])

  # Close the window
  remDr$closeWindow()
```

```
# Swith to main window
remDr$switchToWindow(windows[[1]])

# This finds the gene link
webElem <-
  remDr$findElement(using = 'css selector',
                    "body > div.top > div.links > a:nth-child(1)")

# click the search link
webElem$clickElement()

# This enters the gene and hits enter
webElem <-
  remDr$findElement(
    using = 'css selector',
    'body > div.content > form:nth-child(2) > table > tbody > tr > td:nth-child(1) > input[type="te
  )
webElem$sendKeysToElement(list(gene, "\uE007"))

# This finds the mutation link
webElem <-
  remDr$findElement(
    using = 'css selector',
    'body > div.content > form > table:nth-child(5) > tbody > tr:nth-child(12) > td:nth-child(3) > 
  )

# click the search link
webElem$clickElement()

# This extracts the contents of the website
HGMD_html <- read_html(remDr$getPageSource()[[1]])

# This uses rvest to get the missense table from Selenium
HGMDMissense <- HGMD_html %>%
  html_nodes("body > div.content > table:nth-child(6)") %>%
  html_table
dfHGMDMissense = as.data.frame(HGMDMissense)
colnames(dfHGMDMissense)[3] <- "Consequence"

# This uses rvest to get thesplicing  table from Selenium
HGMDSplice <- HGMD_html %>%
  html_nodes("body > div.content > table:nth-child(9)") %>%
  html_table
dfHGMDSplice = as.data.frame(HGMDSplice)
colnames(dfHGMDSplice)[3] <- "Consequence"

# This uses rvest to get thesplicing  table from Selenium
HGMDDelete <- HGMD_html %>%
  html_nodes("body > div.content > table:nth-child(12)") %>%
  html_table
dfHGMDDelete = as.data.frame(HGMDDelete)
colnames(dfHGMDDelete)[3] <- "Consequence"
```

```
    # This uses rvest to get the inserts  table from Selenium
    HGMDInsert <- HGMD_html %>%
      html_nodes("body > div.content > table:nth-child(15)") %>%
      html_table
    dfHGMDInsert = as.data.frame(HGMDInsert)
    colnames(dfHGMDInsert)[3] <- "Consequence"

    # close Selenium
    remDr$close()
  }
}
```

6.2 Shaping the Data

Now that the data has been collected, it is time to compare the two and remove the duplicates and keep what is remaining of the HGMD data. I also removed some columns that are not needed that does not contain pertinent information

6.2.1 Remove Duplicates

Bfore removing the duplicates, the ExAC data needs to be reformated so that the variants are esier to read and the Consequence column will match the Consequence column in the HGMD data:

```
# This section removes duplicates and prints out the unique file,
# This also prints out the duplicates in a seperate file for reference

# formate the ExAC data to make it easier to read and compatible with HGMD

# THe headers were stored in row one, this makes row one the header and then deletes
# the row
colnames(dfExac) <- dfExac[1, ]
dfExac <- dfExac[-1, ]

# This reformats the Variant data, there was a bunch of junk that was scrapped
# with it.
# Remove whitespace
dfExac$Variant <- gsub(" ", "", dfExac$Variant, fixed = TRUE)

# Remove extra characters
dfExac$Variant <- gsub("\n", "", dfExac$Variant, fixed = TRUE)

# This makes the consequence column compatible with Exac by removing the p.
dfExac$Consequence <-
  gsub("p.", "", dfExac$Consequence, fixed = TRUE)

# We only need data that has an entry in the Consequence column, any blank data
# can be removed, to do this, NA is added to the Consequence column and then all
# NA's are removed.
dfExac$Consequence[dfExac$Consequence == ""] <- NA
dfExac_sub <- na.omit(dfExac)
```

6.2.2 Adding columns and getting the final output

Once the ExAC data is formatted, then the data can be compared. This code will check to make sure there is a data frame and then compare the data and remove all duplicates then print it to a CSV file (HGMD_final) it also prints the duplicates to a separate file for reference.

6.2.2.1 Using regex to make new columns

This code also searches the reference info and pulls out the first publish date of the mutation and puts it into a new column called year.

I also made a column for mutation type where the data is split up between missense, nonsense, splice, frameshift, or nonframeshift.

To find missense and nonsense data I used grep. I started by finding all nonsense data and using an if statement to add nonsense or missense to the mutation type column:

```
# Find nonsense
nonsense <-
  grep(".+Term$",
       dfHGMDMissense_final$Consequence,
       perl = TRUE,
       value = TRUE)

# Make new column
dfHGMDMissense_final$Mutation.type <-
  ifelse(dfHGMDMissense_final$Consequence == nonsense,
         "Nonsense",
         "Missense")
```

For splicing, if it was from the splice table I added a mutation type column with the word splice:

```
# Create mutation type column
dfHGMDSplice_final$Mutation.type <-
  rep("splice", nrow(dfHGMDSplice_final))
```

If the splice was in an insertion or deletion section then I used:

```
gsub(pattern = ".+\\+.+|.+\\-.+",
     dfHGMDDelete_final_1$Mutation.type,
     replacement = "splice")
```

To find frameshifts and nonframeshifts I checked to see if the number of insertions or deletions were a multiple of three. This took many steps and is outlined in the code below.

Once I had the mutation type, I could tell if it caused a loss of function or not. If it is a missense or nonframeshift, then it is “not a loss of function” (nLoF) otherwise it is a “loss of function” (LoF).

```
# Compare the data

# This makes sure there is data in the dataframe before anlaying,
# If there is no data it prints a statem that there is no data
# and moves to the next data set.
if (exists("dfHGMDMissense") &&
    is.data.frame(get("dfHGMDMissense")) == 'FALSE') {
  paste("There is no data to analyze for HGMD missense")
} else {
  # Removing duplicates for Missens
  # This combines the data frames by using plyr
```

```r
df1_mis <- join(dfHGMDMissense, dfExac, type = "full")

# This removes the duplicates
df1_mis <-
  df1_mis[!(
    duplicated(df1_mis$Consequence) |
      duplicated(df1_mis$Consequence, fromLast = TRUE)
  ), ]

# This keeps the rows needed
keep <-
  c(
    "Consequence",
    "HGMD.codonchange",
    "HGVS.nucleotide.",
    "HGVS.protein.",
    "Variantclass",
    "Reported.phenotype",
    "Reference"
  )
df2_mis <- df1_mis[keep]

# This omits all na's and gives the final value with no duplicates
dfHGMDMissense_final <- na.omit(df2_mis)

# Add columns for additional analysis
# Extract publish year from reference and make new column
dfHGMDMissense_final$Year <-
  as.numeric(sub(
    "\\D*\\((\\d\\d\\d\\d)\\).*",
    "\\1",
    dfHGMDMissense_final$Reference
  ))

# Create mutation type column
# If the ending in Consequence is Term then it is a nonsense
# Define nonsense
nonsense <-
  grep(".+Term$",
       dfHGMDMissense_final$Consequence,
       perl = TRUE,
       value = TRUE)

if (length(nonsense)==0){
  dfHGMDMissense_final$Mutation.type <-
    rep("missense", nrow(dfHGMDMissense_final))
} else {
  # Make new column
  dfHGMDMissense_final$Mutation.type <-
    ifelse(dfHGMDMissense_final$Consequence == nonsense,
           "nonsense",
           "missense")
}
```

```
  # Make overal Consequence column
  dfHGMDMissense_final$Overall.consequence <-
    ifelse(dfHGMDMissense_final$Mutation.type == "missense",
           "nLoF",
           "LoF")

  # For reference, this will get all of the duplicates
  # This collects the duplicates
  dfHGMDMissense_Duplicate <-
    merge(dfExac, dfHGMDMissense, by = "Consequence", all = FALSE)

  # remove data frames and other objects no longer needed
  rm("df1_mis", "df2_mis", "keep", "nonsense")

}
```

```
## Joining by: Consequence, Reference
```

```
# This makes sure there is data in the dataframe before anlaying,
# If there is no data it prints a statem that there is no data
# and moves to the next data set.
if (exists("dfHGMDSplice") &&
    is.data.frame(get("dfHGMDSplice")) == 'FALSE') {
  paste("There is no data to analyze for HGMD Splices")
} else {
  # Removing duplicates for Splice

  # This combines the data frames
  df1_spl <- join(dfHGMDSplice, dfExac, type = "full")

  # This removes the duplicates
  df1_spl <-
    df1_spl[!(
      duplicated(df1_spl$Consequence) |
        duplicated(df1_spl$Consequence, fromLast = TRUE)
    ),]

  # This keeps the rows needed
  keep <-
    c(
      "Consequence",
      "HGMD.splicing.mutation",
      "Variantclass",
      "Reported.phenotype",
      "Reference"
    )
  df2_spl <- df1_spl[keep]

  # This omits all na's and gives the final value with no duplicates
  dfHGMDSplice_final <- na.omit(df2_spl)

  # Add columns for additional analysis
  # Extract publish year from reference and make new column
  dfHGMDSplice_final$Year <-
```

```
    as.numeric(sub(
      "\\D*\\((\\d\\d\\d\\d)\\).*",
      "\\1",
      dfHGMDSplice_final$Reference
    ))

  # Create mutation type column
  dfHGMDSplice_final$Mutation.type <-
    rep("splice", nrow(dfHGMDSplice_final))

  # Make overal Consequence column
  dfHGMDSplice_final$Overall.consequence <-
    ifelse(dfHGMDSplice_final$Mutation.type == "missense",
           "nLoF",
           "LoF")

  # For reference, this will get all of the duplicates
  # This collects the duplicates
  dfHGMDSplice_Duplicate <-
    merge(dfExac, dfHGMDSplice, by = "Consequence", all = FALSE)

  # remove data frames and other objects no longer needed
  rm("df1_spl", "df2_spl", "keep")
}
```

```
## Joining by: Consequence, Reference
```

```
# This makes sure there is data in the dataframe before anlaying,
# If there is no data it prints a statem that there is no data
# and moves to the next data set.
if (exists("dfHGMDDelete") &&
    is.data.frame(get("dfHGMDDelete")) == 'FALSE') {
  paste("There is no data to analyze for HGMD Deletions")
} else {
  # Removing duplicates for Deletions
  # This combines the data frames
  df1_del <- join(dfHGMDDelete, dfExac, type = "full")

  # This removes the duplicates
  df1_del <-
    df1_del[!(duplicated(df1_del$Consequence) |
                duplicated(df1_del$Consequence, fromLast = TRUE)),]

  # This keeps the rows needed
  keep <-
    c("Consequence",
      "HGMD.deletion",
      "Variantclass",
      "Reported.phenotype",
      "Reference")
  df2_del <- df1_del[keep]

  # This omits all na's and gives the final value with no duplicates
  dfHGMDDelete_final <- na.omit(df2_del)
```

```
#### Add columns for additional analysis ####

### Add column year ###
# Extract publish year from reference and make new column
dfHGMDDelete_final$Year <-
  as.numeric(sub(
    "\\D*\\((\\d\\d\\d\\d)\\).*",
    "\\1",
    dfHGMDDelete_final$Reference
  ))

### Add Mutation.type coumn ###

# make a new dataframe to find mutation types
id <- 1:nrow(dfHGMDDelete_final)
Mutation.type <- dfHGMDDelete_final$Consequence
dfHGMDDelete_final_1 <-
  data.frame(id, Mutation.type)


# If it has a plus or minus it is a splice
dfHGMDDelete_final_1$Mutation.type <-
  gsub(pattern = ".+\\+.+|.+\\-.+",
       dfHGMDDelete_final_1$Mutation.type,
       replacement = "splice")

# Extract number
# If it ends in two numbers
dfHGMDDelete_final_1$Mutation.type <-
  gsub(pattern = "c\\..+(\\d\\d)$",
       dfHGMDDelete_final_1$Mutation.type,
       replacement = "\\1")

# If it ends in one number
dfHGMDDelete_final_1$Mutation.type <-
  gsub(pattern = "c\\..+(\\d)$",
       dfHGMDDelete_final_1$Mutation.type,
       replacement = "\\1")

# change to int
dfHGMDDelete_final_1$Mutation.type2 <-
  as.integer(dfHGMDDelete_final_1$Mutation.type)

# If multiple of 3 it is a non frameshift
dfHGMDDelete_final_1$Mutation.type2 <-
  ifelse(dfHGMDDelete_final_1$Mutation.type2 %% 3 == 0,
         "nonframeshift",
         "frameshift")

# keep splice
dfHGMDDelete_final_1$Mutation.type3 <-
  ifelse(dfHGMDDelete_final_1$Mutation.type == "splice",
         "splice",
           NA)
  dfHGMDDelete_final_1

  # Count number of deletions
  # Make new column to count uppercase
  dfHGMDDelete_final_1$Mutation.type4 <-
    sapply(regmatches(
      dfHGMDDelete_final_1$Mutation.type,
      gregexpr("[A-Z]", dfHGMDDelete_final_1$Mutation.type, perl = TRUE)
    ),
    length)

  # replace 0 with na
  dfHGMDDelete_final_1$Mutation.type4[dfHGMDDelete_final_1$Mutation.type4 == 0] <-
    NA


  # If multiple of 3 it is a non frameshift
  dfHGMDDelete_final_1$Mutation.type4 <-
    ifelse(dfHGMDDelete_final_1$Mutation.type4 %% 3 == 0,
           "nonframeshift",
           "frameshift")

  # Remove original column
  dfHGMDDelete_final_1$Mutation.type <- NULL

  # Combine all columns eliminating the na's
  dfHGMDDelete_final_1 <- apply(dfHGMDDelete_final_1,1,function(x) x[!is.na(x)])
  dfHGMDDelete_final_1 <- data.frame(t(dfHGMDDelete_final_1))
  colnames(dfHGMDDelete_final_1) <- c("id", "Mutation.type")

  # Add mutation type to original dataframe
  dfHGMDDelete_final$Mutation.type <- dfHGMDDelete_final_1$Mutation.type

  # Make overal Consequence column
  dfHGMDDelete_final$Overall.consequence <-
    ifelse(dfHGMDDelete_final$Mutation.type == "nonframeshift",
           "nLoF",
           "LoF")

  # remove data frames and other objects no longer needed
  rm(
    "dfHGMDDelete_final_1",
    "Mutation.type",
    "df1_del",
    "df2_del",
    "keep",
    "id"
  )
}
```

```
## Joining by: Consequence, Reference
```

```
# This makes sure there is data in the dataframe before anlaying,
# If there is no data it prints a statem that there is no data
# and moves to the next data set.
if (exists("dfHGMDInsert") &&
    is.data.frame(get("dfHGMDInsert")) == 'FALSE') {
  paste("There is no data to analyze for HGMD Insertions")
} else {
  # Removing duplicates for Inserts
  # This combines the data frames
  df1_ins <- join(dfHGMDInsert, dfExac, type = "full")

  # This removes the duplicates
  df1_ins <-
    df1_ins[!(
      duplicated(df1_ins$Consequence) |
        duplicated(df1_ins$Consequence, fromLast = TRUE)
    ),]

  # This keeps the rows needed
  keep <-
    c(
      "Consequence",
      "HGMD.insertion",
      "Variantclass",
      "Reported.phenotype",
      "Reference"
    )
  df2_ins <- df1_ins[keep]

  # This omits all na's and gives the final value with no duplicates
  dfHGMDInsert_final <- na.omit(df2_ins)

  #### Add columns for additional analysis ####

  ### Add column year ###
  # Extract publish year from reference and make new column
  dfHGMDInsert_final$Year <-
    as.numeric(sub(
      "\\D*\\((\\d\\d\\d\\d)\\).*",
      "\\1",
      dfHGMDInsert_final$Reference
    ))

  ### Add Mutation.type coumn ###

  # make a new dataframe to find mutation types
  id <- 1:nrow(dfHGMDInsert_final)
  Mutation.type <- dfHGMDInsert_final$Consequence
  dfHGMDInsert_final_1 <-
    data.frame(id, Mutation.type)


  # If it has a plus or minus it is a splice
```

```r
dfHGMDInsert_final_1$Mutation.type <-
  gsub(pattern = ".+\\+.+|.+\\-.+",
       dfHGMDInsert_final_1$Mutation.type,
       replacement = "splice")

# Extract number
# If it ends in two numbers
dfHGMDInsert_final_1$Mutation.type <-
  gsub(pattern = "c\\..+(\\d\\d)$",
       dfHGMDInsert_final_1$Mutation.type,
       replacement = "\\1")

# If it ends in one number
dfHGMDInsert_final_1$Mutation.type <-
  gsub(pattern = "c\\..+(\\d)$",
       dfHGMDInsert_final_1$Mutation.type,
       replacement = "\\1")

# change to int
dfHGMDInsert_final_1$Mutation.type2 <-
  as.integer(dfHGMDInsert_final_1$Mutation.type)

# If multiple of 3 it is a non frameshift
dfHGMDInsert_final_1$Mutation.type2 <-
  ifelse(dfHGMDInsert_final_1$Mutation.type2 %% 3 == 0,
         "nonframeshift",
         "frameshift")

# keep splice
dfHGMDInsert_final_1$Mutation.type3 <-
  ifelse(dfHGMDInsert_final_1$Mutation.type == "splice",
         "splice",
         NA)
dfHGMDInsert_final_1

# Count number of deletions
# Make new column to count uppercase
dfHGMDInsert_final_1$Mutation.type4 <-
  sapply(regmatches(
    dfHGMDInsert_final_1$Mutation.type,
    gregexpr("[A-Z]", dfHGMDInsert_final_1$Mutation.type, perl = TRUE)
  ),
  length)

# replace 0 with na
dfHGMDInsert_final_1$Mutation.type4[dfHGMDInsert_final_1$Mutation.type4 == 0] <-
  NA


# If multiple of 3 it is a non frameshift
dfHGMDInsert_final_1$Mutation.type4 <-
  ifelse(dfHGMDInsert_final_1$Mutation.type4 %% 3 == 0,
         "nonframeshift",
          "frameshift")

  # Remove original column
  dfHGMDInsert_final_1$Mutation.type <- NULL

  # Combine all columns eliminating the na's
  dfHGMDInsert_final_1 <- apply(dfHGMDInsert_final_1,1,function(x) x[!is.na(x)])
  dfHGMDInsert_final_1 <- data.frame(t(dfHGMDInsert_final_1))
  colnames(dfHGMDInsert_final_1) <- c("id", "Mutation.type")

  # Add mutation type to original dataframe
  dfHGMDInsert_final$Mutation.type <- dfHGMDInsert_final_1$Mutation.type

  # Make overal Consequence column
  dfHGMDInsert_final$Overall.consequence <-
    ifelse(dfHGMDInsert_final$Mutation.type == "nonframeshift",
           "nLoF",
           "LoF")

  # remove data frames and other objects no longer needed
  rm(
    "dfHGMDInsert_final_1",
    "Mutation.type",
    "df1_ins",
    "df2_ins",
    "keep",
    "id"
  )
}
```

```
## Joining by: Consequence, Reference
```

```
# This merges all non-duplicate data
# Initialize data frame
dfHGMD_final <- data.frame(stringsAsFactors=FALSE)

# Add data if exists
if (exists("dfHGMDMissense_final")) {
  dfHGMD_final <-
    join(dfHGMD_final, dfHGMDMissense_final, type = "full")
}
```

```
## Joining by:
```

```
if (exists("dfHGMDSplice_final")) {
  dfHGMD_final <-
    join(dfHGMD_final, dfHGMDSplice_final, type = "full")
}
```

```
## Joining by: Consequence, Variantclass, Reported.phenotype, Reference, Year, Mutation.type, Overall.c
```

```
if (exists("dfHGMDDelete_final")) {
  dfHGMD_final <-
    join(dfHGMD_final, dfHGMDDelete_final, type = "full")
}
```

```
## Joining by: Consequence, Variantclass, Reported.phenotype, Reference, Year, Mutation.type, Overall.c
```

```r
if (exists("dfHGMDInsert_final")) {
  dfHGMD_final <-
    join(dfHGMD_final, dfHGMDInsert_final, type = "full")
}
```

```
## Joining by: Consequence, Variantclass, Reported.phenotype, Reference, Year, Mutation.type, Overall.c
```

```r
# This checks to see if there are non-duplicates to write to a file
rowIns_final <- nrow(dfHGMD_final)

# This prints to a csv file if there are non-duplicate.
if (rowIns_final > 0) {
  write.csv(
    dfHGMD_final,
    file = paste("HGMD_final_", gene, ".csv", sep = ""),
    row.names = FALSE
  )
}


# This merges all duplicate data
# Initialize data frame
dfHGMD_Duplicate <- data.frame(stringsAsFactors=FALSE)

# Add data if exists
if (exists("dfHGMDMissense_Duplicate")) {
  dfHGMD_Duplicate <-
    join(dfHGMD_Duplicate, dfHGMDMissense_Duplicate, type = "full")
}
```

```
## Joining by:
```

```r
if (exists("dfHGMDSplice_Duplicate")) {
  dfHGMD_Duplicate <-
    join(dfHGMD_Duplicate, dfHGMDSplice_Duplicate, type = "full")
}
```

```
## Joining by: Consequence, Variant, Chrom, Position, RSID, Reference.x, Alternate, Protein Consequence
```

```r
if (exists("dfHGMDDelete_Duplicate")) {
  dfHGMD_Duplicate <-
    join(dfHGMD_Duplicate, dfHGMDDelete_Duplicate, type = "full")
}

if (exists("dfHGMDInsert_Duplicate")) {
  dfHGMD_Duplicate <-
    join(dfHGMD_Duplicate, dfHGMDInsert_Duplicate, type = "full")
}


# This checks to see if there are duplicates to write to a file
rowIns_Duplicate <- nrow(dfHGMD_Duplicate)

# This prints to a csv file if there are duplicates.
if (rowIns_Duplicate > 0) {
  write.csv(
    dfHGMD_Duplicate,
    file = paste("HGMD_Duplicate_", gene, ".csv", sep = ""),
    row.names = FALSE
  )
}
```

6.2.2.2 Verifying Results

The missense data was the only table with redundant information. To verify the results, I will use the merge function to find what is similar. Then compare before and after to make sure what was the same is now removed.

```
comparison <- merge(dfExac_sub, dfHGMDMissense, by = "Consequence", all=FALSE)
comparison$Consequence
```

```
## [1] "Arg1000Gln" "Arg1685His" "Val1479Met"
```

```
# This is the original and it contains the values
apply(dfHGMDMissense, 1, function(r)
any (r %in% c("Arg1000Gln", "Arg1685His", "Val1479Met")))
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE
## [12] FALSE  TRUE FALSE FALSE FALSE FALSE FALSE  TRUE
```

```
# This is the final and it does not contain the values
apply(dfHGMDMissense_final, 1, function(r)
any (r %in% c("Arg1000Gln", "Arg1685His", "Val1479Met")))
```

```
##     1     2     3     4     5     6     7     9    10    11    12    14
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##    15    16    17    18
## FALSE FALSE FALSE FALSE
```

6.3 Analyzing the Data

I will perform clustering to look how the data is grouped.

6.3.1 Cluster HGMD Data

I cleard the previous data and configured R with the following parameters before beginning the analysis:

```
# clears the console in RStudio
cat("\014")
```

```
# clears environment
rm(list = ls())
```

6.3.1.1 Load Data.

I opened the variant data from the gene CHD2 containing all known mutations.

To format the data, the data is separated by ‘,’, stringsAsFactors = FALSE so that the strings in a data frame will be treated as plain strings and not as factor variables. I set na strings for missing data.

Below is my R code:

```r
# Some csv files are really big and take a while to open.  This command checks to
# see if it is already opened, if it is, it does not open it again.
# I also omitted the first column
if (!exists("dfHGMD_final")) {
  dfHGMD_final <-
    read.csv2(
      "HGMD_final_CHD2.csv",
      sep = ",",
      stringsAsFactors = FALSE,
      na.strings = c("", "NA")
    )
}
```

6.3.1.2 Formating the data

Inorder to cluster the data I need numeric data. I will make new columns with Consequence, Variantclass, Reported.phenotype, Mutation.type, Overall.consequence, and year converted to numeric.

For Consequence the data will change to: Create column and assign a Consequence with a num Ser112Term = 1, Arg121Term = 2, Pro218Leu = 3, Arg466Term = 4, Leu823Pro = 5, Asp856Gly = 6, Gln906Term = 7, Gly1174Asp = 8, Arg1313Gly = 9, Arg1345Gln = 10, Ser1406Gly = 11, Trp1534Cys = 12, Arg1637Term = 13, Gln1641Term = 14, Trp1657Term = 15, Arg1679Term = 16, c.390C>T = 17, c.1502+1G>A = 18, c.1719+5G>A = 19, c.1810-2A>C = 20, c.1552delC = 21, c.1809+1delG = 22, c.1880_1883delCTTT = 23, c.2895_2898delAGAA = 24, c.3734delA = 25, c.4233_4236delAGAA = 26, c.4256_4274del19 = 27, c.3787dupG = 28, c.4173dupA = 29, c.4949dupG = 30, c.5094dupG = 31

```r
# Make Consequence numeric
dfHGMD_final$Consequence.num <- 1:nrow(dfHGMD_final)
```

For Reported Phenotype the data will change to: Autism = 1 Epilepsy = 2 Eyelid myoclonia = 3 Intellectual disability = 4

```r
# Make Reported.phenotype numeric
# Assign number to Reported.phenotype
# Create column
dfHGMD_final$Reported.phenotype.num <-
  dfHGMD_final$Reported.phenotype

# Use regex to make data numeric
# Autism = 1
dfHGMD_final$Reported.phenotype.num <-
  gsub("^Autism.*",
       dfHGMD_final$Reported.phenotype.num,
       replacement = "1")

# Epilepsy = 2
dfHGMD_final$Reported.phenotype.num <-
  gsub("^Epilep.*|^Dravet.*|^Lennox-Gastaut.*",
       dfHGMD_final$Reported.phenotype.num,
       replacement = "2")

# Eyelid myoclonia = 3
dfHGMD_final$Reported.phenotype.num <-
  gsub("^Eyelid.*",
       dfHGMD_final$Reported.phenotype.num,
       replacement = "3")

# Intellectual disability = 4
dfHGMD_final$Reported.phenotype.num <-
  gsub("^Intellectual.*|.*intellectual disability.*",
       dfHGMD_final$Reported.phenotype.num,
       replacement = "4")
```

For Variant Class the data will change to: DM = 1 DM? = 2

```r
# Make Variantclass numeric
# Assign number to Variantclass.num
dfHGMD_final$Variantclass.num<-dfHGMD_final$Variantclass
dfHGMD_final$Variantclass.num<-ifelse(dfHGMD_final$Variantclass.num=="DM", 1, 2)
```

For Overall.consequence the data will change to: LoF = 1 nLoF = 2

```r
# Make Overall.consequence numeric
# Assign number to Overall.consequence.num
dfHGMD_final$Overall.consequence.num <-
dfHGMD_final$Overall.consequence
dfHGMD_final$Overall.consequence.num <-
ifelse(dfHGMD_final$Overall.consequence.num == "LoF", 1, 2)
```

For Mutation.type the data will change to: frameshift = 1 Missense = 2 Nonsense = 3 splice = 4

```r
# Make Reported.phenotype numeric
# Assign number to Reported.phenotype
# Create column
dfHGMD_final$Mutation.type.num <-
  dfHGMD_final$Mutation.type

# Use regex to make data numeric
# frameshift = 1
dfHGMD_final$Mutation.type.num <-
  gsub("^frameshift",
       dfHGMD_final$Mutation.type.num,
       replacement = "1")

# Missense = 2
dfHGMD_final$Mutation.type.num <-
  gsub("^missense",
       dfHGMD_final$Mutation.type.num,
       replacement = "2")

# Nonsense = 3
dfHGMD_final$Mutation.type.num <-
  gsub("^nonsense",
       dfHGMD_final$Mutation.type.num,
       replacement = "3")

# Splice = 4
dfHGMD_final$Mutation.type.num <-
  gsub("^splice",
       dfHGMD_final$Mutation.type.num,
       replacement = "4")
```

```
# nonframeshift = 5
dfHGMD_final$Mutation.type.num <-
  gsub("^nonframeshift",
       dfHGMD_final$Mutation.type.num,
       replacement = "5")
```

Change data to numeric and make new dataframe:

```
# Convert to numeric
dfHGMD_final[8] <- sapply(dfHGMD_final[8], as.numeric)
dfHGMD_final[14:18] <- sapply(dfHGMD_final[14:18], as.numeric)


# Columns to keep
keep <-
  c(
    "Consequence.num",
    "Reported.phenotype.num",
    "Variantclass.num",
    "Overall.consequence.num",
    "Mutation.type.num",
    "Year")

# Make new dataframe with keep data
dfHGMDCluster <- dfHGMD_final[keep]

# Check dataframe
str(dfHGMDCluster)
```

```
## 'data.frame':    31 obs. of  6 variables:
##  $ Consequence.num        : num  1 2 3 4 5 6 7 8 9 10 ...
##  $ Reported.phenotype.num : num  4 2 3 2 2 1 1 1 2 2 ...
##  $ Variantclass.num       : num  1 1 2 1 1 2 2 2 1 1 ...
##  $ Overall.consequence.num: num  1 1 2 2 2 2 2 2 2 2 ...
##  $ Mutation.type.num      : num  3 3 2 2 2 2 2 2 2 2 ...
##  $ Year                   : num [1:31, 1] 2014 2013 2015 2013 2013 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr "Year"
```

```
summary(dfHGMDCluster)
```

```
##  Consequence.num Reported.phenotype.num Variantclass.num
##  Min.   : 1.0    Min.   :1.000          Min.   :1.000
##  1st Qu.: 8.5    1st Qu.:1.500          1st Qu.:1.000
##  Median :16.0    Median :2.000          Median :1.000
##  Mean   :16.0    Mean   :2.097          Mean   :1.484
##  3rd Qu.:23.5    3rd Qu.:2.000          3rd Qu.:2.000
##  Max.   :31.0    Max.   :4.000          Max.   :2.000
##  Overall.consequence.num Mutation.type.num      Year.Year
##  Min.   :1.000           Min.   :1.000     Min.   :2012.0000
##  1st Qu.:1.000           1st Qu.:1.000     1st Qu.:2013.0000
##  Median :1.000           Median :2.000     Median :2015.0000
##  Mean   :1.323           Mean   :2.194     Mean   :2014.5806
##  3rd Qu.:2.000           3rd Qu.:3.000     3rd Qu.:2016.0000
##  Max.   :2.000           Max.   :4.000     Max.   :2017.0000
```

6.3.2 Clustering

Clustering is grouping like with like such that:

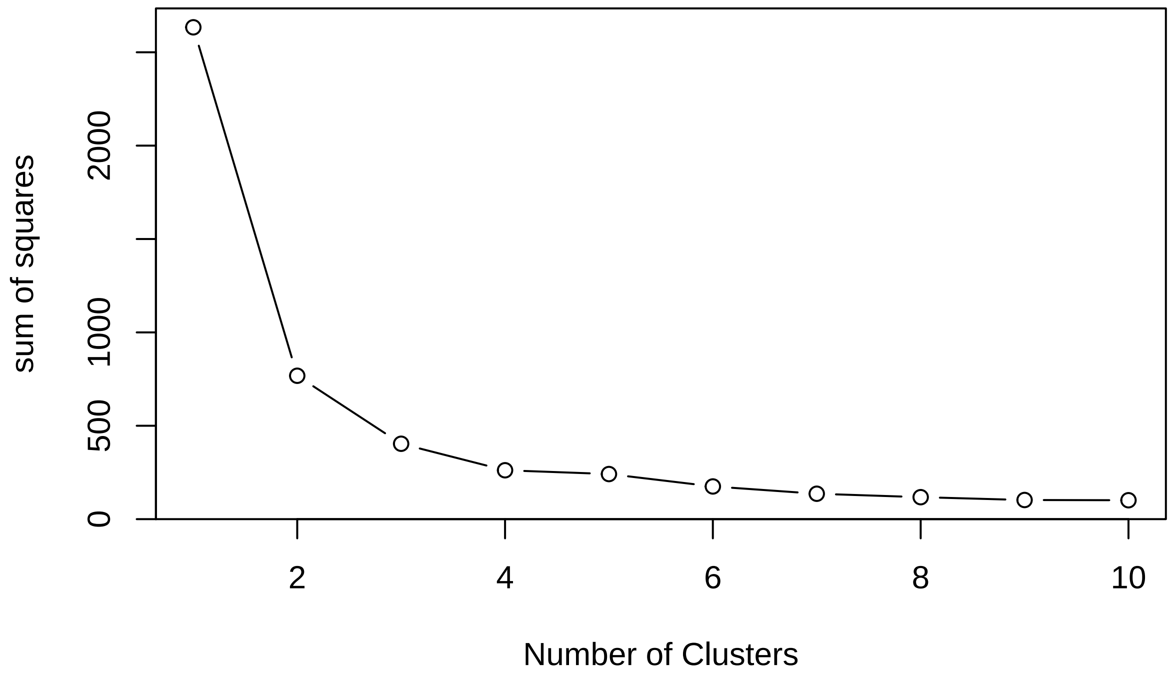
1. Similar objects are close to one another within the same cluster.
2. Dissimilar to the objects in other clusters.

6.3.2.1 Number of Clusters

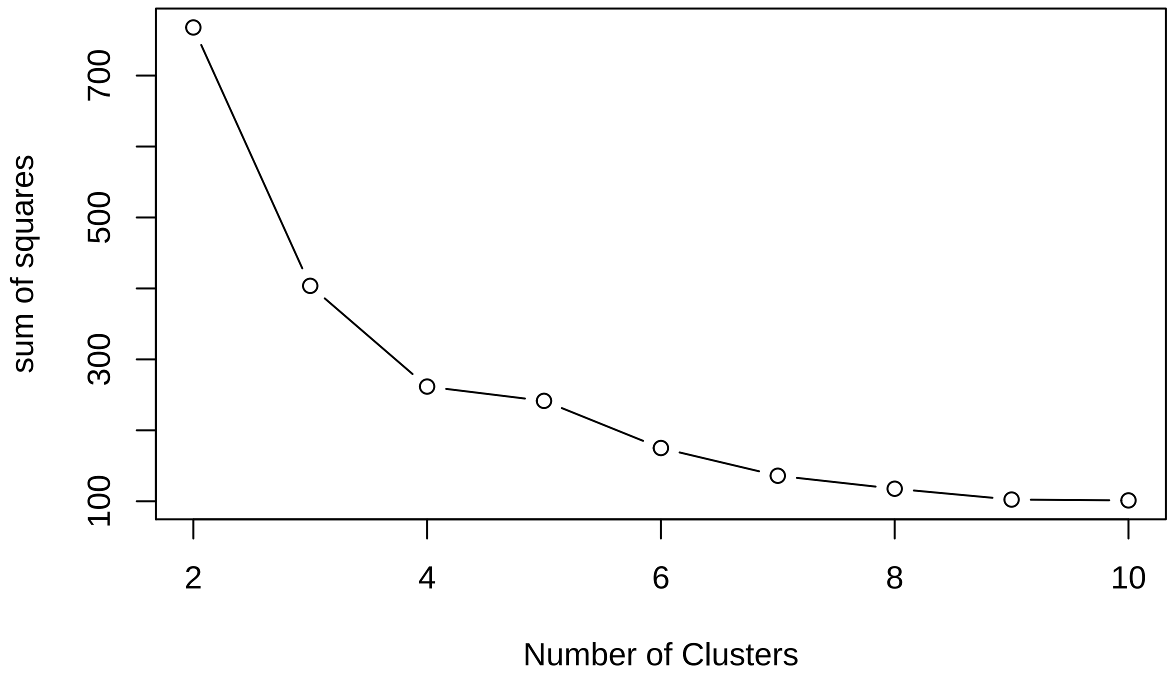
Before I can begin clustering analysis, I need to determine the number of clusters. For determining "the right number of clusters", for this analysis I will use the averaged Silhouette width and Gap statistic and Hartigan's rule.

6.3.2.1.1 Averaged Silhouette Width and Gap Statistic

```r
# Determining number of clusters
sos <- (nrow(dfHGMDCluster) - 1) * sum(apply(dfHGMDCluster, 2, var))
for (i in 2:10)
  sos[i] <- sum(kmeans(dfHGMDCluster, centers = i)$withinss)
plot(1:10,
     sos,
     type = "b",
     xlab = "Number of Clusters",
     ylab = "sum of squares")
```

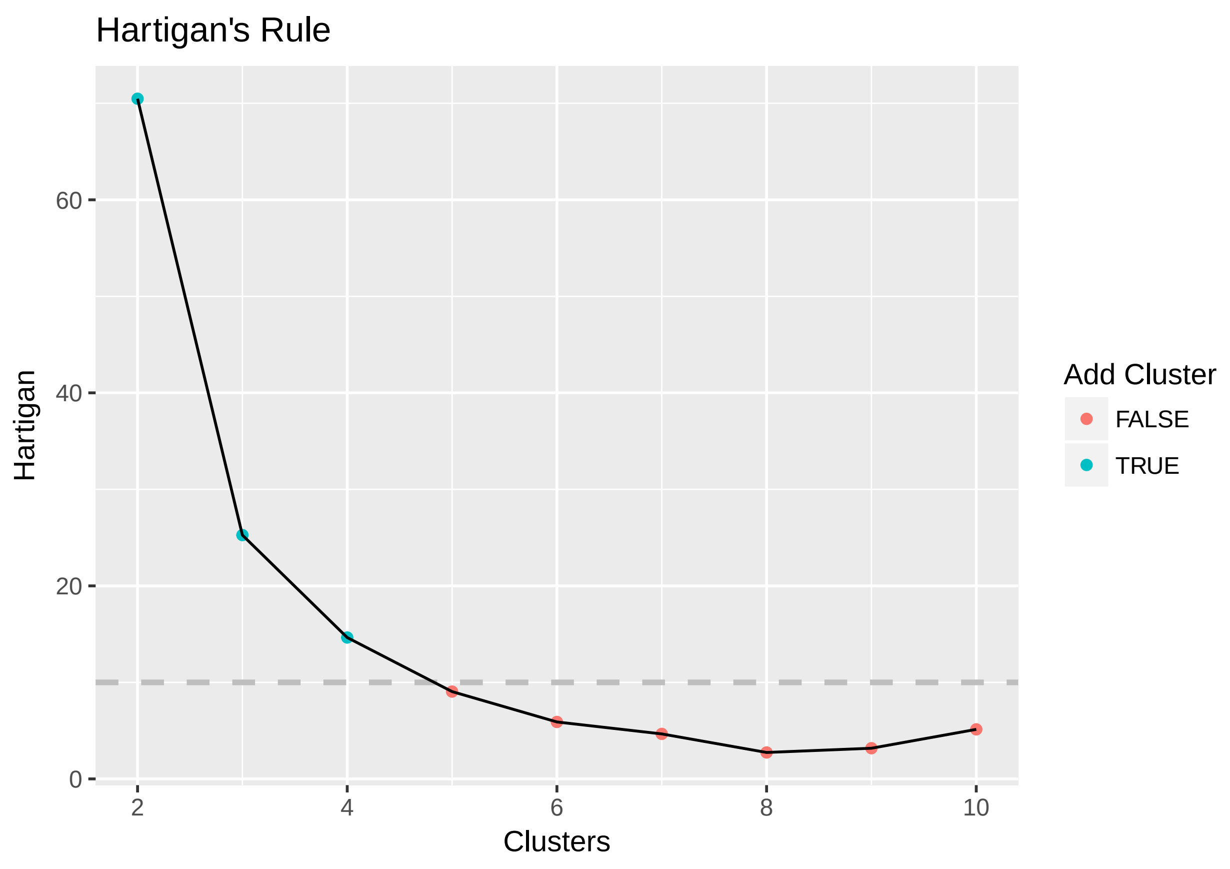


```
plot(2:10,
     sos[c(2:10)],
     type = "b",
     xlab = "Number of Clusters",
     ylab = "sum of squares")
```



6.3.2.1.2 Hartigan's rule

```
# Hartigans's rule FitKMean (similarity)
# require(useful)
best<-FitKMeans(dfHGMDCluster,max.clusters=10, seed=111)
PlotHartigan(best)
```



6.3.2.1.3 Number of Clusters Results

Based off of the graphs above it looks like I should perform clustering with 2 to 4 clusters. The analysis looks better with 2 after some trial and error.

6.3.2.2 Partitioning-based clustering

Partitioning algorithms construct various partitions and then evaluate them by some criterion Hierarchy algorithms, two examples are k-means and k-mediods algorithms.

6.3.2.2.1 k-means

k-means clustering is a method of vector quantization, originally from signal processing. k-means clustering aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster. This results in a partitioning of the data space into Voronoi cells.

Below I apply the k-means algorithm with a calculation that makes the k-means calculation more stable, it performs this analysis 1000 times and takes the ones with the least error:

```r
# Clustering with 2 clusters
k <- 2
trails<-1000
dfHGMDCluster.2.cluster <- kmeans(dfHGMDCluster,k, nstart = trails)
dfHGMDCluster.2.cluster
```

```
## K-means clustering with 2 clusters of sizes 16, 15
##
## Cluster means:
##   Consequence.num Reported.phenotype.num Variantclass.num
## 1             8.5               1.875000         1.500000
## 2            24.0               2.333333         1.466667
##   Overall.consequence.num Mutation.type.num     Year
## 1                   1.625             2.375 2014.625
## 2                   1.000             2.000 2014.533
##
## Clustering vector:
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##
## Within cluster sum of squares by cluster:
## [1] 397.0 370.8
##  (between_SS / total_SS =  70.8 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

Evaluating model performance

```r
# Evaluating model performance
# look at the size of the clusters
dfHGMDCluster.2.cluster$size
```

```
## [1] 16 15
```

```r
# look at the cluster centers
dfHGMDCluster.2.cluster$centers
```

```
##   Consequence.num Reported.phenotype.num Variantclass.num
## 1             8.5               1.875000         1.500000
## 2            24.0               2.333333         1.466667
##   Overall.consequence.num Mutation.type.num     Year
## 1                   1.625             2.375 2014.625
## 2                   1.000             2.000 2014.533
```

```r
names(dfHGMDCluster)
```

```
## [1] "Consequence.num"         "Reported.phenotype.num"
## [3] "Variantclass.num"        "Overall.consequence.num"
## [5] "Mutation.type.num"       "Year"
```

```r
# mean of 'Consequence.num' by cluster
Consequence <-
  aggregate(data = dfHGMDCluster, Consequence.num ~ dfHGMDCluster.2.cluster$cluster, mean)
Consequence
```

```
##   dfHGMDCluster.2.cluster$cluster Consequence.num
## 1                               1             8.5
## 2                               2            24.0
```

```r
# mean of 'Reported.phenotype.num' by cluster
Reported.phenotype <-
```

```
  aggregate(data = dfHGMDCluster,
            Reported.phenotype.num ~ dfHGMDCluster.2.cluster$cluster,
            mean)
Reported.phenotype
```

```
##   dfHGMDCluster.2.cluster$cluster Reported.phenotype.num
## 1                               1               1.875000
## 2                               2               2.333333
```

```
# mean of 'Variantclass' by cluster
Variantclass <-
  aggregate(data = dfHGMDCluster,
            Variantclass.num ~ dfHGMDCluster.2.cluster$cluster,
            mean)
Variantclass
```

```
##   dfHGMDCluster.2.cluster$cluster Variantclass.num
## 1                               1         1.500000
## 2                               2         1.466667
```

```
# mean 'year'  by cluster
year <-
  aggregate(data = dfHGMDCluster, Year ~ dfHGMDCluster.2.cluster$cluster, mean)
year
```

```
##   dfHGMDCluster.2.cluster$cluster     Year
## 1                               1 2014.625
## 2                               2 2014.533
```

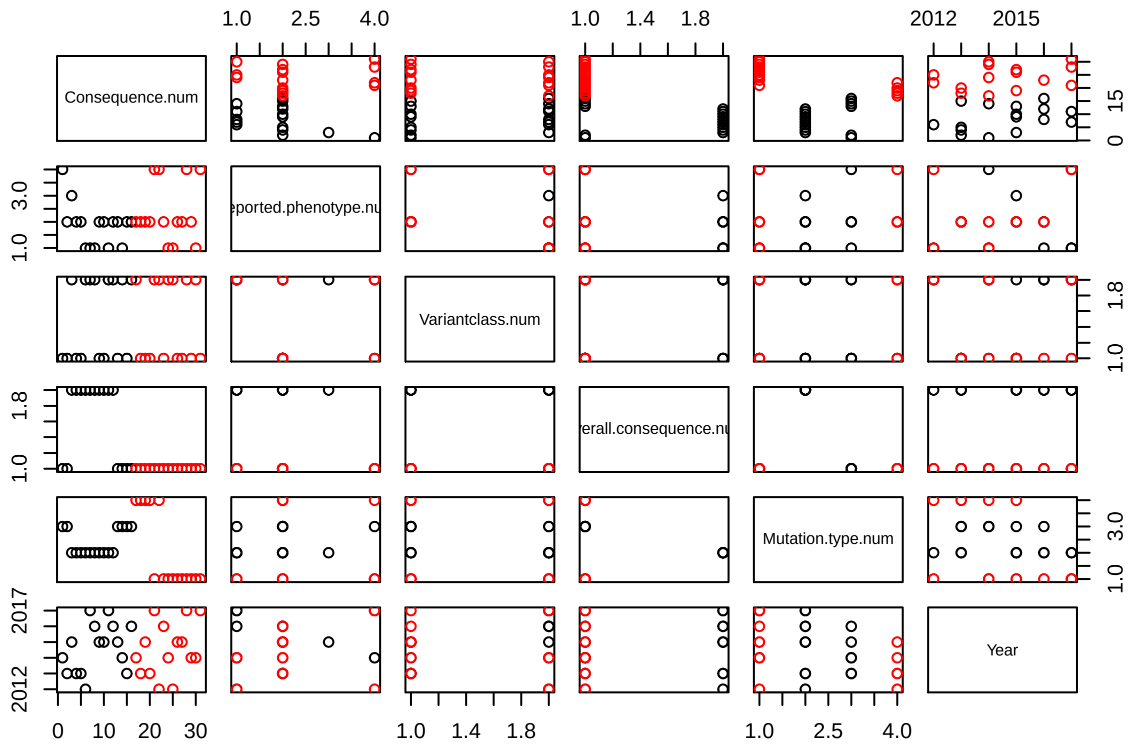
Multidimensional scaling (MDS)

Multidimensional scaling (MDS) is a means of visualizing the level of similarity of individual cases of a high-dimenional dataset.

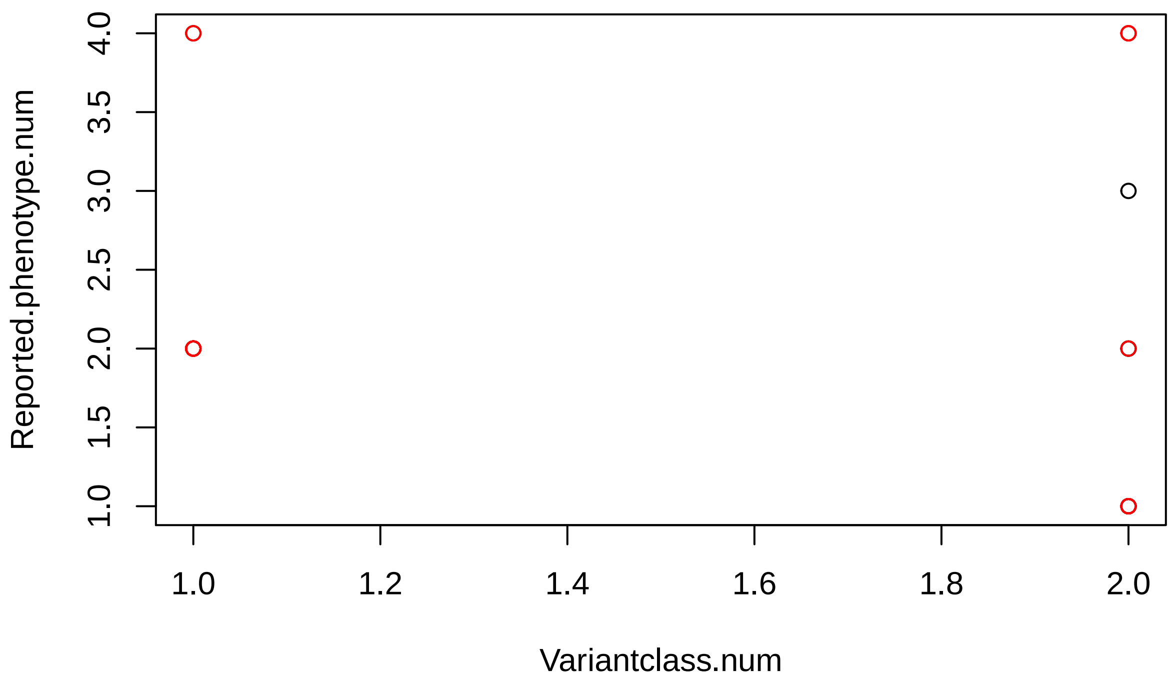
MDS attempts to find an embedding from the I objects into $\mathbb{R}^N$ such that distances are preserved.

Below are some MDS plots:

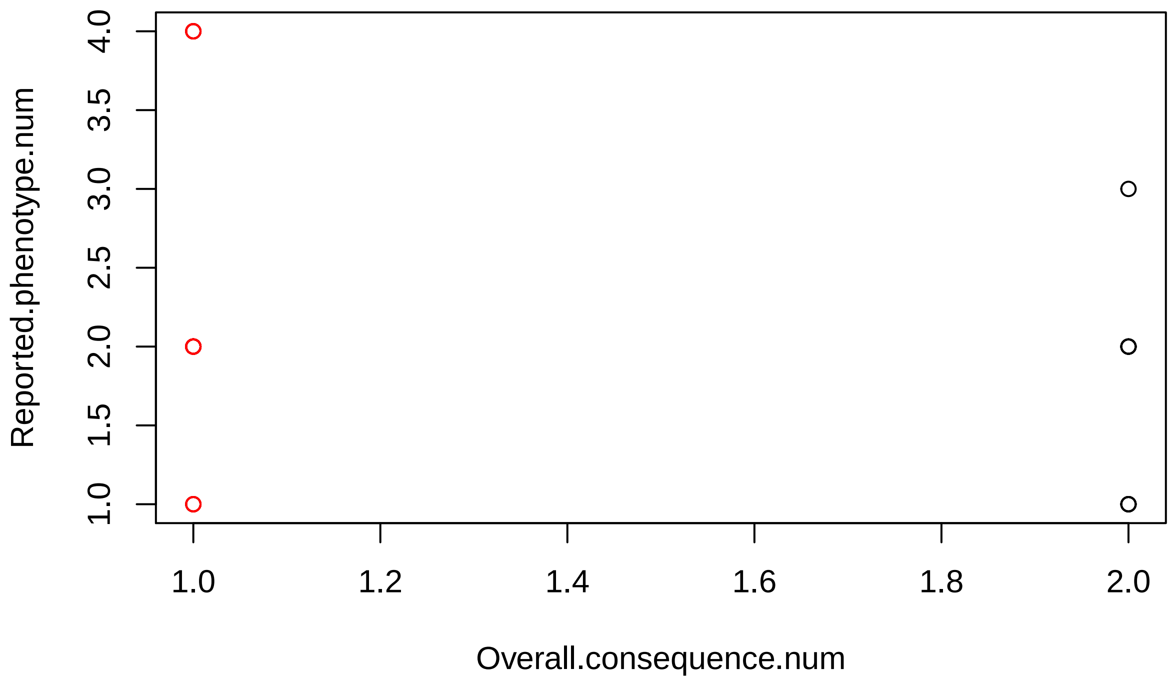
```
plot(dfHGMDCluster, col = dfHGMDCluster.2.cluster$cluster)     # Plot Clusters
```



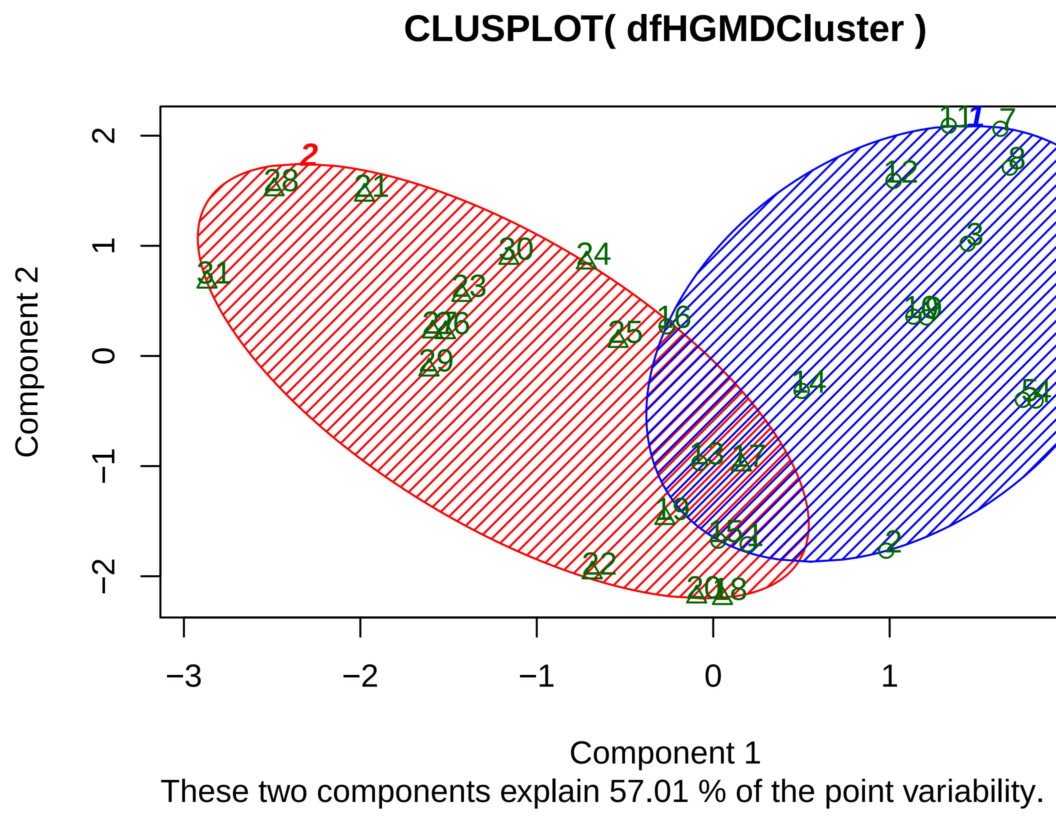
```r
# Plot Consequence.num and Reported.phenotype.num
plot(dfHGMDCluster[c("Variantclass.num", "Reported.phenotype.num")], col =
       dfHGMDCluster.2.cluster$cluster)
points(dfHGMDCluster.2.cluster$centers,
       col = 1:2,
       pch = 8)
```



```r
# Plot Mutation.type.num and Reported.phenotype.num
plot(dfHGMDCluster[c("Overall.consequence.num", "Reported.phenotype.num")], col =
       dfHGMDCluster.2.cluster$cluster)
points(dfHGMDCluster.2.cluster$centers,
       col = 1:2,
       pch = 8)
```



```
# Centroid Plot against 1st two discriminant functions
clusplot(
  dfHGMDCluster,
  dfHGMDCluster.2.cluster$cluster,
  color = TRUE,
  shade = TRUE,
  labels = 2,
  lines = 0
)
```



```
# library(fpc)
# plotcluster(dfHGMDCluster,dfHGMDCluster.2.cluster$cluster)
```

For Overall.consequence the data will change to: LoF = 1 nLoF = 2

For Variant Class the data will change to: DM = 1 DM? = 2

For Reported Phenotype the data will change to: Autism = 1 Epilepsy = 2 Eyelid myoclonia = 3 Intellectual disability = 4

The plots look OK. There is overlapping in the centroid plot. I wish this would have clustered better.

6.3.2.2.2 K-medoids clustering in R

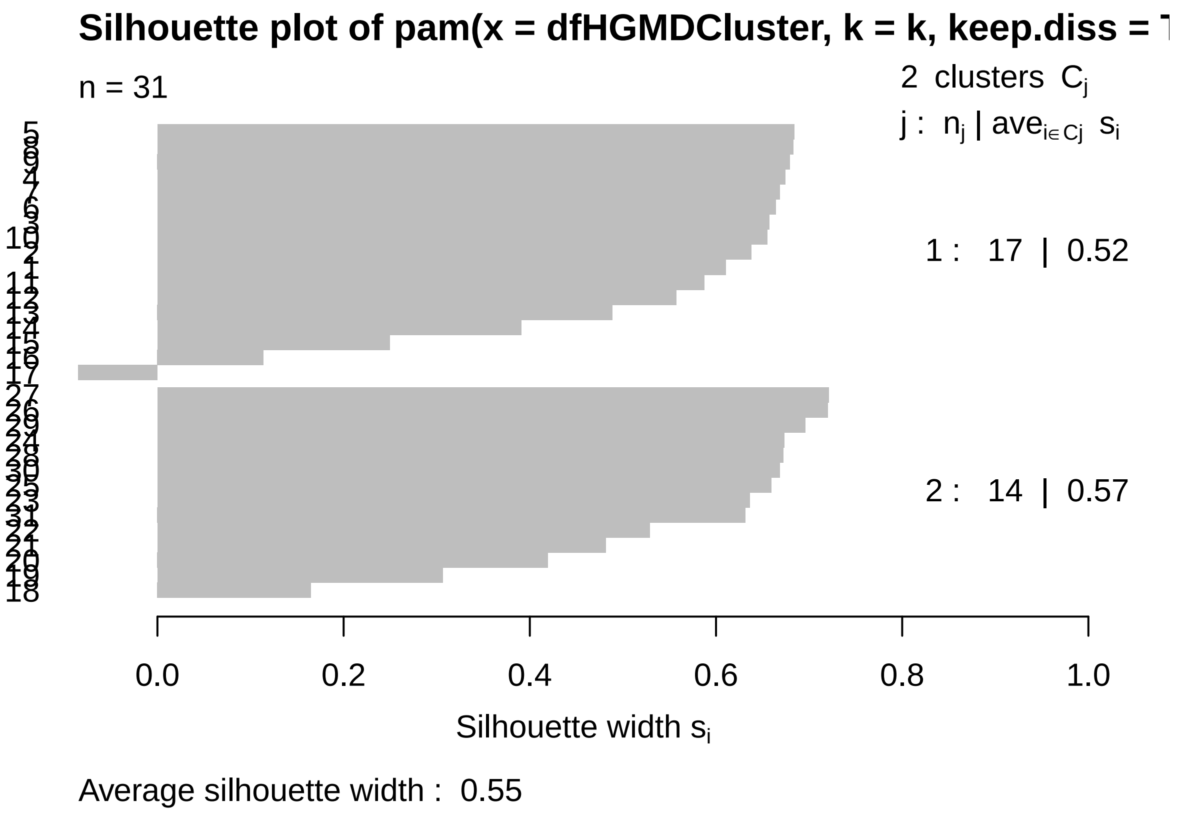
The K-medoids or Partitioning Around Medoids (PAM) algorithm is related to the k-means algorithm and but uses medoid shifts rather than reassigning points based on Euclidean distance. Each cluster is represented by one of the objects (i.e. points) in the cluster A medoid is a point in a cluster whose dissimilarity to all the points in the cluster is minimal. Medoids are similar in concept to means or centroids, but medoids are always members of the data set. That is, in 2D Cartesian space a centroid can be any valid x.y coordinate. Whereas a medoid must be one of the data points.

Below I use R to apply the k-medoids algorithm:

```
# PAM
k <- 2
dfHGMDCluster.pam.2.clust <-
pam(dfHGMDCluster, k, keep.diss = TRUE, keep.data = TRUE)
dfHGMDCluster.pam.2.clust
```

```
## Medoids:
##      ID Consequence.num Reported.phenotype.num Variantclass.num
## [1,]  9               9                      2                1
## [2,] 26              26                      2                1
##      Overall.consequence.num Mutation.type.num Year
## [1,]                       2                 2 2015
## [2,]                       1                 1 2015
## Clustering vector:
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## Objective function:
##    build     swap
## 5.869938 4.628850
##
## Available components:
##  [1] "medoids"    "id.med"     "clustering" "objective"  "isolation"
##  [6] "clusinfo"   "silinfo"    "diss"       "call"       "data"
```

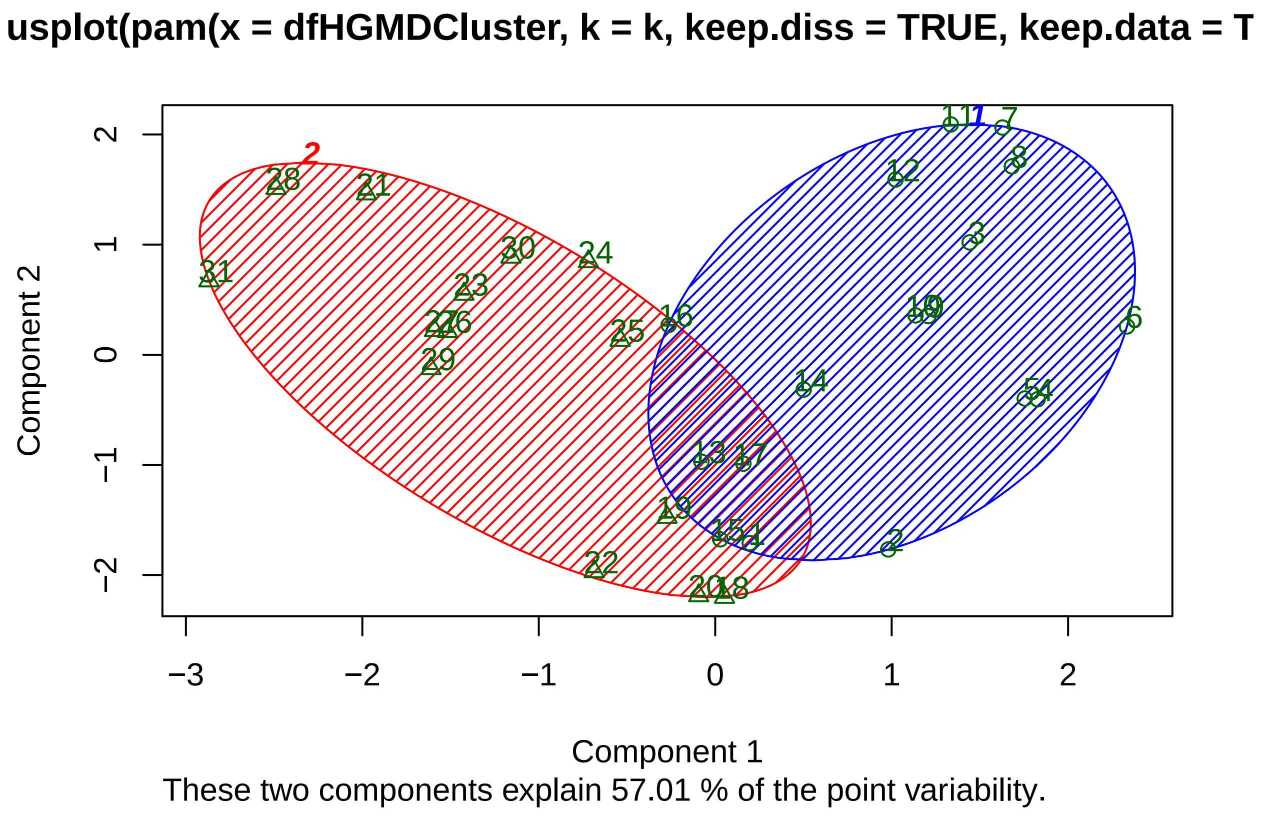
```
plot(dfHGMDCluster.pam.2.clust, which.plots = 2)
```



```
# long lines good - means greater within cluster similarity

# Centroid Plot against 1st two discriminant functions
clusplot(
dfHGMDCluster.pam.2.clust,
color = TRUE,
shade = TRUE,
labels = 2,
```

```r
lines = 0
)
```



The silhouette plot looks good, long lines are good, short ones are not. It appears that there are more long lines.

The centroid plot looks a little better than the k-means plot, but there seems to be overlap with the clusters.

6.3.3 Gap statistic

clusGap() calculates a goodness of clustering measure, the â gapâ statistic. For each number of clusters k, it compares $\log(W(k))$ with $E^*[\log(W(k))]$ where the latter is defined via bootstrapping, i.e. simulating from a reference distribution.

maxSE(f, SE.f) determines the location of the maximum of f, taking a â 1-SE ruleâ into account for the *SE* methods. The default method “firstSEmax” looks for the smallest k such that its value f(k) is not more than 1 standard error away from the first local maximum. This is similar but not the same as “Tibs2001SEmax”, Tibshirani et al’s recommendation of determining the number of clusters from the gap statistics and their standard deviations.

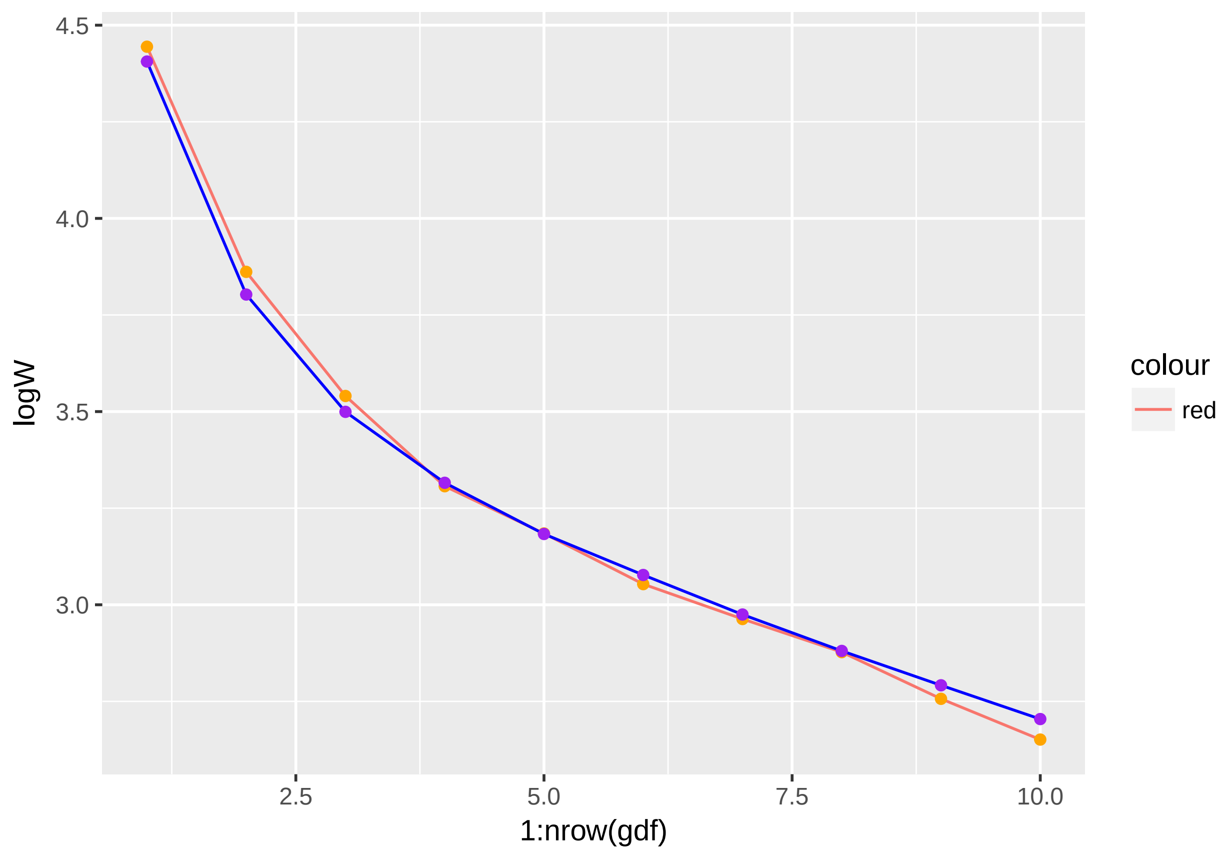
```r
gap <-
  clusGap(dfHGMDCluster, FUNcluster = pam, K.max = 10) # Bootstrapping
  gap$Tab
```

```
##          logW   E.logW          gap     SE.sim
##  [1,] 4.444358 4.405934 -0.038424818 0.07974331
##  [2,] 3.861596 3.802927 -0.058668961 0.06715870
##  [3,] 3.540359 3.499303 -0.041056420 0.06031533
##  [4,] 3.306797 3.315699  0.008901448 0.05136976
##  [5,] 3.184518 3.183143 -0.001375534 0.05836193
##  [6,] 3.053437 3.077105  0.023668841 0.05769219
##  [7,] 2.963147 2.974743  0.011595714 0.05789634
##  [8,] 2.877389 2.880581  0.003192080 0.06044070
##  [9,] 2.756605 2.791579  0.034974451 0.06270258
## [10,] 2.651120 2.704217  0.053096953 0.06781101
```

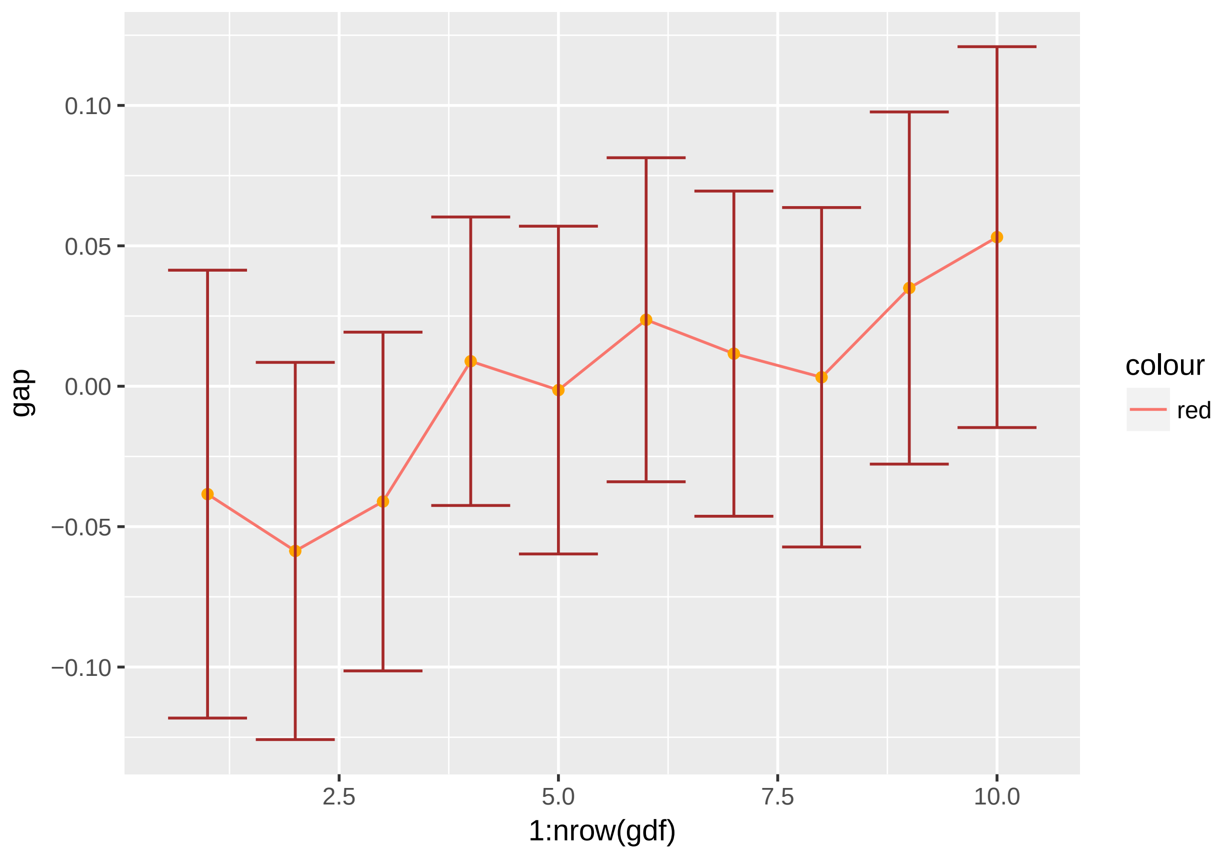
```
gdf <- as.data.frame(gap$Tab)
head(gdf)
```

```
##       logW   E.logW          gap     SE.sim
## 1 4.444358 4.405934 -0.038424818 0.07974331
## 2 3.861596 3.802927 -0.058668961 0.06715870
## 3 3.540359 3.499303 -0.041056420 0.06031533
## 4 3.306797 3.315699  0.008901448 0.05136976
## 5 3.184518 3.183143 -0.001375534 0.05836193
## 6 3.053437 3.077105  0.023668841 0.05769219
```

```
qplot(
x = 1:nrow(gdf),
y = logW,
data = gdf,
geom = "line",
color = "red"
) +
geom_point(aes(y = logW), color = "orange") +
geom_line(aes(y = E.logW), color = "blue") +
geom_point(aes(y = E.logW), color = "purple")
```



```
# Gap statistic
qplot(
x = 1:nrow(gdf),
y = gap,
data = gdf,
geom = "line",
color = "red"
) +
geom_point(aes(y = gap), color = "orange") +
geom_errorbar(aes(ymin = gap - SE.sim, ymax = gap + SE.sim), color = "brown")
```



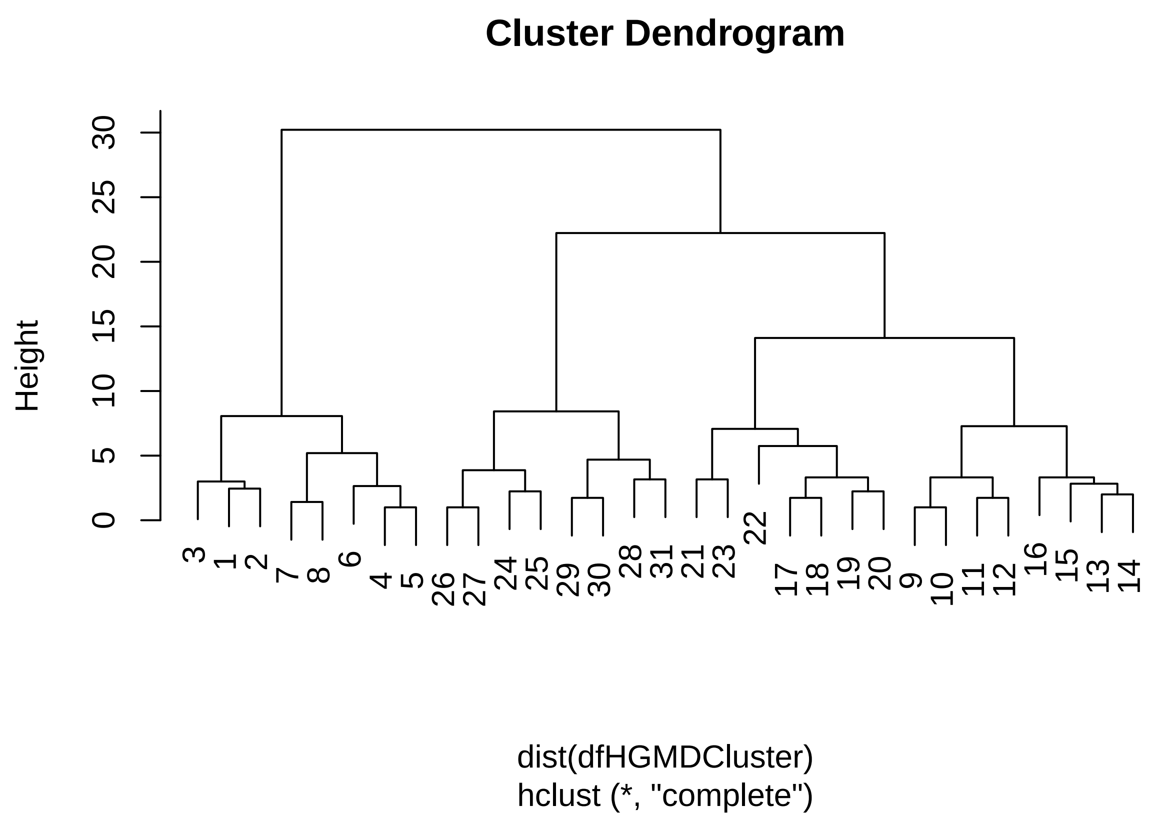
It looks like around 2, the gap increases at a higher slope, so I think a k of 2 is good.

6.3.4 Hierarchical clustering in R

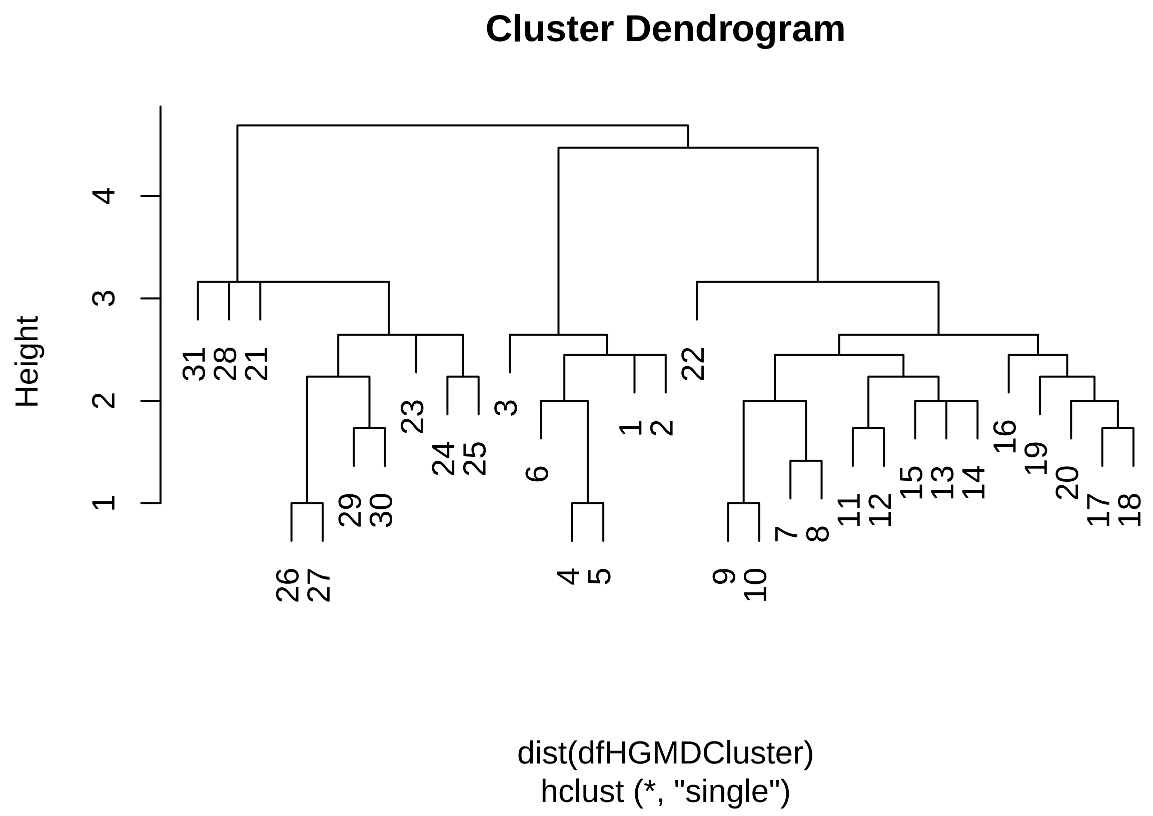
In hierarchical clustering the idea is to group data objects (i.e. points) into a tree of clusters. That is, hierarchical clustering is a method of cluster analysis which seeks to build a hierarchy of clusters.

Below is my analysis using Hierarchical clustering:

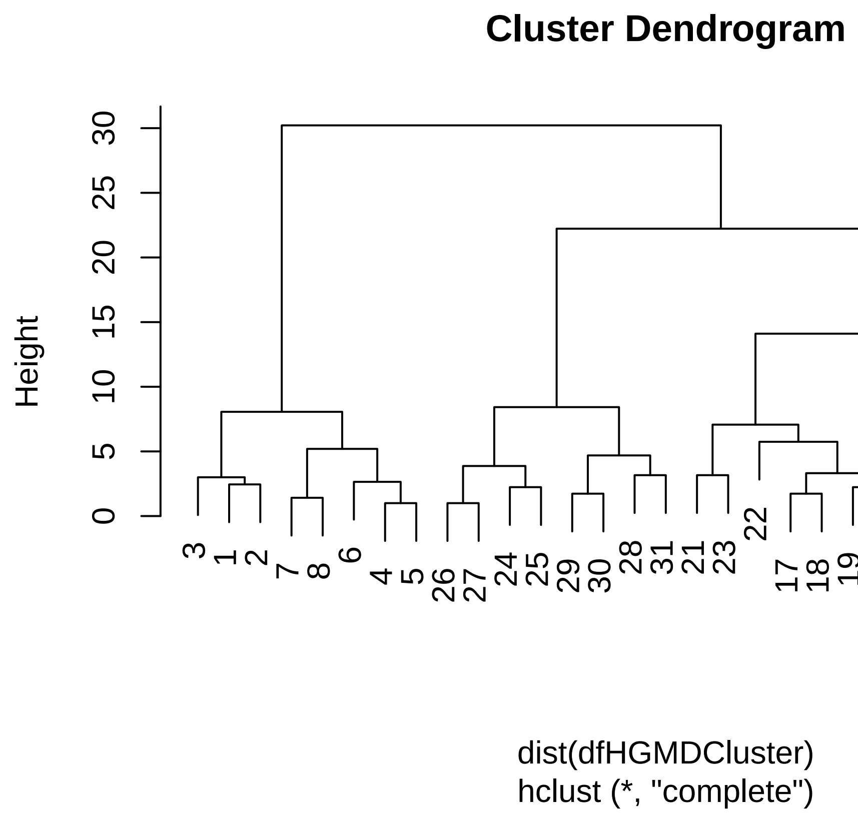
```
dfHGMDCluster.h.clust<- hclust(d=dist(dfHGMDCluster))
plot(dfHGMDCluster.h.clust)
```



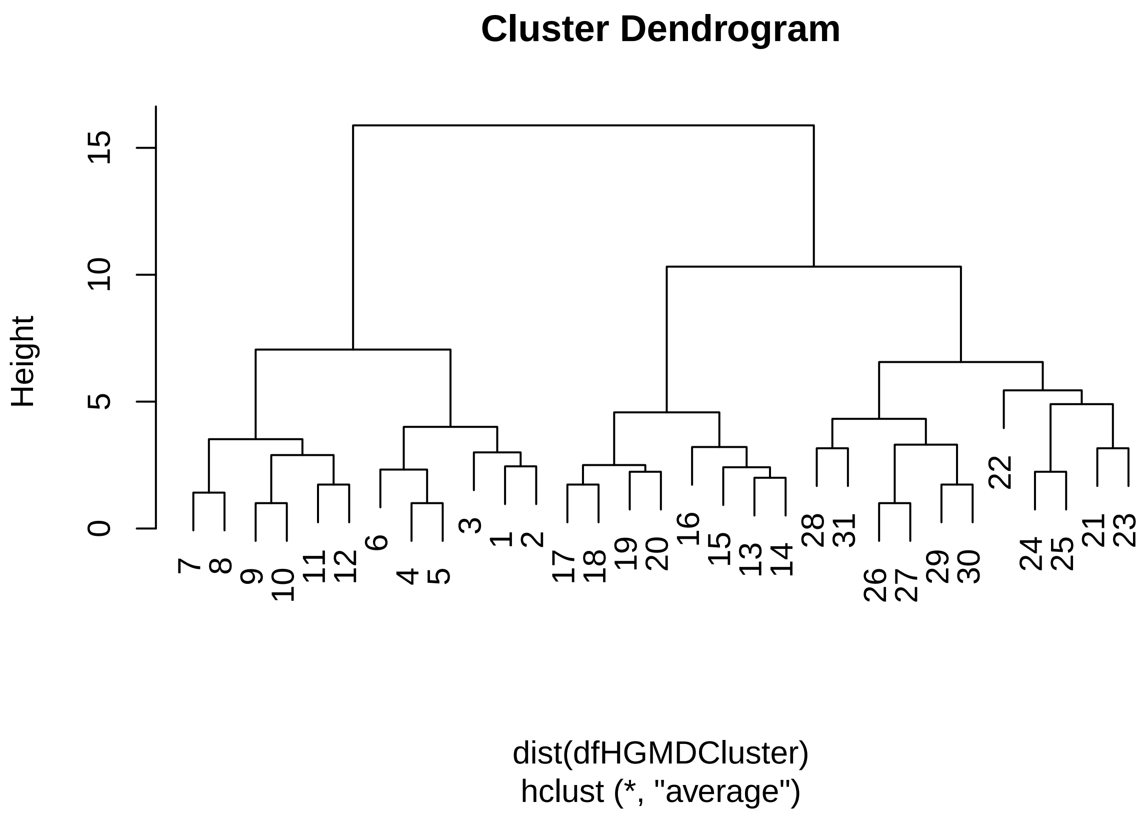
```
dfHGMDCluster.h.clust.si<- hclust(dist(dfHGMDCluster), method = "single")
dfHGMDCluster.h.clust.co<- hclust(dist(dfHGMDCluster), method = "complete")
dfHGMDCluster.h.clust.av<- hclust(dist(dfHGMDCluster), method = "average")
dfHGMDCluster.h.clust.ce<- hclust(dist(dfHGMDCluster), method = "centroid")
dfHGMDCluster.h.clust.me<- hclust(dist(dfHGMDCluster), method = "ward.D")
plot(dfHGMDCluster.h.clust.si)
```



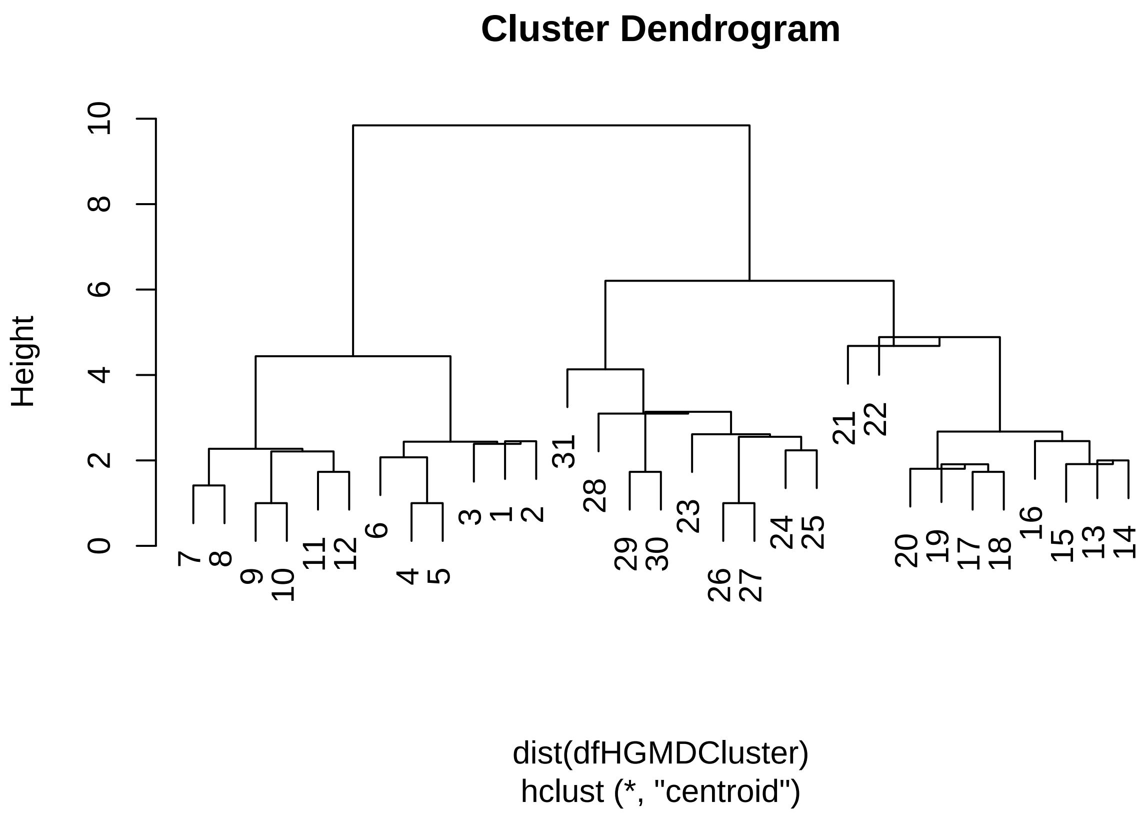
```
plot(dfHGMDCluster.h.clust.co)
```



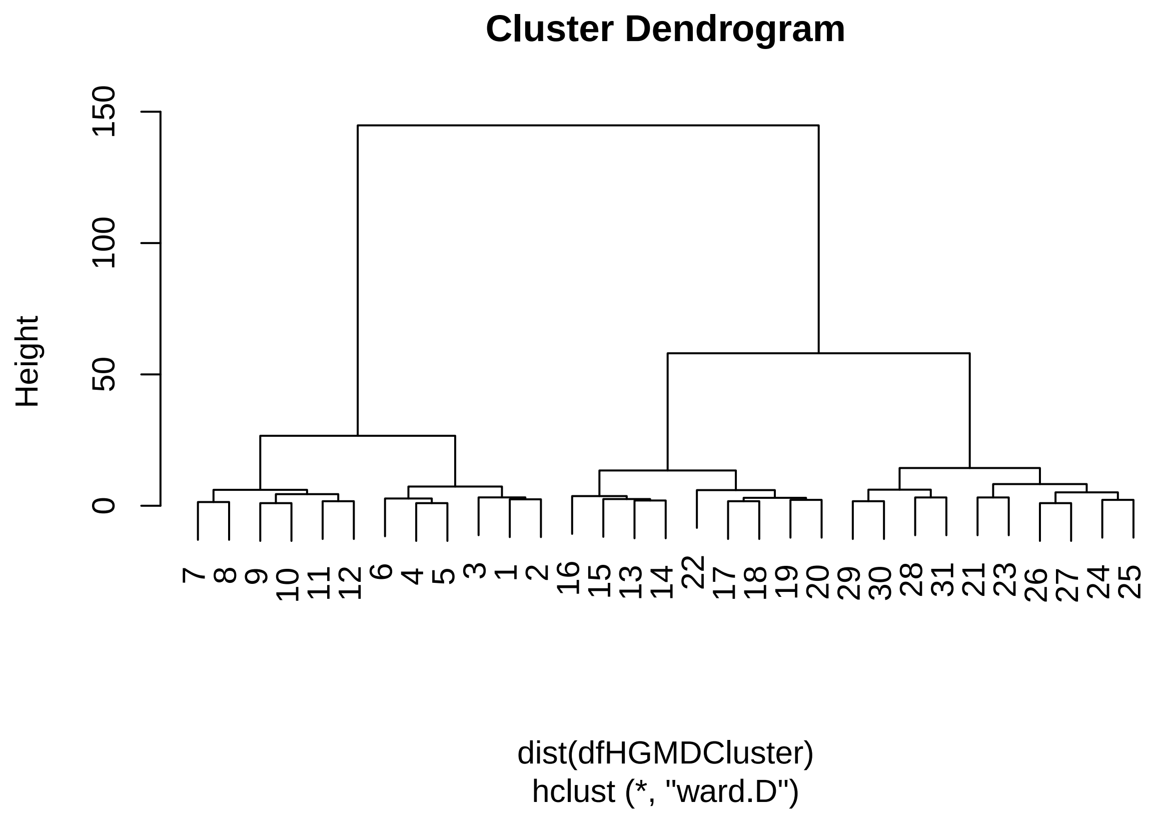
```
plot(dfHGMDCluster.h.clust.av)
```



```r
plot(dfHGMDCluster.h.clust.ce)
```

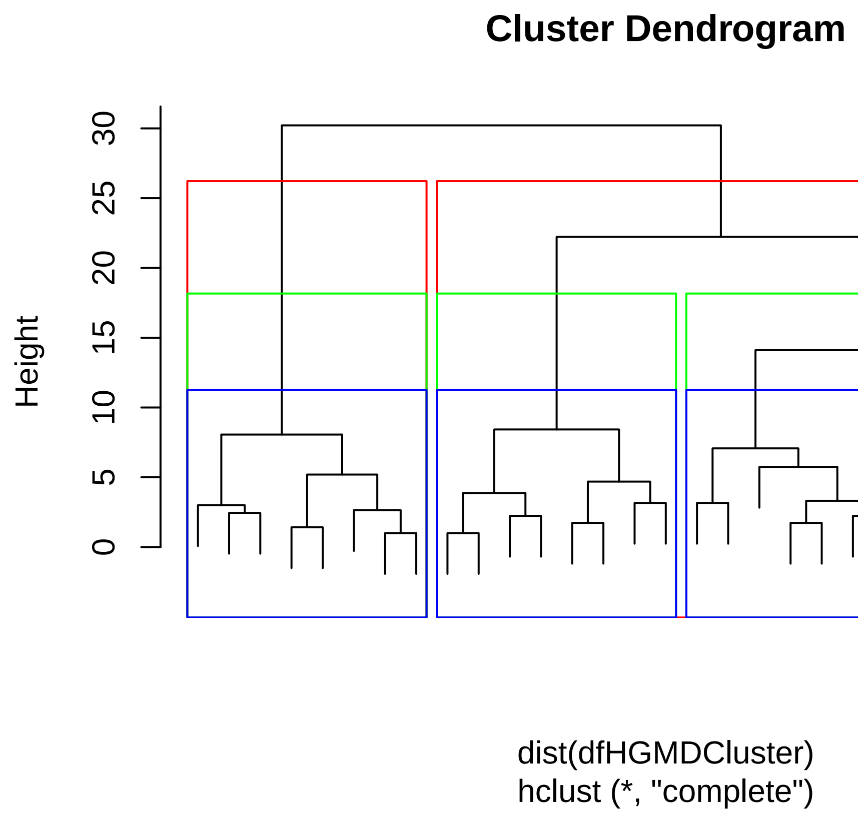


```
plot(dfHGMDCluster.h.clust.me)
```



6.3.4.1 Plotting to deterimine the cluster level.

```r
plot(dfHGMDCluster.h.clust, labels = FALSE)
rect.hclust(dfHGMDCluster.h.clust, k=2, border="red")
rect.hclust(dfHGMDCluster.h.clust, k=3, border="green")
rect.hclust(dfHGMDCluster.h.clust, k=4, border="blue")
```



This looks like k = 3 is a good number to cluster with.

7 Evaluation

7.1 System Time

system run time:

Table 1: Total Run Time

user	system	elapsed
74.72	2.23	132.91

7.2 Precision and Recall

Precision: fraction of retrieved docs that are relevant = relevant/retrieved Recall: fraction of relevant docs that are retrieved = retrieved/relevant

Table 2: Precision and Recall

Retrieved?	Relevant	Non-relevant
Retrieved	3	0
Not Retrieved	0	16

Retrieved?	Relevant	Non-relevant

Precision P = 3/(3 + 0) = 1

Recall R = 3/(3 + 0) = 1

8 Conclusion

9 References

1. Wickham, H. Easily Harvest (Scrape) Web Pages. 2016; Available from: https://cran.r-project.org/web/packages/rvest/rvest.pdf.
2. Harrison, J. R Bindings for 'Selenium WebDriver'. 2017; Available from: https://cran.r-project.org/web/packages/RSelenium/RSelenium.pdf.
3. Wickham, H., Tools for Splitting, Applying and Combining Data. 2016.
4. Wickham, H. ggplot2: Elegant Graphics for Data Analysis. 2009; Available from: http://ggplot2.org.
5. Martin Maechler [aut, c., Peter Rousseeuw [aut] (Fortran original), Anja Struyf [aut] (S original), Mia Hubert [aut] (S original), Kurt Hornik [trl, ctb] (port to R; maintenance(1999-2000)), Matthias Studer [ctb], Pierre Roudier [ctb], Juan Gonzalez [ctb]. Finding Groups in Data": Cluster Analysis Extended Rousseeuw et al. 2017; Available from: https://cran.r-project.org/web/packages/cluster/cluster.pdf.
6. Lucas, A. Another Multidimensional Analysis Package. 2014; Available from: https://cran.r-project.org/web/packages/amap/amap.pdf.
7. Lander, J.P. A Collection of Handy, Useful Functions. 2017; Available from: https://cran.r-project.org/web/packages/useful/useful.pdf.
8. Brownlee, J. Supervised and Unsupervised Machine Learning Algorithms. 2016; Available from: https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/.
9. Cluster analysis, in Wikipedia, the free encyclopedia. 2017.
10. k-means clustering, in Wikipedia, the free encyclopedia. 2017.
11. Brown, N.B., Clustering. Norhteastern University.
12. Hierarchical clustering, in Wikipedia, the free encyclopedia. 2017.