ETH zürich

Bioinformatic Data Analysis

PhD Course in Basic and Applied Cancer Biology

Franziska Singer, 23.11.2015

nexus Personalized Health Technologies

Outline

Introduction
- Motivation
- Typical analysis pipelines

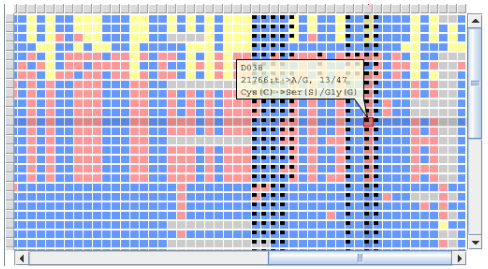
Mapping
- Mapper Choice
- File formats

Variant Call
- Caller choice
- File formats

Example analysis
- Filtering
- Fisher’s exact test

Motivation

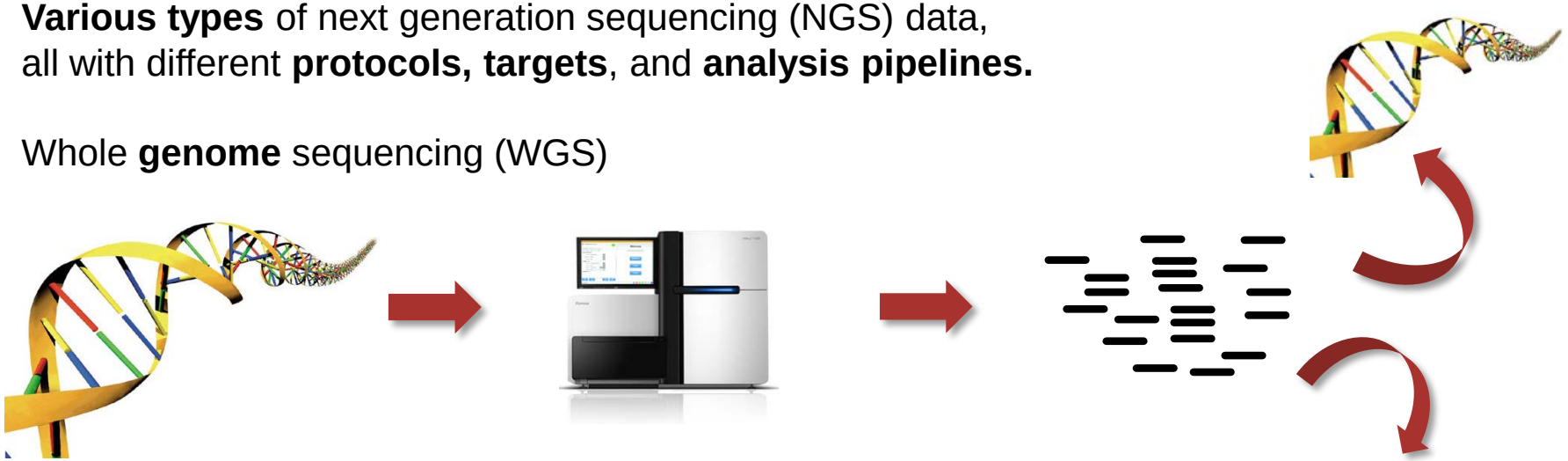
?



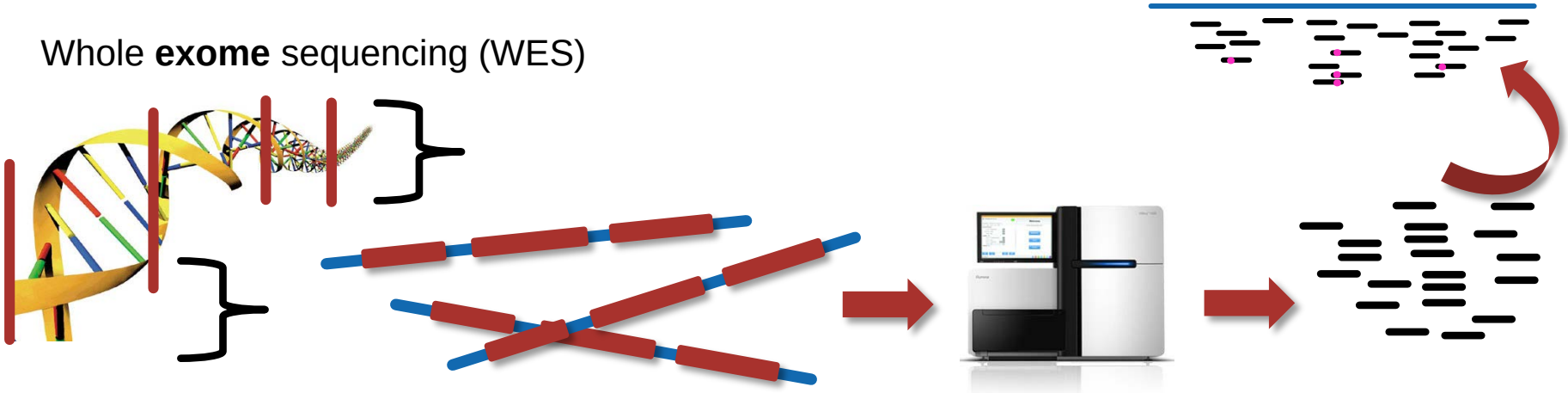
Typical analyses

Various types of next generation sequencing (NGS) data,
all with different **protocols, targets**, and **analysis pipelines.**

Whole **genome** sequencing (WGS)

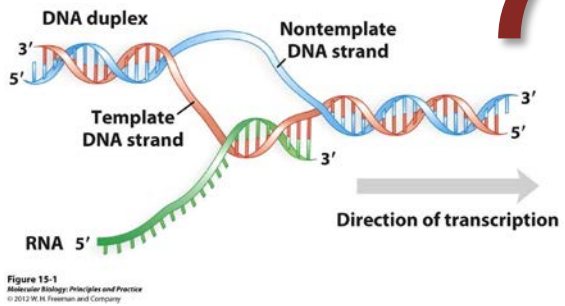


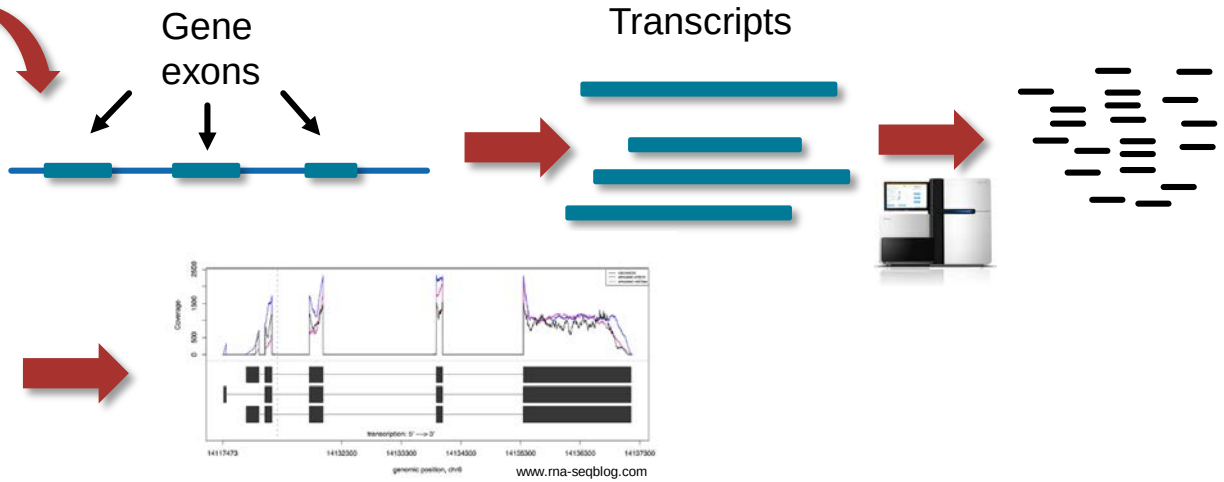
Whole **exome** sequencing (WES)



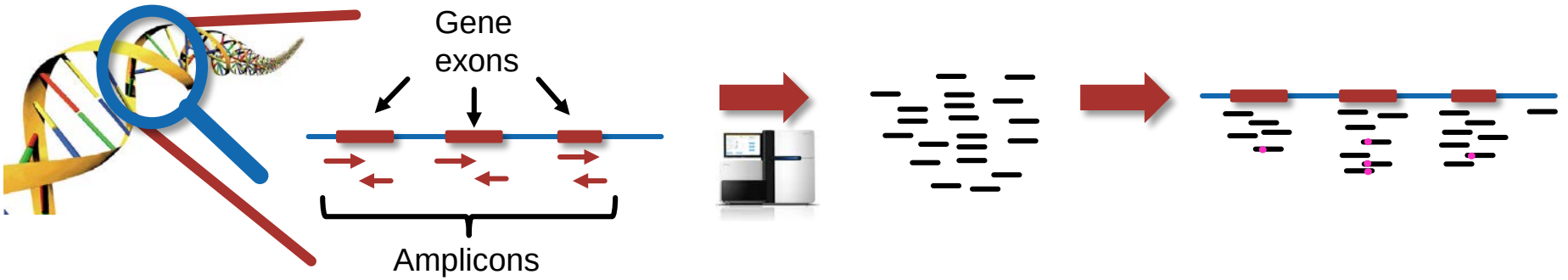
Typical analyses

RNA-Seq





Today:
Focus on variant calling on **targeted** deep amplicon sequencing data



Example exercise

Question:

8 breast cancer patients, treated with trastuzumab
4 **responder**, 4 **nonresponder** (rezidiv)

- Compare mutations of responders and nonresponders, identify genes distinguishing the categories

Targeted deep amplicon sequencing:

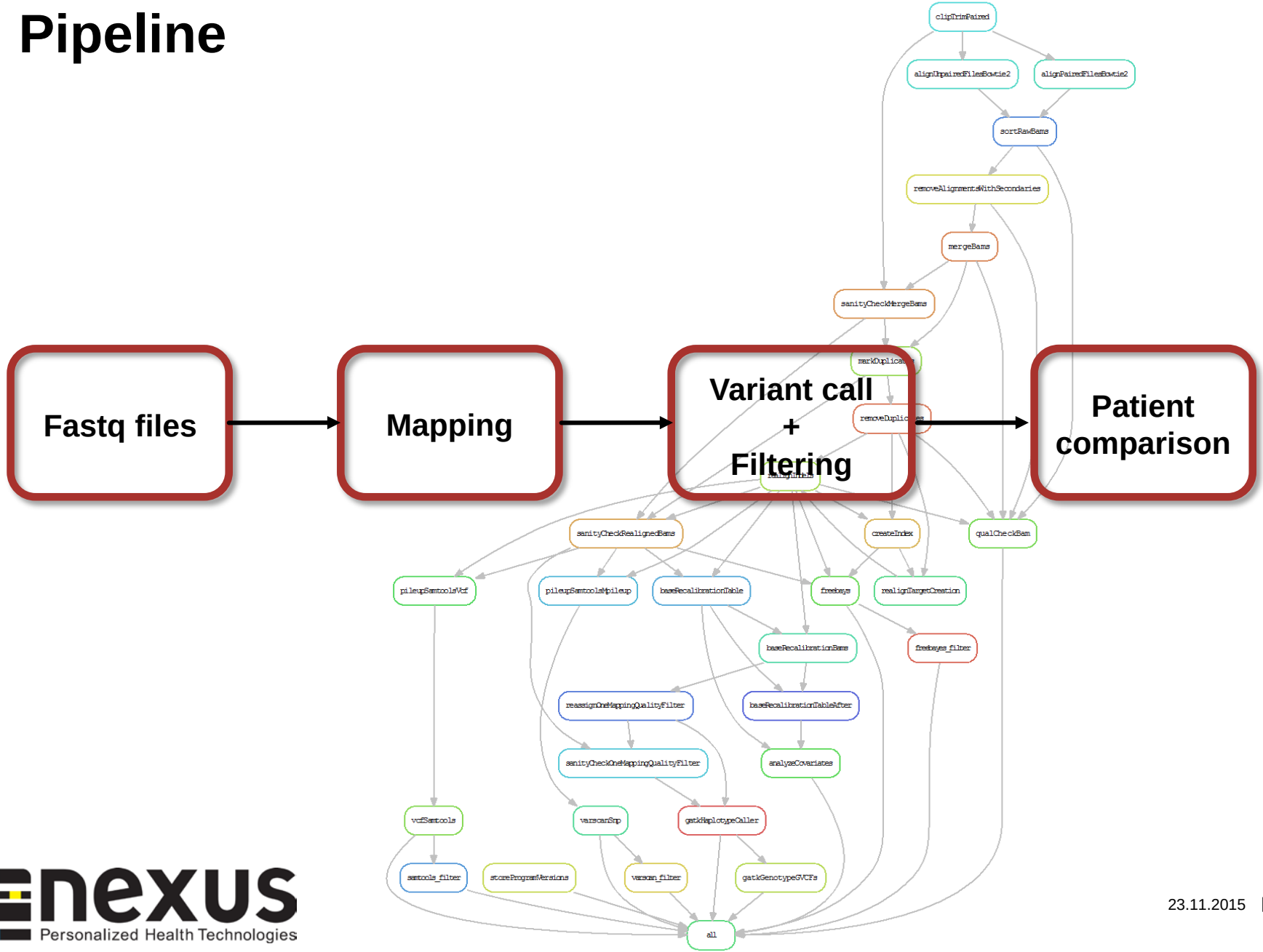
Gene panels:

- Much higher coverage per base
- Variant calls with greater confidence

Genes of interest:

MPL	NRAS	NOTCH2	IRF6	ALK
CASP8	IDH1	ERBB4	VHL	MLH1
CTNNB1	PIK3CA	TP63		

Pipeline


Fastq files
Mapping
Variant call
+
Filtering
Patient comparison
clipTrimPaired
alignUnpairedFilesBowtie2
alignPairedFilesBowtie2
sortRawBams
removeAlignmentsWithSecondaries
mergeBams
sanityCheckMergeBams
markDuplicates
removeDuplicates
realignIntervals
sanityCheckRealignedBams
createIndex
qualCheckBam
pileupSamtoolsVcf
pileupSamtoolsMpileup
baseRecalibrationTable
freebays
realignTargetCreation
baseRecalibrationBams
freebayes_filter
reassignOneMappingQualityFilter
baseRecalibrationTableAfter
sanityCheckOneMappingQualityFilter
analyzeCovariates
vcfSamtools
varscanSnp
gatkHaplotypeCaller
samtools_filter
storeProgramVersions
varscan_filter
gatkGenotypeGVCFs
all

Mapping

FASTA reference sequence

```
>chr1
AGCTTTTCATTCTGACTGCAACGGGCAATATGTCTCTGTGTGGATTAAAAAAAGAGTGTCTGATAGCAGC
TTCTGAACTGGTTACCTGCCGTGAGTAAATTAAAATTTTATTGACTTAGGTCACTAAATACTTTAACCAA
TATAGGCATAGCGCACAGACAGATAAAAATTACAGAGTACACAACATCCATGAAACGCATTAGCACCACC
ATTACCACCACCATCACCATTACCACAGGTAACGGTGCGGGCTGACGCGTACAGGAAACACAGAAAAAAG
CCCGCACCTGACAGTGCGGGCTTTTTTTTTCGACCAAAGGTAACGAGGTAACAACCATGCGAGTGTTGAA
GTTCGGCGGTACATCAGTGGCAAATGCAGAACGTTTTCTGCGTGTTGCCGATATTCTGGAAAGCAATGCC
AGGCAGGGGCAGGTGGCCACCGTCCTCTCTGCCCCCGCCAAAATCACCAACCACCTGGTGGCGATGATTG
AAAAAACCATTAGCGGCCAGGATGCTTTACCCAATATCAGCGATGCCGAACGTATTTTTGCCGAACTTTT
GACGGGACTCGCCGCCGCCCAGCCGGGGTTCCCGCTGGCGCAATTGAAAACTTTCGTCGATCAGGAATTT
```

FASTQ read sequences

```
@WU1C0:05822:03040
GTAACCATCATACTATTCTCTGCTTCTATGAGTTCAGATTTTTTAGATTCTAAGTGAGATCATGCAATATTTGTCTTTCTGTGTCTAGCTTATTTCACTTAGCTTAATG
+
://*/7779:::9:897;::===<8<<;<87486:99577777%5::;17597;:888;:9:<<<>7<<?A5<<655(5:9677776677)...(.667-339:6:6:<
@WU1C0:08807:05116
GGCTCTTATATTGATAGCTGTTCCAGAGGCAATCAATAGCTATTAGTCGGTTTTATTCTTATTTTTCTGTCTGATCTTACAGGGGAGCAAACTGTGGCAAAGTATGAAC
+
8/5::;48<<>7<<><<<<<<9?<>>>=9=<9==C9>=<=><=9=<<<<8<<<4<<5;;5:;<<<1<<<888<;;;;8<===?>7=>=BB8>=@>>9==>8>><<<<38
@WU1C0:01264:04847
GCTTGAATTCCCACAGTGCTGCTTGAGCCTCACACCATGTCATCCTGCCAGGCACCCAGATCCAGTTCTAGAGTTTCACATGATCATGAGTGTTGGTTAATAAGTCAAT
+
==>9=>9=9=?8=>?==<==>==9====9=>>>>@8=>>>;88898888<<8=;;;6;<=<<8<>>6;<<<<<>?7889<<<<<=<<==??BA9>6<5<7<<7<>?@?B
```

Mapping

Mapper choice:

- Variety available
- Often specific purposes, e.g., RNA-Seq read mapper or read mapper for long reads
- Single end vs. Paired end

Popular mapper:

Bowtie2 (Langmead and Salzberg, 2012), BWA (Li and Durbin, 2010), Star (Dobin et al., 2013), SMALT (http://www.sanger.ac.uk/science/tools/smalt)…

➢ Data type, Benchmarks, Experience, Question at hand….?

BarraCUDA	A GPGPU accelerated Burrows-Wheeler transform (FM-index) short read alignment program based on BWA, supports
BBMap	Uses a short kmers to rapidly index genome; no size or scaffold count limit. Higher sensitivity and specificity than Burro more accurate than Smith-Waterman. Handles Illumina, 454, PacBio, Sanger, and Ion Torrent data. Splice-aware; capa
BFAST	Explicit time and accuracy tradeoff with a prior accuracy estimation, supported by indexing the reference sequences. Op map ABI SOLiD color space reads). Performs a full Smith Waterman alignment.
BLASTN	BLAST's nucleotide alignment program, slow and not accurate for short reads, and uses a sequence database (EST, sa
BLAT	Made by Jim Kent. Can handle one mismatch in initial alignment step.
Bowtie	Uses a Burrows-Wheeler transform to create a permanent, reusable index of the genome; 1.3 GB memory footprint for
HIVE-hexagon	Uses a hash table and bloom matrix to create and filter potential positions on the genome. For higher efficiency uses cr allows indels and divergent sensitive alignments on viruses and bacteria as well as more conservative eukaryotic alignn
BWA	Uses a Burrows-Wheeler transform to create an index of the genome. It's a bit slower than bowtie but allows indels in al
BWA-PSSM	A probabilistic short read aligner based on the use of position specific scoring matrices (PSSM). The aligner is adaptabl observed in Ancient DNA, PAR-CLIP data or genomes with biased nucleotide compositions.[28]
CASHX	Quantify and manage large quantities of short-read sequence data. CASHX pipeline contains a set of tools that can be
Cloudburst	Short-read mapping using Hadoop MapReduce
CUDA-EC	Short-read alignment error correction using GPUs.
CUSHAW	A CUDA compatible short read aligner to large genomes based on Burrows-Wheeler transform.
CUSHAW2	Gapped short-read and long-read alignment based on maximal exact match seeds. This aligner supports both base-spa
CUSHAW2-GPU	GPU-accelerated CUSHAW2 short-read aligner.
CUSHAW3	Sensitive and Accurate Base-Space and Color-Space Short-Read Alignment with Hybrid Seeding
drFAST	Read mapping alignment software that implements cache obliviousness to minimize main/cache memory transfers like map locations for improved structural variation discovery.
ELAND	Implemented by Illumina. Includes ungapped alignment with a finite read length.
ERNE	Extended Randomized Numerical alignEr for accurate alignment of NGS reads. It can map bisulfite-treated reads.
GASSST	Finds global alignments of short DNA sequences against large DNA banks
GEM	High-quality alignment engine (exhaustive mapping with substitutions and indels). More accurate and several times fast
Genalice MAP	Ultra fast and comprehensive NGS read aligner with high precision and small storage footprint.
Geneious Assembler	Fast, accurate overlap assembler with the ability to handle any combination of sequencing technology, read length, any
GensearchNGS	Complete framework with user-friendly GUI to analyse NGS data. It integrates a proprietary high quality alignment algor detect variants and generate reports. It is geared towards re-sequencing projects, namely in a diagnostic setting.
GMAP and GSNAP	Robust, fast short-read alignment. GMAP: longer reads, with multiple indels and splices (see entry above under Genom and indel genotyping. Developed by Thomas Wu at Genentech. Used by the National Center for Genome Resources (N

nexus Personalized Health Technologies

Mapping

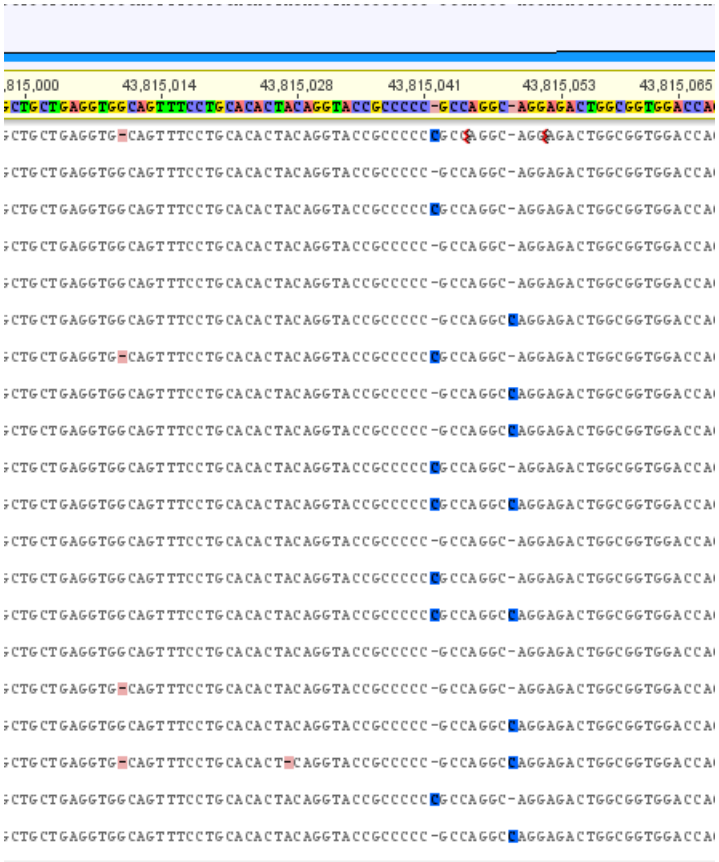
Example call:

smalt index -s 3 referenceIndex referenceSequence.fa

smalt map -f bam -n 10 -o mapping.bam referenceIndex fastqFile.fastq

SAM file:

```
@HD VN:1.4  SO:coordinate
@SQ SN:chr1 LN:249250621
@SQ SN:chr2 LN:243199373
@SQ SN:chr3 LN:198022430
@SQ SN:chr4 LN:191154276
@PG ID:smalt    PN:smalt    VN:0.7.6    CL:smalt map -f bam -n 10 -o smaltMapping/ra
R1UZA:00070:01765   0   chr1    43814957    57  53M1D77M    *   1   0   CTGCATCTAGTG
R1UZA:00206:00987   0   chr1    43814957    57  45M3D83M    *   1   0   CTGCATCTAGTG
R1UZA:00205:01164   0   chr1    43814957    57  92M1I39M    *   1   0   CTGCATCTAGTG
R1UZA:00226:01839   0   chr1    43814957    57  53M1D77M    *   1   0   CTGCATCTAGTG
R1UZA:00200:02207   0   chr1    43814957    60  34M1D15M1D2M1D5M1D7M1I14M   *   1
R1UZA:00297:00798   0   chr1    43814957    57  85M1I46M    *   1   0   CTGCATCTAGTG
R1UZA:00297:01416   0   chr1    43814957    57  53M1D31M1I46M   *   1   0   CTGCATCT
R1UZA:00338:00146   0   chr1    43814957    57  85M1I46M    *   1   0   CTGCATCTAGTG
R1UZA:00340:01504   0   chr1    43814957    57  53M1D23M    *   1   0   CTGCATCTAGTG
R1UZA:00391:00131   0   chr1    43814957    57  85M1I46M    *   1   0   CTGCATCTAGTG
R1UZA:00361:00233   0   chr1    43814957    57  131M    *   1   0   CTGCATCTAGTGCTGG
R1UZA:03374:01012   16  chr1    43814957    57  12M1I72M1D42M3S *   1   0   CTGCATCT
R1UZA:03721:01792   16  chr1    43814957    57  139M    *   1   0   CTGCATCTAGTGCTGG
R1UZA:02749:03201   16  chr1    43814957    57  5M1I126M    *   1   0   CTGCACTCTAGT
R1UZA:02905:00984   16  chr1    43814957    57  9M1I3M1I55M1I64M    *   1   0   CTGC
R1UZA:01268:02916   0   chr1    43814960    57  4S21M1I29M1D77M *   1   0   CTGCCATC
R1UZA:02322:03166   0   chr1    43814962    59  7S48M1D2M1I21M  *   1   0   CTGCACTT
R1UZA:02417:01893   0   chr1    43814966    52  12S8M2I22M1I14M1D6M1I2M1I9M1I6M6S
R1UZA:02016:02395   0   chr1    43814966    57  93S83M1I11M1I28M    *   1   0   AAAA
R1UZA:00074:00786   0   chr1    43814968    57  120M    *   1   0   GCTGGGCCTCAGCGCC
R1UZA:00070:01275   0   chr1    43814968    57  42M1D48M1I29M   *   1   0   GCTGGGCC
R1UZA:00106:00175   0   chr1    43814968    57  120M    *   1   0   GCTGGGCCTCAGCGCC
```



Variant call

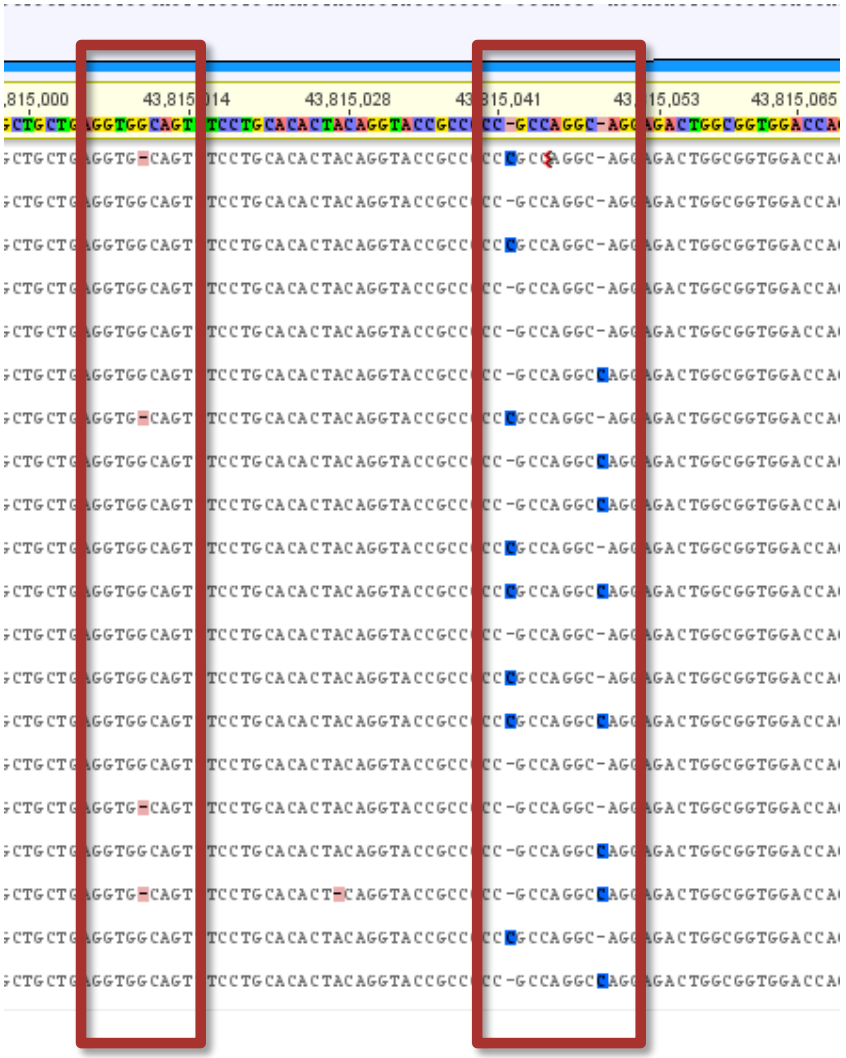
Look at differences between reads and reference

Variety of variant callers

- Assess variant quality by nucleotide coverage, variant frequency, base quality,….

Somatic:
Matched samples Tumor vs. Normal

Popular callers:
GATK (McKenna et al., 2010), Mutect (Cibulskis et al., 2013), VarScan (Koboldt et al., 2012), Freebayes (Garrison and Marth, 2012)…

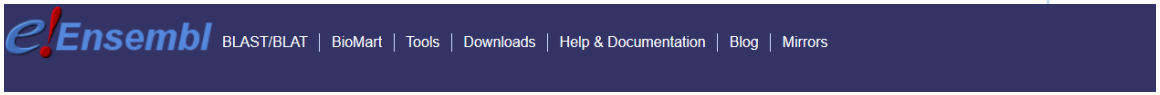


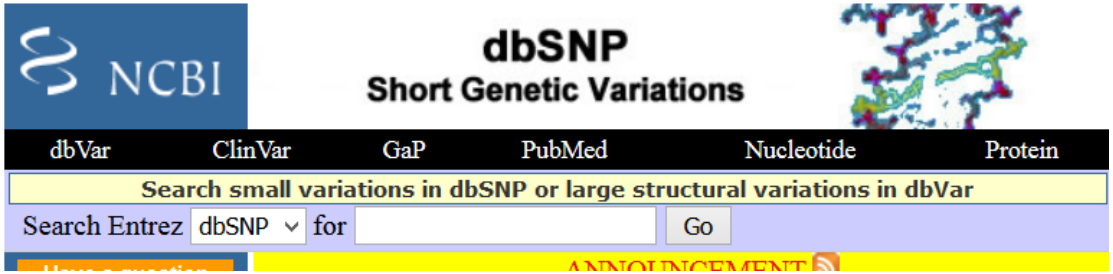
Deletion ?

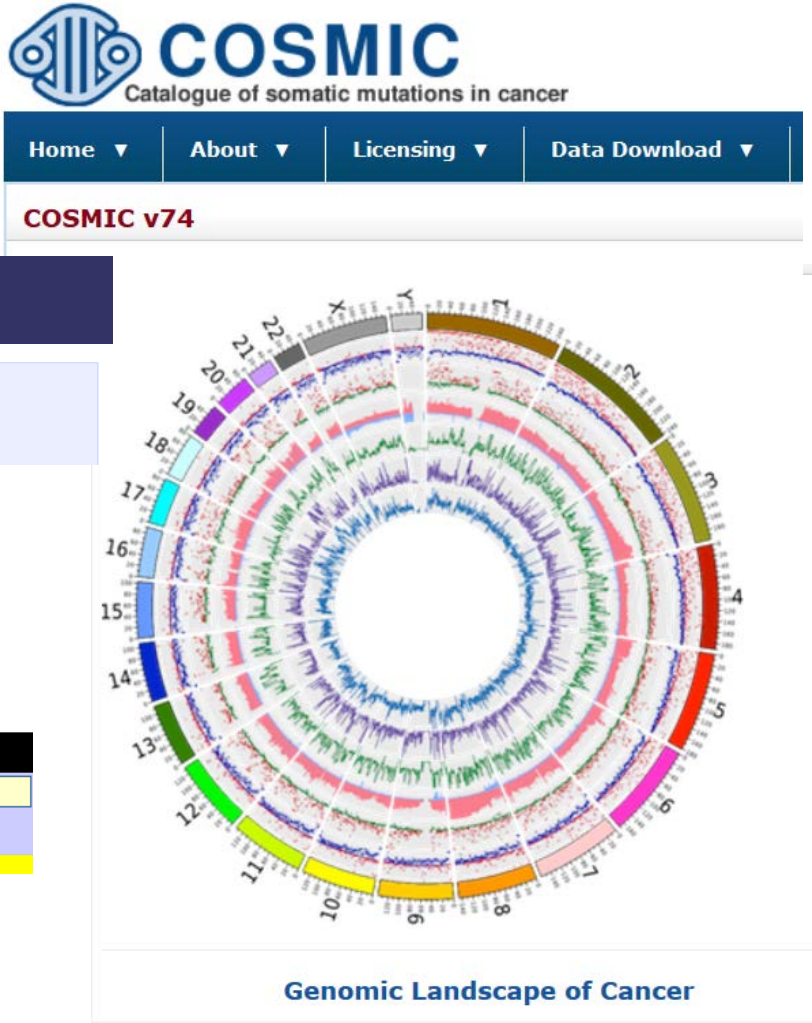
Insert of C ?

Annotation

Compare identified mutations with existing databases







Variant call

Example call:

samtools mpileup -R -d 100000 -l genePanel.bed -Q 1 -B -A -t DP,DP4,SP,DPR,DV -f referenceSequence.fa Mapping.bam | java –jar VarScan.v2.3.9.jar mpileup2cns --p-value 0.05 --min-var-freq 0.01 --min-coverage 8 --output-vcf 1 --strand-filter 1 > varScan.vcf

java -jar snpEff.jar ann GRCh37.70 -dataDir snpEff/ -nodownload –onlyProtein varScan.vcf > varScan_snpEffAnnotated.vcf

VCF file:

#CHROM	POS	ID	REF	ALT	QUAL	FILTER	INFO	FORMAT	Sample1
chr1	120458020	.	CT	C	.	PASS	ADP=3050;WT=0;HET=1;HOM=0;NC=0;ANN=C|frameshift_variant|HIGH|N		
chr1	120458382	.	A	AC	.	PASS	ADP=943;WT=0;HET=1;HOM=0;NC=0;ANN=AC|frameshift_variant|HIGH|N		
chr1	120458636	.	T	TC	.	PASS	ADP=18643;WT=0;HET=1;HOM=0;NC=0;ANN=TC|frameshift_variant|HIGH		
chr1	120458741	.	A	G	.	PASS	ADP=1283;WT=0;HET=1;HOM=0;NC=0;ANN=G|missense_variant|MODERATE		
chr1	120458786	.	C	CG	.	PASS	ADP=1682;WT=0;HET=1;HOM=0;NC=0;ANN=CG|frameshift_variant|HIGH|		
chr1	120465246	.	CA	C	.	PASS	ADP=2669;WT=0;HET=1;HOM=0;NC=0;ANN=C|sequence_feature|LOW|NOTC		
chr1	120465247	.	A	AC	.	PASS	ADP=2677;WT=0;HET=1;HOM=0;NC=0;ANN=AC|sequence_feature|LOW|NOT		

GT:GQ:SDP:DP:RD:AD:FREQ:PVAL:RBQ:ABQ:RDF:RDR:ADF:ADR	0/1:61:2049:1866:1844:24:1.28%:7.2364E-7:26:22:1006:838:24:0
GT:GQ:SDP:DP:RD:AD:FREQ:PVAL:RBQ:ABQ:RDF:RDR:ADF:ADR	0/1:234:2054:1836:1769:82:4.41%:3.5472E-24:27:19:990:779:81:1
GT:GQ:SDP:DP:RD:AD:FREQ:PVAL:RBQ:ABQ:RDF:RDR:ADF:ADR	0/1:102:2058:1644:1606:38:2.31%:5.926E-11:24:21:962:644:38:0
GT:GQ:SDP:DP:RD:AD:FREQ:PVAL:RBQ:ABQ:RDF:RDR:ADF:ADR	0/1:52:2079:1815:1803:21:1.15%:5.2025E-6:25:17:1058:745:1:20
GT:GQ:SDP:DP:RD:AD:FREQ:PVAL:RBQ:ABQ:RDF:RDR:ADF:ADR	0/1:61:2073:1988:1964:24:1.21%:7.2665E-7:25:22:1039:925:24:0

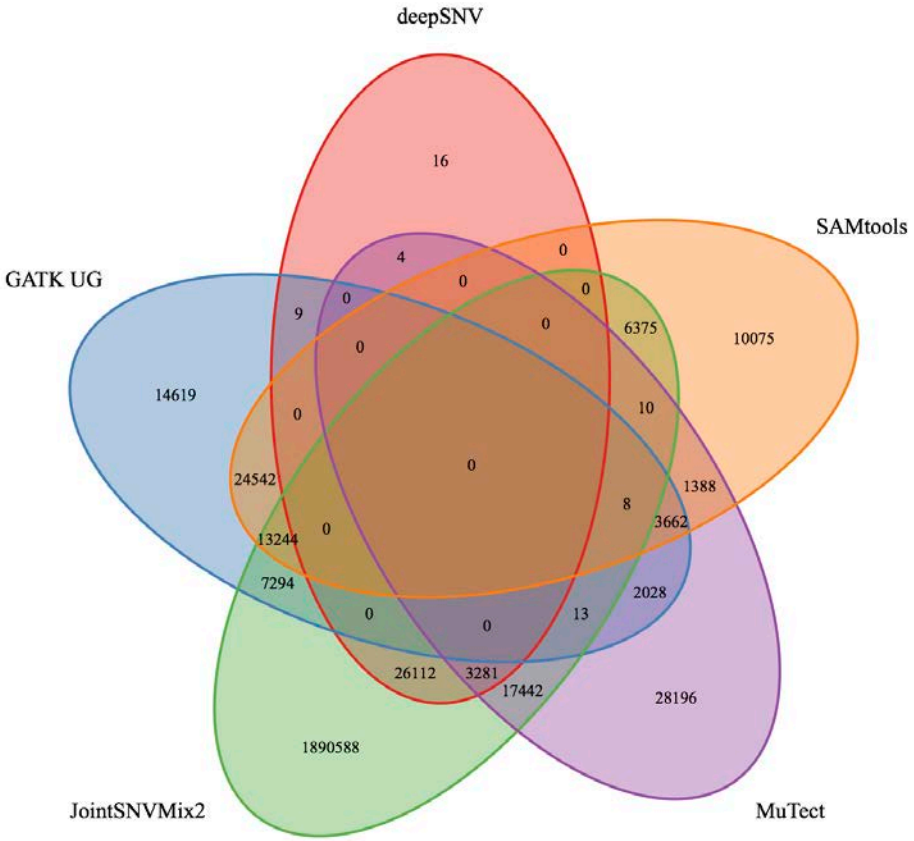
nexus Personalized Health Technologies

Variant call

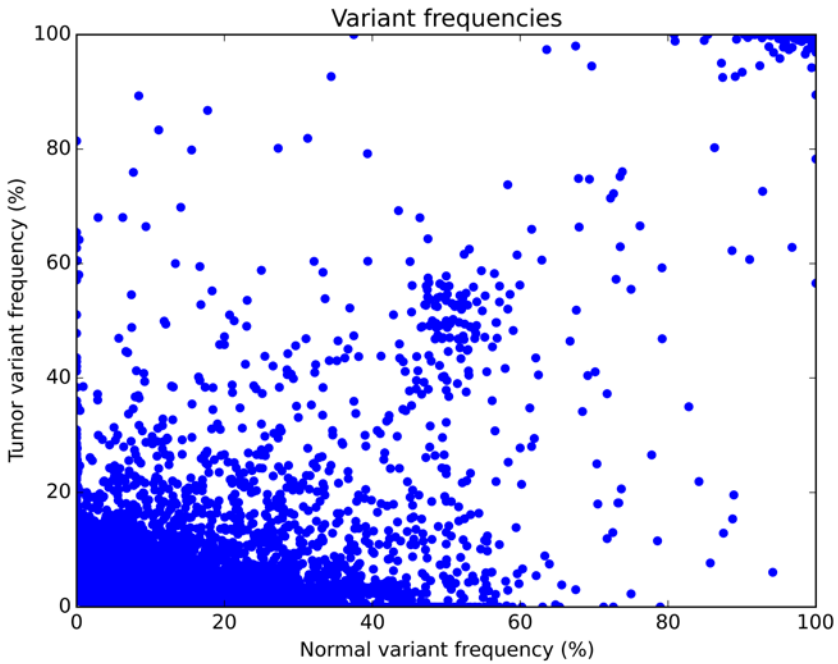
Getting some SNVs – easy!

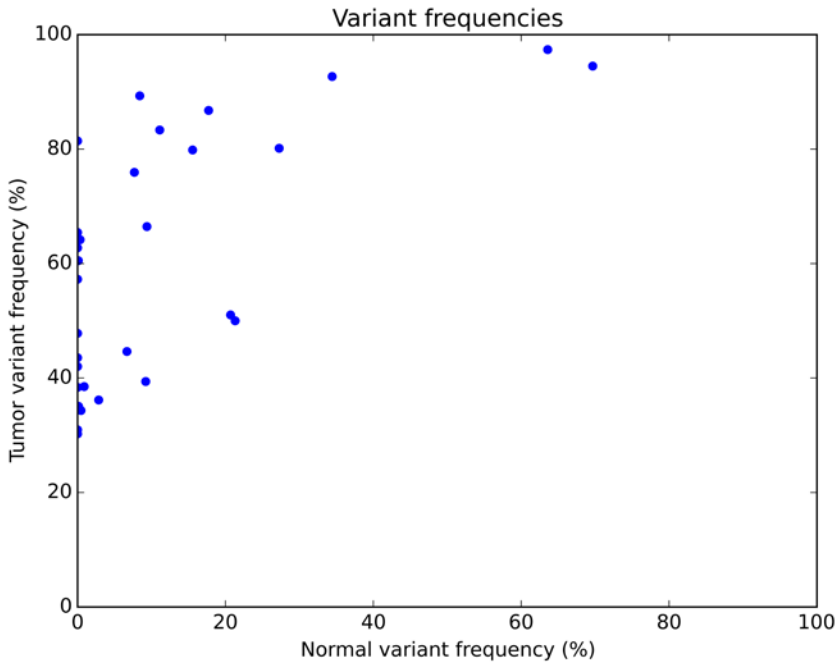
Getting the real SNVs – not so easy!

- Caller choice based on data type, benchmarks, experience, question….
- Raw calls need to carefully filtered and annotated in order to identify false positives



Impact of post-processing and quality control





Example: matched tumor normal pair; identify true somatic SNVs

Starting from the raw vcf…

Task 1:

(a)
Look at raw vcf files und use the provided script ***plotFreqPval_all.py*** to visualize the data.

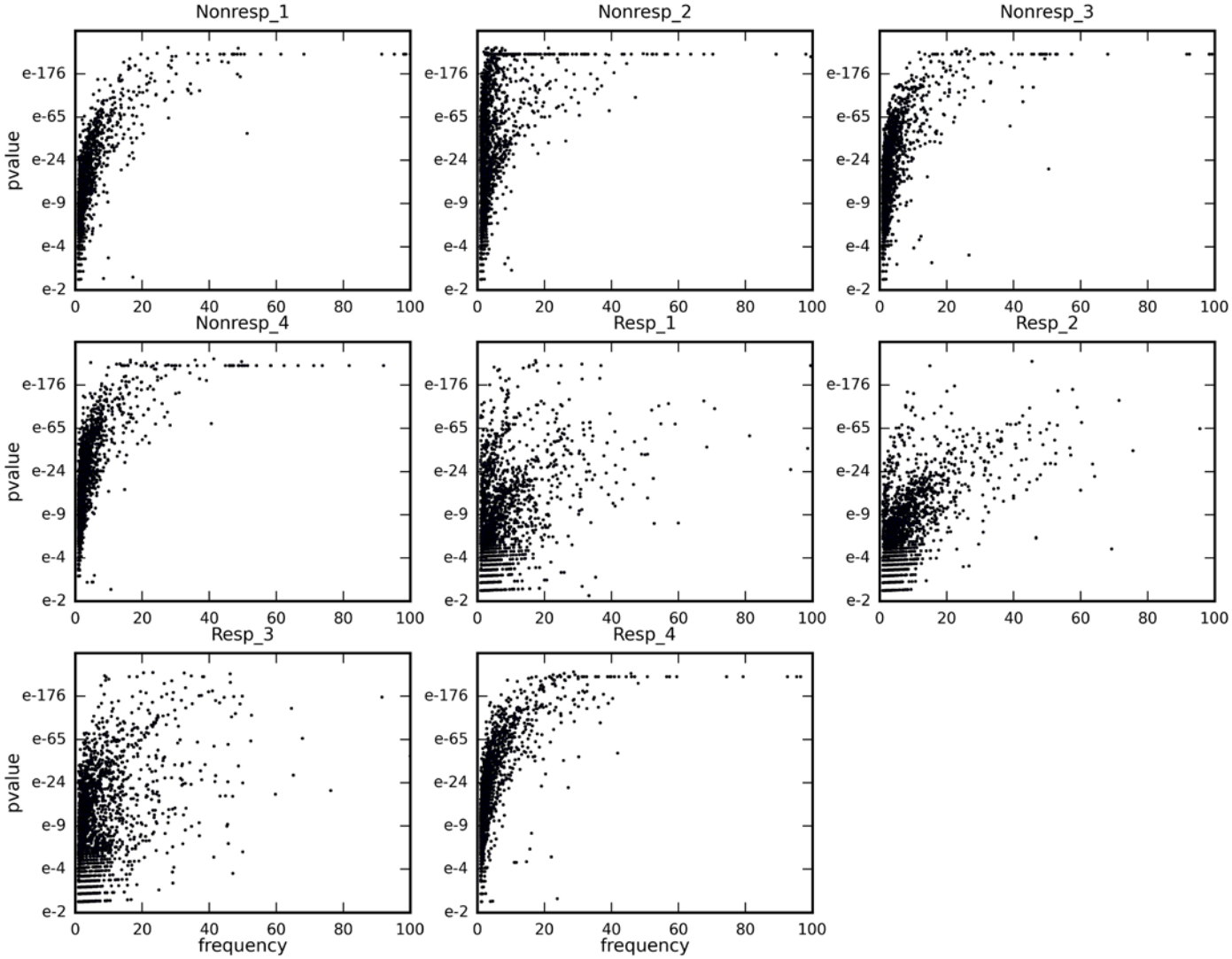
(b)
Discuss likely noise and the necessity of filtering.

(c)
Adapt the provided filtering script ***filterRawCalls_assignGenes.py*** to include different filters, visualize the filtered data and discuss the effects.

```
C:\Users\fzickman\Desktop\Workshop\dataset>python scripts\plotFreqPval_all.py -h

Compare variants based on pvalue vs frequency for vcf files in given directory.
Usage: python plotFreqPval_all.py [vcfDirec] [outfile]
```

Filtering


Nonresp_1
Nonresp_2
Nonresp_3
Nonresp_4
Resp_1
Resp_2
Resp_3
Resp_4
pvalue
e-176
e-65
e-24
e-9
e-4
e-2
0
20
40
60
80
100
frequency

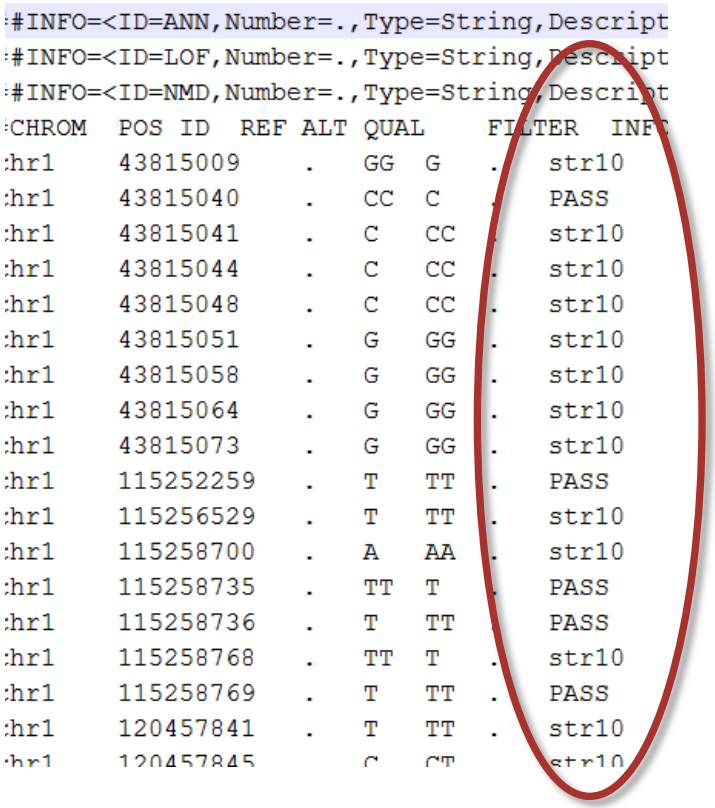
Filtering

Most variant callers already provide several filter options

Here:

- Exclude Wild type variants
- Exclude strand bias errors
- Exclude likely InDel errors

Open and change the file ***filterRawCalls_assignGenes.py*** in *Spyder* or a text editor.
Test your script using the console.



```
#INFO=<ID=ANN,Number=.,Type=String,Descript
#INFO=<ID=LOF,Number=.,Type=String,Descript
#INFO=<ID=NMD,Number=.,Type=String,Descript
CHROM   POS ID  REF ALT QUAL    FILTER  INFO
chr1    43815009    .   GG  G   .   str10
chr1    43815040    .   CC  C   .   PASS
chr1    43815041    .   C   CC  .   str10
chr1    43815044    .   C   CC  .   str10
chr1    43815048    .   C   CC  .   str10
chr1    43815051    .   G   GG  .   str10
chr1    43815058    .   G   GG  .   str10
chr1    43815064    .   G   GG  .   str10
chr1    43815073    .   G   GG  .   str10
chr1    115252259   .   T   TT  .   PASS
chr1    115256529   .   T   TT  .   str10
chr1    115258700   .   A   AA  .   str10
chr1    115258735   .   TT  T   .   PASS
chr1    115258736   .   T   TT  .   PASS
chr1    115258768   .   TT  T   .   str10
chr1    115258769   .   T   TT  .   PASS
chr1    120457841   .   T   TT  .   str10
chr1    120457845   .   C   CT      str10
```

```
C:\Users\fzickman\Desktop\Workshop\dataset>python scripts\filterRawCalls_assignG
enes.py -h
Filter the output of SNV calling by VarScan (non somatic mode).
Usage: python filterRawCalls_assignGenes.py [vcfFileDirec] [outfileDirec]
```

Filtering

```
#!/usr/bin/env python

'''
Filter the output of SNV calling by VarScan (non somatic mode).
Assign remaining mutations to the genes of interest.
Franziska Singer, August 2015
'''

import sys
import os

if "-h" in sys.argv[1]:
    print "Filter the output of SNV calling by VarScan (non somatic mode)."
    print "Usage: python filterRawCalls_assignGenes.py [vcfFileDirec] [outfileDirec]"
    sys.exit(1)

'''
main
'''

vcfFileDirec = sys.argv[1]  # read in the parameter for the input directory
outDirec = sys.argv[2]

```

General script information

Import required packages

Help message

Read in parameters

Filtering

```
40 for file in os.listdir(vcfFileDirec):
41     vcfFile = "%s%s" %(vcfFileDirec,os.path.basename(file))
42     if os.path.isfile(vcfFile) and vcfFile.endswith(".vcf"):  # check if the file exi
43         name = os.path.basename(file).split(".vcf")[0]
44         sampleDict[name] = [0] * len(genes)   # create new sample entry in dictionary, for each gene we ini
45         sampleNames.append(name)
46         allFiles += 1
47         infile = open(vcfFile,'r')
48         outName = "%s%s_filtered.vcf" %(outDirec,os.path.basename(file).split(".vcf")[0])
49         outfile = open(outName,'w')  # create the filtered file
50
51         allVariants = 0
52         filteredVariants = 0
53
54         for line in infile:
55             if line.startswith("#"):   # these lines only need to be written to the filtered file
56                 outfile.write(line)
57                 continue
58
59             allVariants += 1
60             lineSplit = line.strip().split("\t")
61             if ("INSERT_HERE" in line) or ("INSERT_HERE" in line) or ("INSERT_HERE" in line):   # there are
62                 filteredVariants += 1
63                 continue
64
65             annoTag = line.strip().split("\t")[7].split("ANN=")[1]        # this is necessary for the gene
66             geneName = annoTag.split("|")[3]                              # get the gene name and count the
67             sampleDict[name][dictGenes[geneName]] += 1
68             outfile.write(line)
69
70         infile.close()
71         outfile.close()
72         sumOfPercents += float((float(filteredVariants)/float(max(1,allVariants)))*100.0)  # necessary to cc
73
```

Parse vcf files present in given directory

Create filtered outfile

Go through lines of each sample, process lines with variant information

Variant to gene assignment and filter statistics

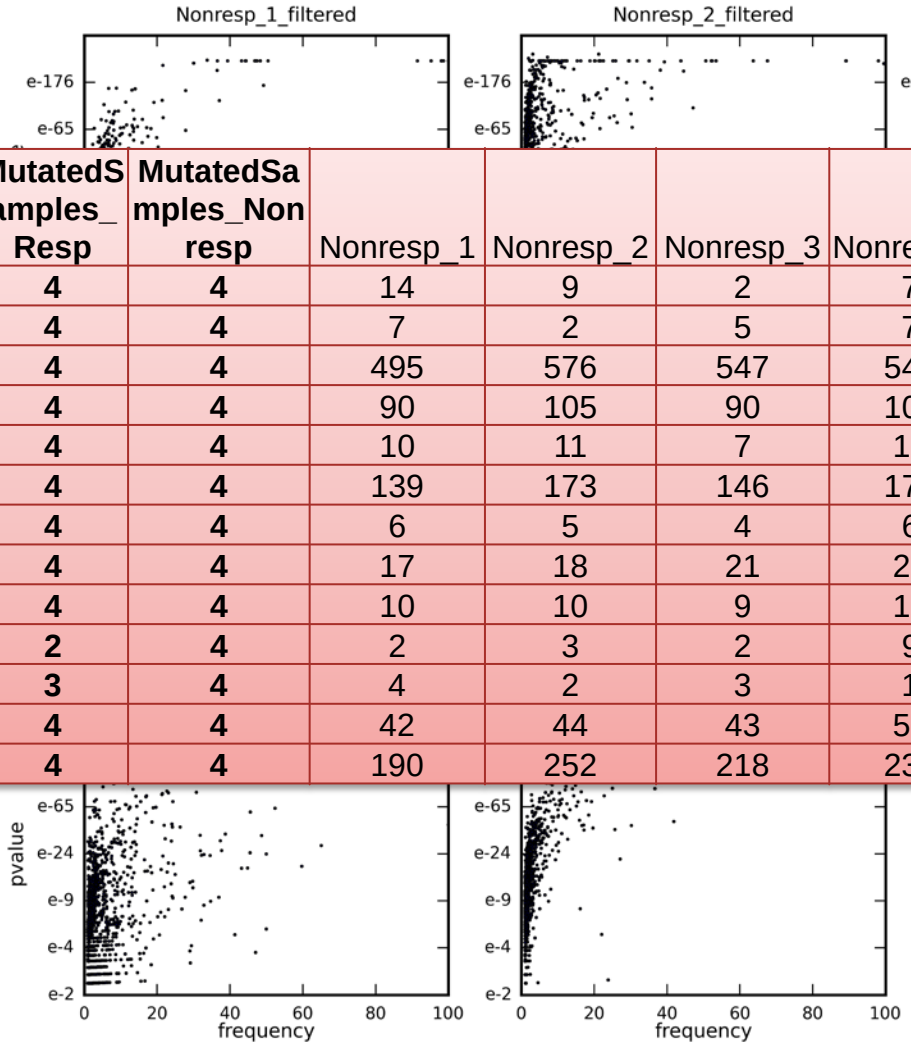
Filtering

```
58
59              allVariants += 1
60              lineSplit = line.strip().split("\t")
61              if ("WT=1" in line) or ("indelError" in line) or ("str10" in line):
62                  filteredVariants += 1
63                  continue
64
65              annoTag = line.strip().split("\t")[7].split("ANN=")[1]
```

```
C:\Users\fzickman\Desktop\Workshop\dataset>python scripts\filterRawCalls_assignG
enes.py patientsVCFs\ demo\defaultFilter\
Nonresp_1.vcf: Filtered 718 of 1744 variants (41.170%).
Nonresp_2.vcf: Filtered 782 of 1992 variants (39.257%).
Nonresp_3.vcf: Filtered 765 of 1862 variants (41.085%).
Nonresp_4.vcf: Filtered 758 of 1937 variants (39.133%).
Resp_1.vcf: Filtered 1154 of 2342 variants (49.274%).
Resp_2.vcf: Filtered 1188 of 2516 variants (47.218%).
Resp_3.vcf: Filtered 1364 of 2560 variants (53.281%).
Resp_4.vcf: Filtered 1018 of 1914 variants (53.187%).

On average filtered 45.451% of the variants.
```

Filtering



Gene	Mutated Samples All	MutatedSamples_Resp	MutatedSamples_Nonresp	Nonresp_1	Nonresp_2	Nonresp_3	Nonresp_4	Resp_1	Resp_2	Resp_3	Resp_4
MPL	**8**	**4**	**4**	14	9	2	7	1	1	2	5
NRAS	**8**	**4**	**4**	7	2	5	7	4	5	4	6
NOTCH2	**8**	**4**	**4**	495	576	547	541	565	670	637	407
IRF6	**8**	**4**	**4**	90	105	90	103	113	129	111	77
ALK	**8**	**4**	**4**	10	11	7	12	9	10	7	16
CASP8	**8**	**4**	**4**	139	173	146	174	147	189	153	120
IDH1	**8**	**4**	**4**	6	5	4	6	5	3	3	7
ERBB4	**8**	**4**	**4**	17	18	21	22	21	22	19	22
VHL	**8**	**4**	**4**	10	10	9	17	9	8	7	9
MLH1	**6**	**2**	**4**	2	3	2	9	0	1	0	2
CTNNB1	**7**	**3**	**4**	4	2	3	1	3	3	0	1
PIK3CA	**8**	**4**	**4**	42	44	43	50	49	46	41	50
TP63	**8**	**4**	**4**	190	252	218	230	262	241	212	174

nexus Personalized Health Technologies

Filtering

What other filter criteria are possible?

```
11
12 if "-h" in sys.argv[1]:
13     print "Filter the output of SNV calling by VarScan (non somatic mode)."
14     print "Usage: python filterRawCalls_assignGenes.py [vcfFileDirec] [outfileDirec] [filter1 y/n] [filter2 y\n]"
15     sys.exit(1)
16
17 '''
18 main
19 '''
20
21 vcfFileDirec = sys.argv[1]  # read in the parameter for the input directory
22 outDirec = sys.argv[2]
23 filter1 = sys.argv[3]
24 filter2 = sys.argv[4]
25
26 if not os.path.exists(outDirec):        # if the output directory does not exist, create it
27     os.makedirs(outDirec)
28
```

```
56         for line in infile:
57             if line.startswith("#"):   # these lines only need to be written to the filtered file
58                 outfile.write(line)
59                 continue
60
61             if "y" in filter1:
62                 # do stuff for filter 1
63
64             if "y" in filter2:
65                # do stuff for filter 2
66
```

nexus Personalized Health Technologies

Filtering

Frequency?

```
11
12 if "-h" in sys.argv[1]:
13     print "Filter the output of SNV calling by VarScan (non somatic mode)."
14     print "Usage: python filterRawCalls_assignGenes.py [vcfFileDirec] [outfileDirec] [filterFrequency]"
15     sys.exit(1)
16
17 '''
18 main
19 '''
20
21 vcfFileDirec = sys.argv[1]  # read in the parameter for the input directory
22 outDirec = sys.argv[2]
23 filterFrequency = float(sys.argv[3])
24
```

```
55         for line in infile:
56             if line.startswith("#"):  # these lines only need to be written to the filtered file
57                 outfile.write(line)
58                 continue
59
60             thisFreq = float(lineSplit[9].split(":")[6].split("%")[0])   # extract frequency, and in next li
61             if thisFreq <= filterFrequency:
62                 filteredVariants += 1
63                 continue
64
```

Filtering

Synonymous?

```
12 if "-h" in sys.argv[1]:
13     print "Filter the output of SNV calling by VarScan (non somatic mode)."
14     print "Usage: python filterRawCalls_assignGenes.py [vcfFileDirec] [outfileDirec] [filterFrequency] [filterSilent]"
15     sys.exit(1)
16
17 '''
18 main
19 '''
20
21 vcfFileDirec = sys.argv[1]  # read in the parameter for the input directory
22 outDirec = sys.argv[2]
23 filterFrequency = float(sys.argv[3])
24 filterSilent = sys.argv[4]
25
```

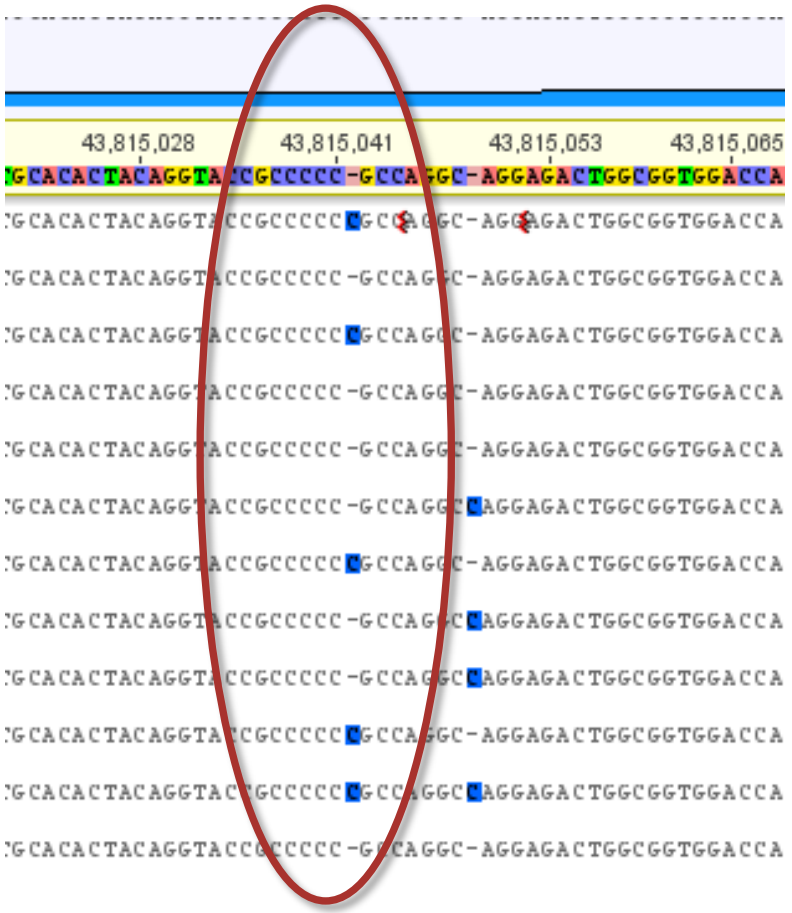
```
59
60             if "y" in filterSilent:
61                 # check if annotated as silent or synonymous
62                 if ("synonymous" in line) or ("silent" in line):
63                     filteredVariants += 1
64                     continue
65
```

Filtering

Homopolymer?

```
#INFO=<ID=ANN,Number=.,Type=String,Descript
#INFO=<ID=LOF,Number=.,Type=String,Descript
#INFO=<ID=NMD,Number=.,Type=String,Descript
CHROM   POS ID  REF ALT QUAL    FILTER  INFO
chr1    43815009    .   GG  G   .   str10
chr1    43815040    .   CC  C   .   PASS
chr1    43815041    .   C   CC  .   str10
chr1    43815044    .   C   CC  .   str10
chr1    43815048    .   C   CC  .   str10
chr1    43815051    .   G   GG  .   str10
chr1    43815058    .   G   GG  .   str10
chr1    43815064    .   G   GG  .   str10
chr1    43815073    .   G   GG  .   str10
chr1    115252259   .   T   TT  .   PASS
chr1    115256529   .   T   TT  .   str10
chr1    115258700   .   A   AA  .   str10
chr1    115258735   .   TT  T   .   PASS
chr1    115258736   .   T   TT  .   PASS
chr1    115258768   .   TT  T   .   str10
chr1    115258769   .   T   TT  .   PASS
chr1    120457841   .   T   TT  .   str10
chr1    120457845   .   C   CT  .   str10
```

Ion Torrent data are known to be error prone in homopolymer regions



nexus Personalized Health Technologies

Filtering

```
12 if "-h" in sys.argv[1]:
13     print "Filter the output of SNV calling by VarScan (non somatic
14     print "Usage: python filterRawCalls_assignGenes.py [vcfFileDire
15     sys.exit(1)
16
17 '''
18 functions
19 '''
20 def checkBases(allAllels):   # helper function that counts the numbe
21     baseArr = [0,0,0,0]  # number of A, C, T, G that is found in th
22     for thisAllel in allAllels:
23         if ("A" in thisAllel) or ("a" in thisAllel):
24             baseArr[0] += 1
25         if ("C" in thisAllel) or ("c" in thisAllel):
26             baseArr[1] += 1
27         if ("T" in thisAllel) or ("t" in thisAllel):
28             baseArr[2] += 1
29         if ("G" in thisAllel) or ("g" in thisAllel):
30             baseArr[3] += 1
31     return baseArr
32
33 '''
34 main
35 '''
```

Function definition

```
72         for line in infile:
73             if line.startswith("#"):   # these lines only need
74                 outfile.write(line)
75                 continue
76
77             if "y" in filterHomopolymer:
78                 refAllel = lineSplit[3]  # first get reference
79                 altAllel = lineSplit[4]
80
81                 baseArr = checkBases([refAllel,altAllel])  # c
82                 sumDiffBases = 0
83                 for baseTemp in baseArr:   # go through the ar
84                     if baseTemp > 0:
85                         sumDiffBases += 1
86                 if sumDiffBases == 1:   # only one entry in ar
87                     filteredVariants += 1
88                     continue
```

Include filter

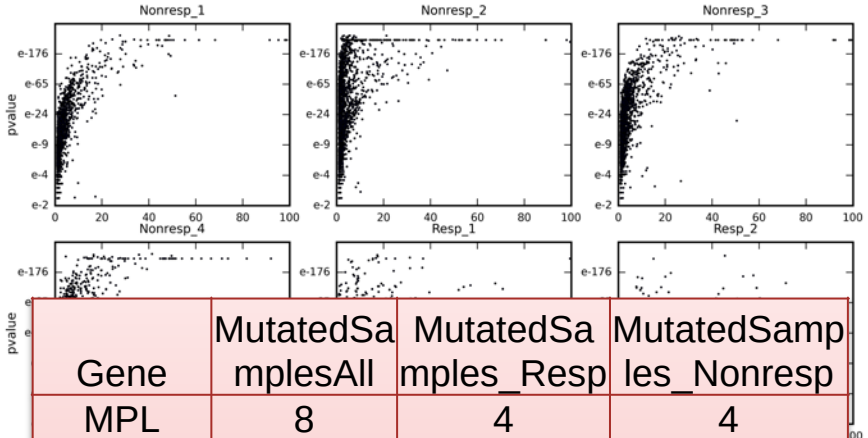
Filtering

Apply **default**, **synonomous**, and **homopolymer** filter, without filtering the frequency. (Frequency leads to loss of possible true positive mutations, e.g. c.464-27C>T in VHL in Resp1; overall amplicon coverage allows low frequency mutations)

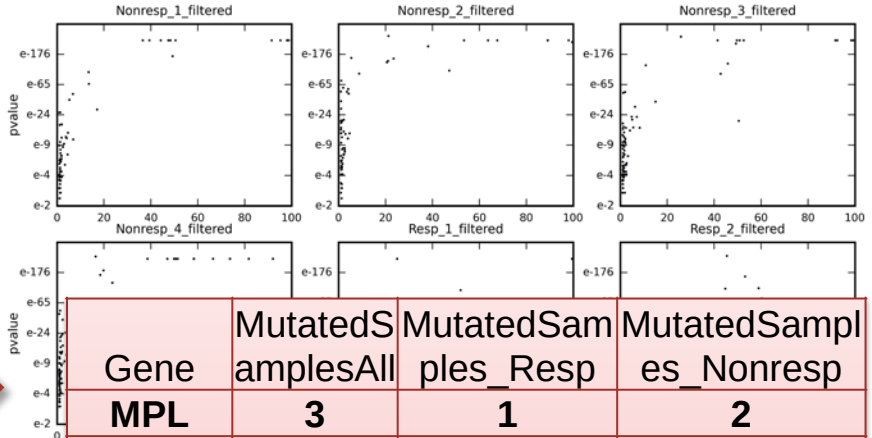
```
C:\Users\fzickman\Desktop\Workshop\dataset>python scripts\filterRawCalls_assignG
enes_complete.py patientsVCFs\ demo\allFilters_noFreq_noExtrastrand\ n y y 0
Nonresp_1.vcf: Filtered 1670 of 1744 variants (95.757%).
Nonresp_2.vcf: Filtered 1922 of 1992 variants (96.486%).
Nonresp_3.vcf: Filtered 1782 of 1862 variants (95.704%).
Nonresp_4.vcf: Filtered 1858 of 1937 variants (95.922%).
Resp_1.vcf: Filtered 2087 of 2342 variants (89.112%).
Resp_2.vcf: Filtered 2236 of 2516 variants (88.871%).
Resp_3.vcf: Filtered 2352 of 2560 variants (91.875%).
Resp_4.vcf: Filtered 1753 of 1914 variants (91.588%).

On average filtered 93.164% of the variants.
```

Filtering



Gene	MutatedSamplesAll	MutatedSamples_Resp	MutatedSamples_Nonresp
MPL	8	4	4
NRAS	8	4	4
NOTCH2	8	4	4
IRF6	8	4	4
ALK	8	4	4
CASP8	8	4	4
IDH1	8	4	4
ERBB4	8	4	4
VHL	8	4	4
MLH1	6	2	4
CTNNB1	7	3	4
PIK3CA	8	4	4
TP63	8	4	4



Gene	MutatedSamplesAll	MutatedSamples_Resp	MutatedSamples_Nonresp
MPL	**3**	**1**	**2**
NRAS	**1**	**1**	**0**
NOTCH2	8	4	4
IRF6	8	4	4
ALK	**4**	**1**	**3**
CASP8	8	4	4
IDH1	**1**	**1**	**0**
ERBB4	**4**	**4**	**0**
VHL	**5**	**2**	**3**
MLH1	2	1	1
CTNNB1	**3**	**2**	**1**
PIK3CA	6	3	3
TP63	8	4	4

Fisher's exact test

Fishers exact test is a **statistical significance test** to compare samples from two or more different categories. We test for a significant difference between the number of observations made for each category by computing the **probability** of the given observations. [Fisher, *J R Stat Soc*,1922]

	Responder	Nonresponder	Row total
Mutated	a	b	a+b
Not mutated	c	d	c+d
Column total	a+c	b+d	n=(a+b+c+d)

Probability *p* for such a set:

Observe *a* mutated samples

Observe *c* not mutated samples

$$p = \frac{\binom{a+b}{a}\binom{c+d}{c}}{\binom{n}{a+c}}$$

Proportion of responder samples

Fisher's exact test

Example:
4 responder samples, 4 nonresponder samples

Gene	Mutated samples responder	Mutated samples nonresponder
ALK	1	3

ALK	Responder	Nonresponder	Row Total
Mutated	1	3	4
Not mutated	3	1	4
Column total	4	4	8

$$p = \frac{\binom{4}{1}\binom{4}{3}}{\binom{8}{4}} = 0.49$$

Task 2:
Adapt the script ***performFisherTest.py*** and perform test on the overview table generated in Task 1. (Set correct entry for the ten *INSERT_HERE dummy values*)

Fisher's exact test

```
21
22 for line in infile:
23     lineSplit = line.strip().split("\t")
24     if line.startswith("Gene"):  # table header, extract information necessary for new table
25         for i in range(0,4):
26             outfileTable.write("%s\t" %(lineSplit[i]))
27         outfileTable.write("p-value\n")
28         continue
29     gene = lineSplit[0]
30     if (int(lineSplit[1]) == 0) or (int(lineSplit[1]) == 8):
31         # no test necessary, set p-value to 1.0
32         outfileTable.write("%s\t%s\t%s\t%s\t1.0\n" %(gene,lineSplit[1],lineSplit[2],lineSplit[3]))
33     else:
34         odds,pval = scp.fisher_exact([[int(lineSplit[2]), int(lineSplit[3])], [(4-int(lineSplit[2])), (4-int(lineSplit[3]))]])
35         outfileTable.write("%s\t%s\t%s\t%s\t%s\n" %(gene,lineSplit[1],lineSplit[2],lineSplit[3],pval))
36
37 infile.close()
38 outfileTable.close()
39
```

Fisher's exact test

Gene	MutatedSamples All	MutatedSamples _Resp	MutatedSamples _Nonresp	p-value
MPL	3	1	2	1.0
NRAS	1	1	0	1.0
NOTCH2	8	4	4	1.0
IRF6	8	4	4	1.0
ALK	4	1	3	0.48571
CASP8	8	4	4	1.0
IDH1	1	1	0	1.0
ERBB4	**4**	**4**	**0**	**0.02857**
VHL	5	2	3	1.0
MLH1	2	1	1	1.0
CTNNB1	3	2	1	1.0
PIK3CA	6	3	3	1.0
TP63	8	4	4	1.0

Issues:
Multiple testing correction still needs to be applied!
Sample size very small