GenePy v1.3

GenePy v1.3 is the latest version of GenePy which implements the following improvement from v1.2:

- GenePy now implements CADD v1.5 for the following reasons: scores both SNPs and INDELS, scores both coding and non-coding, works with both hg19 and hg38
- population frequencies are now obtained from gnomAD_exome but can be modified to use gnomAD_genome.
- novel variant frequency is set to 3.98e-6 (1 allele out 125,748 indiv in gnomADexome (251496 alleles))

GenePy requirements

- A (multi)sample VCF file (can accept compressed vcf.gz)
- List of genes for which generate GenePy scores. (gene.list)
- CADD v1.5 installed
- Vcftools
- Annovar (RefGene and gnomAD v.2.11 annotations)
- Python 2.7.x

How to run GenePy

Before running GenePy, we need to annotate SNVs and generate a GenePy-ready file (ALL_genepy.meta)

The first input required to GenePy is a multi-sample VCF (GENOTYPED_ALL.vcf.gz in this example) containing only BI-ALLELIC variants.

```
vcftools --gzvcf GENOTYPED_ALL.vcf.gz --min-alleles 2 --max-alleles 2 --recode --out FINAL # Keep
```

```
only Biallelic SNVs
```

Make annovar-ready file:

```
./annovar/convert2annovar.pl \
        -format vcf4 FINAL.recode.vcf.gz \
        -outfile ALL_genepy.input \
        -allsample \
        -withfreq \
        -include 2>annovar.log
```

Annotate against refGene,gnomad211_exome using Annovar

```
./annovar/table_annovar.pl \
        ALL_genepy.input \
        ./annovar/humandb/ \
        -buildver hg38 \
        -out ALL_genepy \
        -remove \
        -protocol refGene,gnomad211_exome \
        -operation g,f \
        --thread 40 \
        -nastring . >>annovar.log
```

Annotate VCF with CADD v1.5

For additional notes on how to install CADD please visit https://github.com/kircherlab/CADD-scripts

```
# first remove the header from the vcf and strip off the leading "chr" as required by CADD
zgrep -v "^#" FINAL.recode.vcf.gz >caddin.vcf
sed -i 's|^chr||g' caddin.vcf

# activate CADD environment
module load conda/py2-latest
source activate cadd-env-v1.5

# run CADD
./CADD-scripts/CADD.sh -g GRCh38 -v v1.5 -o caddout.tsv.gz caddin.vcf
```

Combine Genotypes and annotations

```
# extract the genotypes..
cut -f 18- ALL_genepy.input > a1
# and the sample headers
zgrep '^#CHR' FINAL.recode.vcf.gz | cut -f 10- > b1
cat b1 a1 > geneanno

# extract allele frequencies
cut -f 1,2,4,5,6,7,11 ALL_genepy.hg38_multianno.txt >freqanno
```

```
# transform/prepare output, this script automatically fixes missmatch between line in the vcf and
annotation
# if any fix fail, it will be prompted and replaced with "NAN". If NAN are not removed, GenePy
will fail. The cross-annotate-cadd.py script will always fail at positions chr6:75085419 and
chr17:6787257. The scores for these can be filled in manually by cross-checking with caddout.tsv
gunzip caddout.tsv.gz
python cross-annotate-cadd.py

# collate everything together
paste freqanno caddanno geneanno > ALL_genepy.meta

rm a1 b1 caddanno freqanno geneanno
```

Prepare output folders/dependencies

make new folders in your current directory to store raw GenePy score files

```
mkdir CADD15_RAW
```

Take the header from the ALL_genepy.meta file and stores it in a newly created header file

```
grep "^Chr" ALL_genepy.meta> header
```

If willing to use only exonic variants run the following

```
grep -E "exonic|splicing" ALL_genepy.meta > temp
cat header temp > ALL_genepy_exonic.meta
```

Compute GenePy scores

Once the ALL_genepy.meta file is created, GenePy_1.3.sh can be run by simply iterating through the list of deisred genes. Be aware, the make_scores_mat_6.py file **must** be in the same directory of GenePy_1.3.sh. WARNING If using the ALL_genepy_exonic.meta, replace the correct filename in the GenePy_1.3.sh file

```
#Create gene list
cut -f 7 ALL_genepy.hg38_multianno.txt | grep -v ";" | grep -v "Gene.refGene" | sort | uniq
>gene.list

#Run GenePy
while read gene:
do
sh GenePy_1.3.sh $gene ;
done< gene.list
```

Here a smart way of parallelise the generation of GenePy scores using the subber script:

```
# split the gene list in batches of 400 genes
split -d -l 400 gene.list batch_
#put all the batch names in a file
ls batch_* >parts
#check how many batches we have (52 in my case)
wc -l parts
#submit the job array
sbatch --array=1-52 subber.sh
```

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Contributors 5

Languages

- **Python** 84.0%
- **Shell** 16.0%

Suggested Workflows

Based on your tech stack

Actions Importer — Set up

Automatically convert CI/CD files to YAML for GitHub Actions.

SLSA Generic generator — Configure

Generate SLSA3 provenance for your existing release workflows

Python Package using Anaconda — Configure

Create and test a Python package on multiple Python versions using Anaconda for package management.

More workflows

Dismiss suggestions