Emilia Gnatiuk

Raport

Genetyka Ewolucyjna i Populacyjna

Na ćwiczeniach naszym celem było opracowanie metody bioinformatycznej za pomocą, której będziemy w stanie oszacować obciążenie genetyczne u biegusa łyżkodziobego oraz porównanie wyników z blisko spokrewnionym, ale stabilnym jeśli chodzi o populacje, biegusem rdzawoszyjim.

Bazowaliśmy na metodyce pracy Speak i in. 2024 Genomics-informed captive breeding can reduce inbreeding depression and the genetic load in zoo populations (Molecular Ecology Resources). Metoda opracowana w tym artykule służy właśnie do oszacowania obciążenia genetycznego specyficznie w genomach ptaków.

Każdy z nas dostał indywidualny scaffold do pracy – moim jest **scaffold 1**.

Wszyscy dostali także dane pochodzące z odczytów z sekwencjonowania osobnika biegusa lyżkodziobego oraz osobnika biegusa rdzawoszyjego.

Moje dane to:

* biegus łyżkodzioby - **C\_pyg\_22**,
* biegus rdzawoszyi - **C\_ruf\_08**.

### Materiały i metody

Wszystkie kroki analizy zostały wykonane na zewnętrznym serwerze. Dla ułatwienia korzystania z programów i narzędzi pracowaliśmy na wcześniej stworzonych środowiskach conda.

#Opis próbek:

SAMPLES=/home/mkonczal/Teaching/GEiP/Data/SamplesLibraries.txt

#Pliki FASTQ:

FASTQ\_DIR=/home/mkonczal/Teaching/GEiP/Data/Fastq

#Genom referencyjny biegusa łyżkodziobego: REF=/home/mkonczal/Teaching/GEiP/Data/Reference/SBS\_final.scaffolds.fasta

**Zapisanie sekwencji mojego scaffoldu**

seqkit grep -p "scaffold1" /home/mkonczal/Teaching/GEiP/Data/Reference/SBS\_final.scaffolds.fasta > scaffold1.fasta

Długość scaffoldu 1: 10631746

**Indeksowanie pliku FASTA**

samtools faidx scaffold1.fasta

cut -f 2 scaffold1.fasta.fai

### 

### Analiza jakości danych i ich czyszczenie

C\_ruf\_08 SRS3209983 SRR7054138

C\_ruf\_08 SRS3209983 SRR7054157

C\_pyg\_22 SRS3209985 SRR7054140

C\_pyg\_22 SRS3209985 SRR7054154

**Stworzenie symlinków:**

Analizy dla osobnika C\_pyg\_22:

run1

ln -s ${FASTQ\_DIR}/SRS3209985/SRR7054140\_pass\_1.fastq.gz .

ln -s ${FASTQ\_DIR}/SRS3209985/SRR7054140\_pass\_2.fastq.gz .

run2

ln -s ${FASTQ\_DIR}/SRS3209985/SRR7054154\_pass\_1.fastq.gz .

ln -s ${FASTQ\_DIR}/SRS3209985/SRR7054154\_pass\_2.fastq.gz .

Analizy dla osobnika C\_ruf\_08:

run1

ln -s ${FASTQ\_DIR}/SRS3209983/SRR7054138\_pass\_1.fastq.gz .

ln -s ${FASTQ\_DIR}/SRS3209983/SRR7054138\_pass\_2.fastq.gz .

run2

ln -s ${FASTQ\_DIR}/SRS3209983/SRR7054157\_pass\_1.fastq.gz .

ln -s ${FASTQ\_DIR}/SRS3209983/SRR7054157\_pass\_2.fastq.gz .

**Kontrola jakości**

Została ona wykonana za pomocą programu FastQC, który generuje raport oceny jakości surowych danych z sekwencjonowania NGS.

fastqc SRR7054140\_pass\_1.fastq.gz SRR7054140\_pass\_2.fastq.gz

fastqc SRR7054154\_pass\_1.fastq.gz SRR7054154\_pass\_2.fastq.gz

fastqc SRR7054138\_pass\_1.fastq.gz SRR7054138\_pass\_2.fastq.gz

fastqc SRR7054157\_pass\_1.fastq.gz SRR7054157\_pass\_2.fastq.gz

**Filtrowanie danych**

Za pomocą programu fastp, który automatycznie filtruje pliki fastq, co poprawia ich jakość.

fastp -i SRR7054140\_pass\_1.fastq.gz -I SRR7054140\_pass\_2.fastq.gz -o SRR7054140\_pass\_1.filt.fastq -O SRR7054140\_pass\_2.filt.fastq

fastp -i SRR7054154\_pass\_1.fastq.gz -I SRR7054154\_pass\_2.fastq.gz -o SRR7054154\_pass\_1.filt.fastq -O SRR7054154\_pass\_2.filt.fastq

fastp -i SRR7054138\_pass\_1.fastq.gz -I SRR7054138\_pass\_2.fastq.gz -o SRR7054138\_pass\_1.filt.fastq -O SRR7054138\_pass\_2.filt.fastq

fastp -i SRR7054157\_pass\_1.fastq.gz -I SRR7054157\_pass\_2.fastq.gz -o SRR7054157\_pass\_1.filt.fastq -O SRR7054157\_pass\_2.filt.fastq

**Ponowna kontrola jakości po filtrowaniu**

fastqc SRR7054140\_pass\_1.filt.fastq SRR7054140\_pass\_2.filt.fastq

fastqc SRR7054154\_pass\_1.filt.fastq SRR7054154\_pass\_2.filt.fastq

fastqc SRR7054138\_pass\_1.filt.fastq SRR7054138\_pass\_2.filt.fastq

fastqc SRR7054157\_pass\_1.filt.fastq SRR7054157\_pass\_2.filt.fastq

### Mapowanie odczytów do analizowanego kontigu

**Indeksowanie genomu referencyjnego**

Niezbędny krok przed mapowaniem do genomu referencyjnego. Tworzy zestaw plików indeksujących, które przyspieszają dopasowywanie odczytów NGS.

bwa index scaffold1.fasta

**Mapowanie odczytów do referencji i usuwanie niezmapowanych odczytów**

Mapujemy odczyty z sekwencjonowania NGS do genomu referencyjnego i filtrujemy tylko te odczyty, które są dopasowane.

bwa mem -t 10 -R '@RG\tID:SRR7054140\tSM:C\_pyg\_22\tLB: SRR7054140\tPL:ILLUMINA\tPU:lib1\_unit' scaffold1.fasta SRR7054140\_pass\_1.filt.fastq SRR7054140\_pass\_2.filt.fastq | samtools view -F 4 -o SRR7054140.Mapped.ba

bwa mem -t 10 -R '@RG\tID:SRR7054154\tSM:C\_pyg\_22\tLB: SRR7054154\tPL:ILLUMINA\tPU:lib1\_unit' scaffold1.fasta SRR7054154\_pass\_1.filt.fastq SRR7054154\_pass\_2.filt.fastq | samtools view -F 4 -o SRR7054154.Mapped.ba

bwa mem -t 10 -R '@RG\tID:SRR7054138\tSM:C\_ruf\_08\tLB: SRR7054138\tPL:ILLUMINA\tPU:lib1\_unit' scaffold1.fasta SRR7054138\_pass\_1.filt.fastq SRR7054138\_pass\_2.filt.fastq | samtools view -F 4 -o SRR7054138.Mapped.ba

bwa mem -t 10 -R '@RG\tID:SRR7054157\tSM:C\_ruf\_08\tLB: SRR7054157\tPL:ILLUMINA\tPU:lib1\_unit' scaffold1.fasta SRR7054157\_pass\_1.filt.fastq SRR7054157\_pass\_2.filt.fastq | samtools view -F 4 -o SRR7054157.Mapped.ba

### Sortowanie, markowanie duplikatów i indeksowanie plików

**Sortowanie plików BAM**

samtools sort -T bam SRR7054140.Mapped.bam > SRR7054140.Mapped.sorted.bam

samtools sort -T bam SRR7054154.Mapped.bam > SRR7054154.Mapped.sorted.bam

samtools sort -T bam SRR7054138.Mapped.bam > SRR7054138.Mapped.sorted.bam

samtools sort -T bam SRR7054157.Mapped.bam > SRR7054157.Mapped.sorted.bam

**Usunięcie duplikatów**

Usunięcie zduplikowanych odczytów z sekwencjonowania.

picard MarkDuplicates REMOVE\_DUPLICATES=true REMOVE\_SEQUENCING\_DUPLICATES=true AS=true I=SRR7054140.Mapped.sorted.bam M=test.metric\_SRR7054140.txt O=SRR7054140.Mapped.sorted\_DupRmv.bam 2> MarkDup\_SRR7054140.log

picard MarkDuplicates REMOVE\_DUPLICATES=true REMOVE\_SEQUENCING\_DUPLICATES=true AS=true I=SRR7054154.Mapped.sorted.bam M=test.metric\_SRR7054154.txt O=SRR7054154.Mapped.sorted\_DupRmv.bam 2> MarkDup\_SRR7054154.log

picard MarkDuplicates REMOVE\_DUPLICATES=true REMOVE\_SEQUENCING\_DUPLICATES=true AS=true I=SRR7054138.Mapped.sorted.bam M=test.metric\_SRR7054138.txt O=SRR7054138.Mapped.sorted\_DupRmv.bam 2> MarkDup\_SRR7054138.log

picard MarkDuplicates REMOVE\_DUPLICATES=true REMOVE\_SEQUENCING\_DUPLICATES=true AS=true I=SRR7054157.Mapped.sorted.bam M=test.metric\_SRR7054157.txt O=SRR7054157.Mapped.sorted\_DupRmv.bam 2> MarkDup\_SRR7054157.log

**Indeksowanie plików BAM**

Tworzymy indeks dla pliku BAM, który umożliwia szybki dostęp do odczytów w dowolnym regionie.

samtools index SRR7054140.Mapped.sorted\_DupRmv.bam

samtools index SRR7054154.Mapped.sorted\_DupRmv.bam

samtools index SRR7054138.Mapped.sorted\_DupRmv.bam

samtools index SRR7054157.Mapped.sorted\_DupRmv.bam

Wygenerowanie podsumowania właściwości zmapowania zakodowanego we flagach:

samtools flagstats SRR7054140.Mapped.sorted\_DupRmv.bam

samtools flagstats SRR7054154.Mapped.sorted\_DupRmv.bam

samtools flagstats SRR7054138.Mapped.sorted\_DupRmv.bam

samtools flagstats SRR7054157.Mapped.sorted\_DupRmv.bam

**Połączenie plików BAM:**

samtools merge –r C\_pyg\_22.bam SRR7054140.Mapped.sorted\_DupRmv.bam SRR7054154.Mapped.sorted\_DupRmv.bam

samtools merge –r C\_ruf\_08.bam SRR7054138.Mapped.sorted\_DupRmv.bam SRR7054157.Mapped.sorted\_DupRmv.bam

samtools index C\_pyg\_22

samtools index C\_ruf\_08

samtools tview --reference scaffold1.fasta C\_pyg\_22.bam

interpretacja:

Większość sekwencji pokrywa się idealnie z referencją, widać pojedyncze kropki, czyli miejsca, w których referencja różni się od nukleotydów - może to sugerować mutacje bądź błędy sekwencjonowania. Widać też regiony z przecinkami - te sugerują występowanie wariantów na nici komplementarnej.

samtools tview --reference scaffold1.fasta C\_ruf\_08.bam

interpretacja:

Większość sekwencji pokrywa się idealnie z referencją. Są pozycje, w których pojawia się C zamiast nukleotydu referencyjnego, co może sugerować SNP lub błędy sekwencjonowania. Widać też regiony z przecinkami i c- te sugerują występowanie wariantów na nici komplementarnej. Puste miejsca to miejsca o niskim pokryciu, możliwe błędy mapowania.

**Konwersja pliku FASTA do 2bit**

2bit to skompresowany format do przechowywania sekwencji DNA. Każdy nukleotyd zajmuje 2 bity, zajmuje mniej miejsca niż fasta.

faToTwoBit /home/st3/GEiP/lab1/scaffold1.fasta scaffold1.2bit

**Zapisanie wielkości scaffoldu do pliku**

twoBitInfo scaffold1.2bit sizes.tab

długość: 10631746

Skopiowanie pliku zawierającego elementy ultrakonserwatywne do bieżacego katalogu:

cp /home/mkonczal/Teaching/GEiP/Data/UCE-probes/uce-5k-probes.fasta .

**Indeksowanie pliku z UCE (elementami ultrakonserwatywne)**

samtools faidx uce-5k-probes.fasta bwa index uce-5k-probes.fasta UCEprobe=/home/mkonczal/Lab2/uce-5k-probes.fasta

**Przyrównanie elementów UCE do genomu biegusa**

UCE to ultrakonserwowane elementy DNA, czyli regiony genomowe, które są niemal identyczne u różnych gatunków. Kodują elementy związane z podstawowymi funkcjami organizmów. Przyrównując je do genomu biegusa i znajdując w tych regionach mutacje jesteśmy w stanie zidentyfikować polimorfizmy i możemy założyć, że te polimorfizmy są w dużej mierze szkodliwe i potencjalnie chorobotwórcze.

phyluce\_probe\_run\_multiple\_lastzs\_sqlite --db scaffold1.sqlite \ --output scaffold1-genome-lastz \ --scaffoldlist scaffold1 \ --genome-base-path ./ \ --probefile ${UCEprobe} \ --cores 5

**Wyodrębnienie sekwencji UCE z genomu biegusa zachowując 1000 zasad sąsiadujących z UCE**

phyluce\_probe\_slice\_sequence\_from\_genomes --lastz scaffold1-genome-lastz -flank 1000 --output OUT --conf scaffold1.conf --name-pattern uce-5k probes.fasta\_v\_scaffold1.lastz.clean

**Określenie orientacji każdego z UCE na podstawie nagłówka i zapianie go w pliku**

grep "uce" scaffold1.fasta | cut -d '|' -f 2,3,4,6 | sed -e 's/|/\t/g' | sed -e 's/contig://g' | sed -e 's/slice://g'| sed -e 's/uce://g' | sed -e 's/orient://g' | sed -e 's/uce-/uce\_/g' | sed -e s/"'"//g | sed -e 's/{+}/forward/g' | sed -e 's/{-}/reverse/g'| sed -e 's/-/\t/g' > /home/st3/GEiP/lab2/UCE\_regions/scaffold1\_UCE\_regions.txt

**Zapisanie osobno UCE w orientacji forward i reverse**

W moim przypadku wszystkie znajdują się na reverse.

grep 'forward' UCE\_regions/scaffold1\_UCE\_regions.txt | cut -f 1,2,3 > UCE\_regions/forward/scaffold1\_UCE\_forward\_orient\_regions.txt

grep 'reverse' UCE\_regions/scaffold1\_UCE\_regions.txt | cut -f 1,2,3 > UCE\_regions/reverse/scaffold1\_UCE\_reverse\_orient\_regions.txt

SNP calling - w moim przypadku tylko dla nici reverse

SNP calling to proces w którym identyfikujemy pojedyncze polimorfizmy pojedynczych nukleotydów w genomie.

UCE=UCE\_regions/forward/scaffold1\_UCE\_reverse\_orient\_regions.txt REF=/home/st3/GEiP/lab1/scaffold1.fasta

BAM\_pyg=/home/st3/GEiP/lab1/C\_pyg\_22.bam

UCE=UCE\_regions/forward/scaffold1\_UCE\_reverse\_orient\_regions.txt REF=/home/st3/GEiP/lab1/scaffold1.fasta

BAM\_ruf=/home/st3/GEiP/lab1/C\_ruf\_08.bam

samtools mpileup -l ${UCE} -f ${REF} ${BAM\_pyg} > out.reverse.mpileup

samtools mpileup -l ${UCE} -f ${REF} ${BAM\_ruf} > out.reverse.mpileup

Za pomocą programu bcftools call wywołujemy SNP.

bcftools mpileup --threads 5 -Ou -Q 30 -q 30 -C 50 -a AD,DP -R ${UCE} -f ${REF} ${BAM\_pyg} | bcftools call --threads 10 -c -Ob > out.reverse\_pyg\_22.bcf

bcftools mpileup --threads 5 -Ou -Q 30 -q 30 -C 50 -a AD,DP -R ${UCE} -f ${REF} ${BAM\_ruf} | bcftools call --threads 10 -c -Ob > out.reverse\_ruf\_08.bcf

Do pliku bam\_list.txt zapisujemy w osobnych liniach lokalizacje obu analizowanych plików bam

bcftools mpileup --threads 10 -Ou -Q 30 -q 30 -C 50 -a AD,DP -R ${UCE} -f ${REF} -b bam\_list.txt | bcftools call --threads 10 -c -Ob > out.reverse\_2samples.bcf

Zadanie

Sprawdź liczbę linii w plikach out.forward\_2samples.bcf i out.forward.bcf. Czy jest ona identyczna? Sprawdź rozmiar obu plików i określ ilość Mb danych na osobnika w obu.

17182 out.reverse\_2samples.bcf 3 248 KB

6537 out.reverse.bcf 2 007 KB

-rw-rw-r-- 1 st3 st3 4M Jan 23 10:43 out.reverse\_2samples.bcf

-rw-rw-r-- 1 st3 st3 2M Jan 23 10:36 out.reverse.bcf

Liczba jest identyczna, ponieważ w out.reverse\_2samples.bcf są dwie próbki.

**Filtrowanie plików vcf**

Przed filtrowaniem wygenerowanie tabeli, z informacjami o prawdopodobieństwie, że dane miejsce jest polimorficzne (QUAL), średniej jakości mapowania (MQ) oraz sumarycznym pokryciu (DP) w każdym z miejscu.

bcftools view -v snps -e 'QUAL < 60 || MQ < 30 || FORMAT/DP < 4 || FORMAT/ DP > 20' out.reverse.bcf > out.reverse.filtered.vcf

galgal=/home/mkonczal/Teaching/GEiP/Data/galGal6

scaffold=scaffold1.fasta

### Przyrównanie genomu biegusa do genomu kury domowej

**Wyszukanie homologicznego chromosomu kury względem analizowanego contigu**

Zidentyfikowanie, który chromosom kury domowej jest homologiczny względem analizowanego scaffoldu biegusa.

blastn -query ${scaffold} -db ${galgal}/Gallus\_gallus.GRCg6a.dna\_rm.toplevel.fa -outfmt 6 > Scaffold1Chicken.blastout

cut –f2 Scaffold2Chicken.blastout

Sprawdzenie do którego chromosomu kury domowej przyrównuje się sekwencja z analizowanego scaffoldu

Policzenie liczby wystąpień każdego chromosomu

cut –f2 Scaffold2Chicken.blastout | sort | uniq -c

U mnie nawięcej **chromosomu 5**.

Przyrównanie analizowanego scaffoldu do homologicznego chromosomu kury

Za pomocą programu lastz, który służy porównywaniu długich sekwencji.

scores=/home/mkonczal/Teaching/GEiP/utilities/HoxD55

hom\_chicken\_chr=5

lastz ${galgal}/split/${hom\_chicken\_chr}.fa ${scaffold} --ambiguous=iupac --hspthresh=2200 --inner=2000 --ydrop=3400 --gappedthresh=10000 - scores=${scores} --chain --format=axt > bGalGal6\_chr${hom\_chicken\_chr}.axt

**Tworzenie łańcucha (chaining)**

Przygotowanie plików wejściowych do tworzenia łańcucha

alignment=bGalGal6\_chr${hom\_chicken\_chr}.axt

chicken\_2bit=${galgal}/Gallus\_gallus.GRCg6a.dna\_rm.toplevel.2bit

faToTwoBit scaffold1.fasta scaffold1.2bit

biegus\_2bit=scaffold1.2bit

axtChain -minscore=5000 -linearGap=loose $alignment $chicken\_2bit $biegus\_2bit bgalgalChr${hom\_chicken\_chr}\_scaff1.chain

Posortowanie pliku chain:

chainSort bgalgalChr${hom\_chicken\_chr}\_scaff1.chain sorted\_bgalgalChr${hom\_chicken\_chr}\_scaff1.chain

grep “chain” sorted\_bgalgalChr${hom\_chicken\_chr}\_scaff1.chain | wc -l

1666

**Filtrowanie łańcuchów**

chainNet sorted\_bgalgalChr${hom\_chicken\_chr}\_scaff1.chain 5.chrom.size scaffold1.chrom.size all.net /dev/null

netChainSubset all.net sorted\_bgalgalChr${hom\_chicken\_chr}\_scaff1.chain galGalChr ${hom\_chicken\_chr}ToSBS\_Scaff1.over.chain

Kompresja przefiltrowanego pliku

gzip galGalChr ${hom\_chicken\_chr}ToSBS\_Scaff1.over.chain

### CADD

chCADD\_dir=/home/mkonczal/Teaching/GEiP/Data/chCADD-scores

**Kopiowanie plików chCADD**

cp ${chCADD\_dir}/Header.txt .

cp ${chCADD\_dir}/5.txt.gz .

cat Header.txt > chr5\_chCADD.tsv

zcat 5.txt.gz >> chr5\_chCADD.tsv

**Zmiana pliku tsv na plik bed**

awk '{print $1,$2-1,$2,$3,$4,$5}’ chr5\_chCADD.tsv > chr5\_chCADD.1based.bed

### CrossMap

Crossmap to program używany do konwersji koordynatów pomiędzy różnymi wersjami genomu.

conda activate crossmap

scaffold=scaffold2

CrossMap bed galGalChr${hom\_chicken\_chr}ToSBS\_Scaff1.over.chain.gz chCADD\_chr/chr${hom\_chicken\_chr}\_chCADD.1based.bed | head

Zapisanie w pliku wynikowym tylko zmapowanych pozycji, ponieważ większość z nich nie została zmapowana do genomu kury

CrossMap bed galGalChr${hom\_chicken\_chr}ToSBS\_Scaff1.over.chain.gz chCADD/chr${hom\_chicken\_chr}\_chCADD.1based.bed | grep $scaffold | grep -v „Unmap“ | cut -f 3,4,5,6,7,8 > chr${hom\_chicken\_chr}-SBS\_CADD.bed

**Utworzenie zmiennych do plików VCF**

VCF1\_r=/home/st3/GEiP/lab2/out.reverse\_pyg\_22.filtered.vcf

VCF2\_r=/home/st3/GEiP/lab2/out.reverse\_ruf\_08.filtered.vcf

**Przekonwertowanie pliku VCF do BED**

vcf2bed --max-mem 4G < ${VCF1\_r} > vcf\_C\_pyg\_22\_reverse.bed

vcf2bed --max-mem 4G < ${VCF2\_f} > vcf\_C\_ruf\_08\_reverse.bed

**Usunięcie indeli z plików BED**

grep -v "INDEL" vcf\_C\_pyg\_22\_reverse.bed > vcf\_C\_pyg\_22\_reverse\_indelRm.bed

grep -v "INDEL" vcf\_C\_ruf\_08\_reverse.bed > vcf\_C\_ruf\_08\_reverse\_indelRm.bed

**Intersekcja**

CADD=/home/st3/GEiP/lab3/chr5-SBS\_CADD.bed

bed=vcf\_C\_pyg\_22\_reverse\_indelRm.bed

CADD=/home/st3/GEiP/lab3/chr5-SBS\_CADD.bed

bed=vcf\_C\_ruf\_08\_reverse\_indelRm.bed

bedtools intersect -a $CADD -b $bed -wa -wb >

vcf\_C\_pyg\_22\_reverse\_indelRm\_intersect.bed

bedtools intersect -a $CADD -b $bed -wa -wb > vcf\_C\_ruf\_08\_reverse\_indelRm\_intersect.bed

Nie wykonałam adnotacji SNPów, ponieważ mam tylko pliki reverse.

Przekonwertowanie allelu referencyjnego T na A w celu dopasowania go do genomu kury.

cript\_path= /home/mkonczal/Teaching/GEiP/scripts

awk -v b=6 -v e=100 -f ${script\_path}/SNP\_check\_reverse.awk vcf\_C\_pyg\_22\_reverse\_indelRm\_intersect.bed > vcf\_C\_pyg\_22\_reverse\_indelRm\_intersect\_OrientedReverse.bed

awk -v b=6 -v e=100 -f ${script\_path}/SNP\_check\_reverse.awk vcf\_C\_pyg\_22\_reverse\_indelRm\_intersect\_OrientedReverse.bed > vcf\_C\_pyg\_22\_reverse\_indelRm\_intersect\_OrientedReverse\_annotated.bed

awk -v b=6 -v e=100 -f ${script\_path}/SNP\_check\_reverse.awk vcf\_C\_ruf\_08\_reverse\_indelRm\_intersect.bed > vcf\_C\_ruf\_08\_reverse\_indelRm\_intersect\_OrientedReverse.bed

awk -v b=6 -v e=100 -f ${script\_path}/SNP\_check\_reverse.awk vcf\_C\_ruf\_08\_reverse\_indelRm\_intersect\_OrientedReverse.bed > vcf\_C\_ruf\_08\_reverse\_indelRm\_intersect\_OrientedReverse\_annotated.bed

awk '$23 == "SNP\_is\_ALT\_pp=ref"'

vcf\_C\_pyg\_22\_reverse\_indelRm\_intersect\_OrientedReverse\_annotated.bed > vcf\_C\_pyg\_22\_reverse\_indelRm\_intersect\_OrientedReverse\_annotated\_SNP.bed

awk -e ' $20 ~ /^0\/0/ ' vcf\_C\_pyg\_22\_reverse\_indelRm\_intersect\_OrientedReverse\_annotated\_SNP.bed | cut -f 6 | paste -sd+ - | bc

awk -e ' $20 ~ /^0\/0/ ' vcf\_C\_ruf\_08\_reverse\_indelRm\_intersect\_OrientedReverse\_annotated\_SNP.bed | cut -f 6 | paste -sd+ - | bc

awk -e ' $20 ~ /^0\/1/ ' vcf\_C\_pyg\_22\_reverse\_indelRm\_intersect\_OrientedReverse\_annotated\_SNP.bed | cut -f 6 | paste -sd+ - | bc

awk -e ' $20 ~ /^0\/1/ ' vcf\_C\_ruf\_08\_reverse\_indelRm\_intersect\_OrientedReverse\_annotated\_SNP.bed | cut -f 6 | paste -sd+ - | bc

awk -e ' $20 ~ /^1\/1/ ' vcf\_C\_pyg\_22\_reverse\_indelRm\_intersect\_OrientedReverse\_annotated\_SNP.bed | cut -f 6 | paste -sd+ - | bc

awk -e ' $20 ~ /^1\/1/ ' vcf\_C\_ruf\_08\_reverse\_indelRm\_intersect\_OrientedReverse\_annotated\_SNP.bed | cut -f 6 | paste -sd+ - | bc

1. Ile wariantów zidentyfikowano w genomie każdego z dwóch gatunków?

C\_pyg: 654 wariantów

C\_ruf: 2031 wariantów

2. Ile homozygot i heterozygot zidentyfikowano?

**C\_ruf: 390 heterozygot, 1641 homozygot**

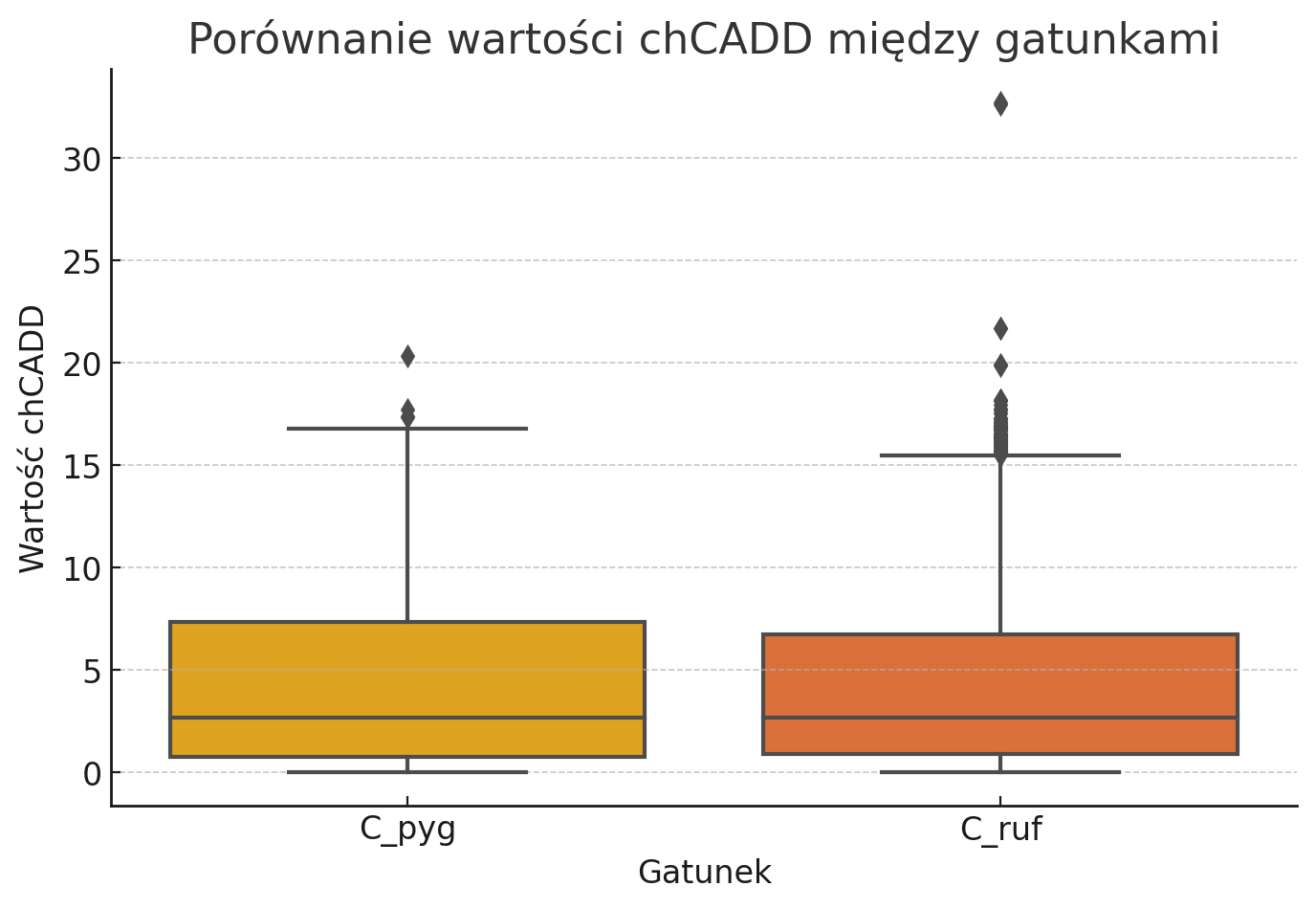
**C\_pyg: 438 heterozygot, 216 homozygot**

3. Czy średnia wartość oszacowań CADD różni się między gatunkami?

C\_pyg: **4.82**

C\_ruf: **4.51**

**Różni się.**

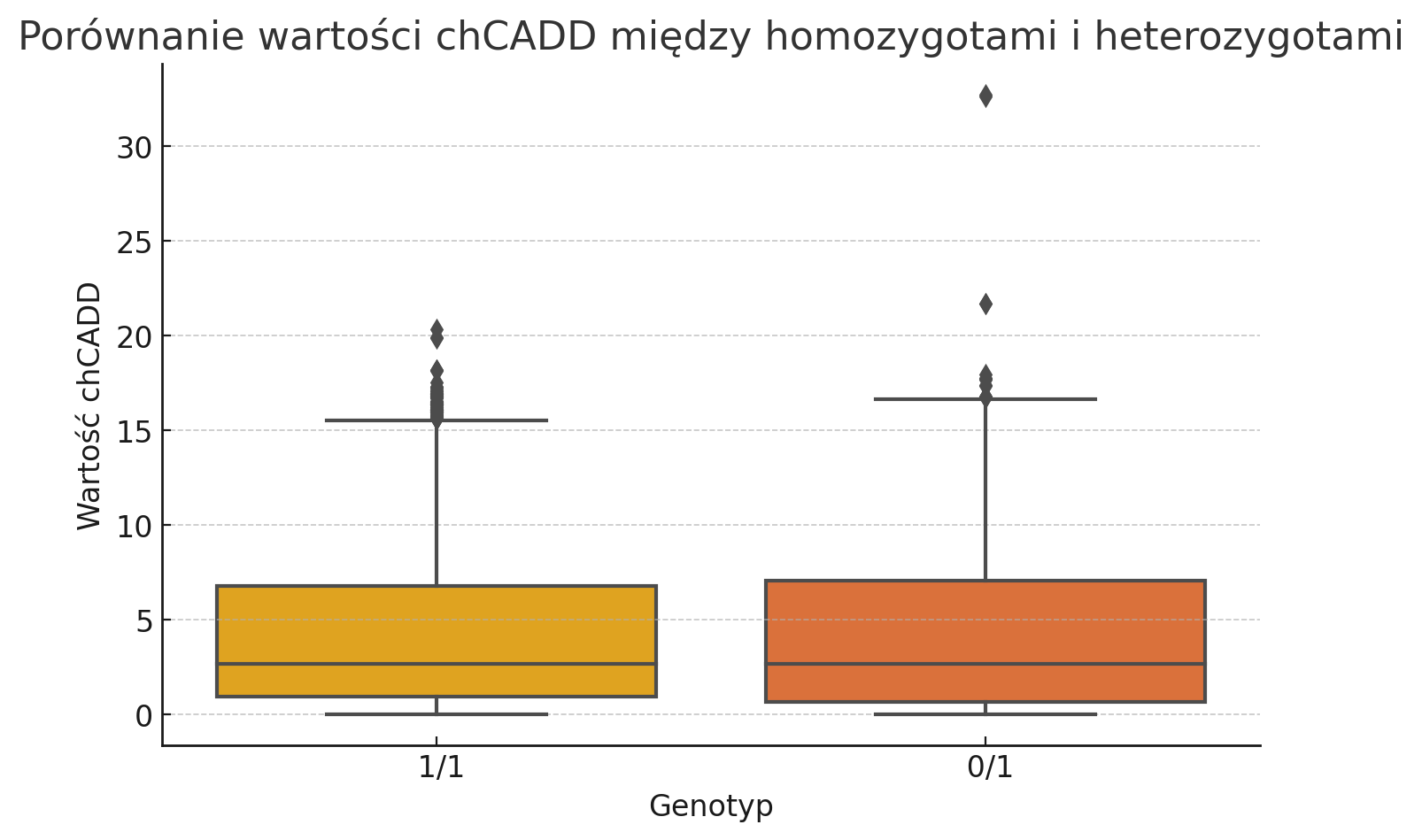


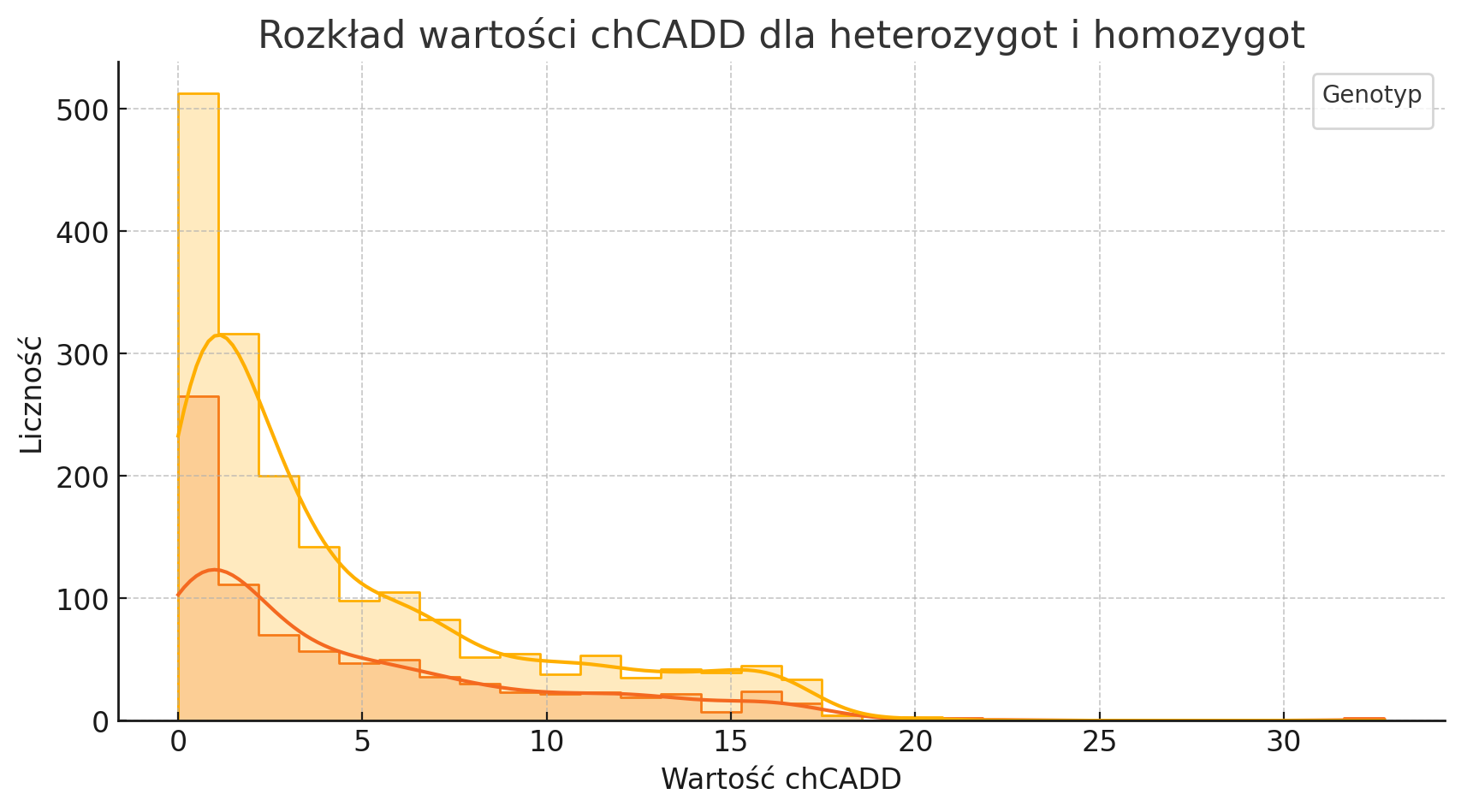
4. Czy średnia wartość oszacowań CADD różni się między heterozygotami i homozygotami?

**heterozygoty - 4.66**

**homozygoty - 4.56**

**Różni się.**





5. Oszacuj całkowite oraz zrealizowane obciążenie genetyczne dla obu gatunków. Weź pod uwagę współczynnik dominacji równy 0.1.

**C\_pyg**

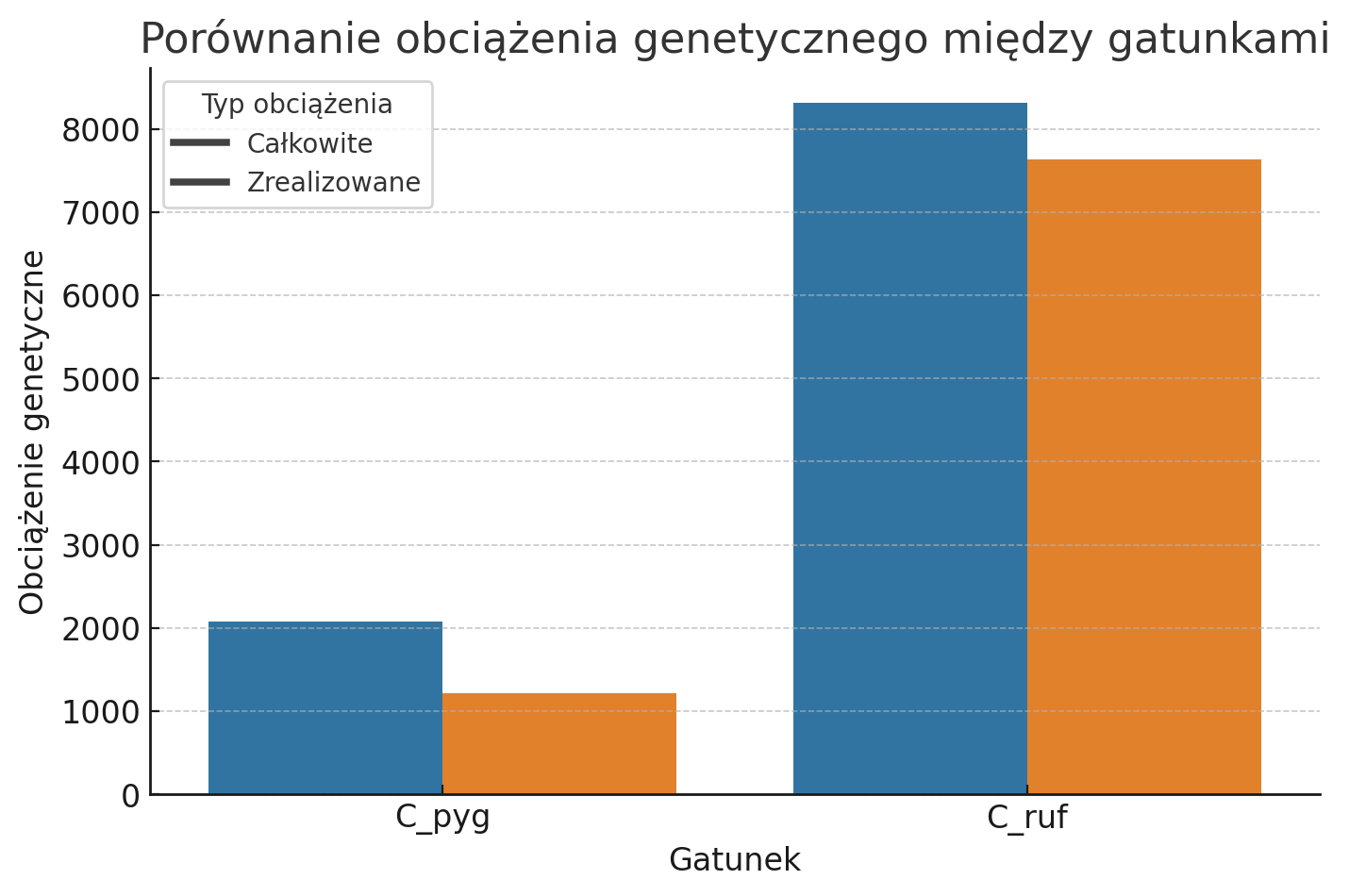
Całkowite obciążenie genetyczne: 2078.16

Zrealizowane obciążenie genetyczne: 1217.47

**C\_ruf**

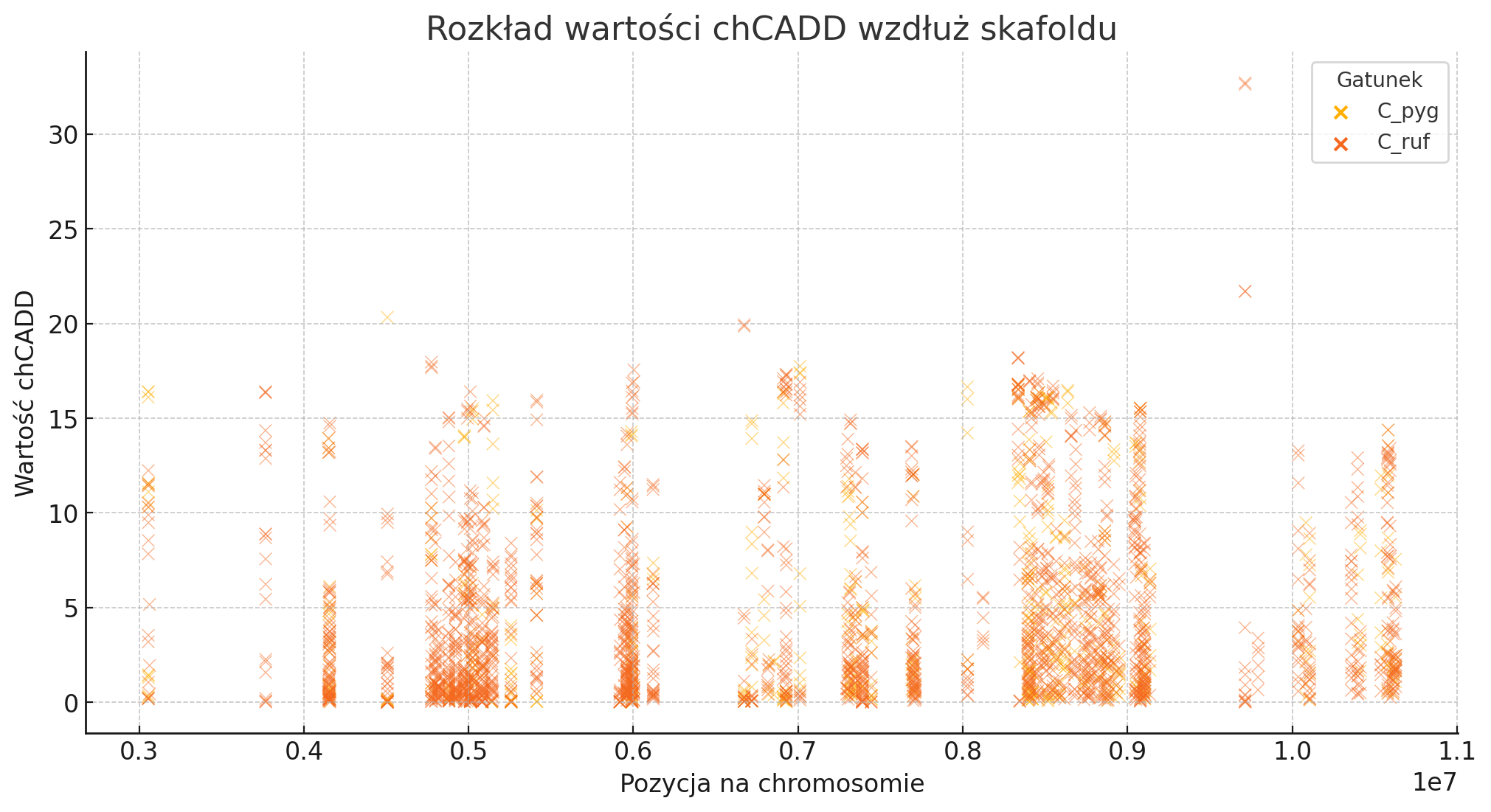
Całkowite obciążenie genetyczne: 8314.36

Zrealizowane obciążenie genetyczne: 7631.07



6. Zwizualizuj wartości wzdłuż analizowanego skafoldu i odpowiedz na pytanie, czy korelują one z pozycją?

Wydaje mi się, że nie ma wyraźnej korelacji z pozycją, wydają się być rozproszone losowo.



### Wnioski i dyskusja

Jeśli chodzi o liczbę wariantów, jest ona większa w genomie *C. ruficollis* w porównaniu do genomu *C. pymae*. Sugeruje to większe zróżnicowanie genetyczne tego gatunku, co by się pokrywało z faktem, że jest on stabilniejszy populacyjnie.

Wysoka ilość homozygot u *C. ruficollis* sugeruje stabilność i mniejszy inbred. Większa ilość heterozygot u *C. pymae* wydaje się być skutkiem wyższej selekcji lub może efektu bottleneck w tej populacji.

Średnia wartość CADD (Combined Annotation Dependent Depletion) jest inna w obu gatunkach - mniejsza u *C. ruficollis*, czyli mutacje u tego gatunku bardziej prawdopodobnie mają wpływ na funkcje i mogą być szkodliwe.

Natomiast wyższe obciążenie genetyczne C. ruficollis pokazuje, że mimo większej liczby mutacji, mogą one być mniej szkodliwe niż u C. pygmae, który ma to obciążenie genetyczne niższe.

Nie zaobserwowano korelacji wartości CADD z pozycją na scaffoldzie. Warianty są rozmieszczone losowo – mutacje nie są powiązane z określonymi regionami na genomie.

Po analizie wyników, C.pygmae wydaje się być bardziej narażony na inbred, jak i wpływ mutacji szkodliwych, niż stabilny populacyjnie C. ruficollis, co nakłania mnie do wniosków, że analiza została przeprowadzona poprawnie. Uważam, że wyniki tej analizy mogą być przykładem na to, że powinniśmy podjąć kroki w stronę ochrony gatunku C. pygmae, takie jak zwiększenie różnorodności genetycznej np. program hodowli ex situ wraz z doborem genetycznym, w celu uniknięcia krzyżowania się blisko spokrewnionych osobników (inbredu). Innym działaniem mogłaby być wymiana osobników między subpopulacjami, jeśli takie są lub identyfikacja i następnie ochrona osobników o korzystnych wariantach. Najmniej inwazyjnym, a według mnie kluczowym, krokiem powinna być ochrona naturalnych siedlisk, na których ten gatunek występuje, jak i ograniczenie działalności człowieka w okolicy - jeśli takie mają miejsce.