```
# 2

#Run RENUM:
echo renum.par | renumf90 | tee renum.log

# 3
#!!! Quality control is off due to indirect predictions in ex 5
#The SNP file for the new animals (new_animals) has not gone through QC so
#if we do QC in one file but not in the other, predf90 wont be able to calculate predictions

#Run ssGBLUP:
mkdir ssgblup; cd ssgblup

grep -v OPTION ../renf90.par > ssgblup.par
sed -i 's:renf90.dat:../renf90.dat:g' ssgblup.par
sed -i 's:renadd02.ped:../renadd02.ped:g' ssgblup.par
echo "OPTION use_yams" >> ssgblup.par
echo "OPTION SNP_file ../snp3.2k" >> ssgblup.par
echo "OPTION map_info ../mrkmap.txt" >> ssgblup.par
echo "OPTION no_quality_control" >> ssgblup.par
echo "OPTION saveGInverse" >> ssgblup.par
echo "OPTION saveA22Inverse" >> ssgblup.par

echo ssgblup.par | blupf90 | tee ssgblup.log
cp solutions ssgblup_solutions

# 4
#Run postGS:
cp ssgblup.par postgs1.par
sed -i 's:saveGInverse:readGInverse:g' postgs1.par
sed -i 's:saveA22Inverse:readA22Inverse:g' postgs1.par

echo postgs1.par | postGSf90 | tee post1.log
head snp_sol


# Ex. 5
echo ../new_animals | predf90 | tee pred.log


# 6
cd ../
mkdir linear ; cd linear
# Run blupf90 to save Gi, A22i and obtain solutions
```

```
cp ../ssgblup/ssgblup.par .
echo ssgblup.par | blupf90 | tee blup.log

# Run postGSf90
cp ../ssgblup/postgs1.par postgs2.par

# Creat a vector of weights D=I at first
awk 'BEGIN { for (i==1;i<45000;i++) print 1}' > W
echo "OPTION weightedG W" >> postgs2.par
echo "OPTION Manhattan_plot" >> postgs2.par
echo "OPTION windows_variance 20" >> postgs2.par

echo postgs2.par | postGSf90 | tee postgs.log


cd ../
mkdir non_linear ; cd non_linear
# Run blupf90 to save Gi, A22i and obtain solutions
cp ../ssgblup/ssgblup.par .
echo ssgblup.par | blupf90 | tee blup.log

# Run postGSf90
# Creat a vector of weights D=I at first
awk 'BEGIN { for (i==1;i<45000;i++) print 1}' > W
cp ../ssgblup/postgs1.par postgs3.par
echo "OPTION which_weight nonlinearA" >> postgs3.par
echo "OPTION weightedG W" >> postgs3.par
echo "OPTION Manhattan_plot" >> postgs3.par
echo "OPTION windows_variance 20" >> postgs3.par

echo postgs3.par | postGSf90 | tee postgs.log

cd ../
# 7
# Use the option to compute p-values
mkdir p_val ; cd p_val
cp ../ssgblup/ssgblup.par blup.par
cp ../ssgblup/postgs1.par postgs.par

echo "OPTION snp_p_value" >> blup.par
echo "OPTION snp_p_value" >> postgs.par

time echo blup.par | blupf90 | tee blup.log
#real    0m9.970s
```

```
#user   1m41.453s
#sys    0m4.148s

time echo postgs.par | postGSf90 | tee postgs2.log
#real   0m25.547s
#user   5m41.579s
#sys    0m6.382s


# 8
# Run one more iteration, updating the weights
cd ../linear

cp solutions solutions_1
cp snp_sol snp_sol_1
cp chrsnp chrsnp_1
cp W W_1
awk 'NR>1 {print $7}' snp_sol > W

echo ssgblup.par | blupf90 | tee ssgblup2.log
echo postgs2.par | postGSf90 | tee postgs2.log


cd ../non_linear
cp solutions solutions_1
cp snp_sol snp_sol_1
cp chrsnp chrsnp_1
cp W W_1
awk 'NR>1 {print $7}' snp_sol > W
# echo "OPTION SNP_variance_limit 8.3319" >> postgs3.par

echo ssgblup.par | blupf90 | tee gblup3.log
echo postgs3.par | postGSf90 | tee postgs3.log


#Example of iteration script for WssGBLUP (2 iterations: for i in {1..2})
mkdir plots
awk 'BEGIN { for (i==1;i<45000;i++) print 1}' > W
 for i in {1..2}
   do
     echo ssgblup.par | blupf90 | tee ssgblup.log_$i
     cp solutions solutions_$i
     echo postgs3.par | postGSf90 | tee postgs.log_$i
     cp snp_sol snp_sol_$i
```

```
    cp chrsnp chrsnp_$i
    cp W W_$i
            cp Sft1e2.R plots/Sft1e2_$i.R
            cp Vft1e2.R plots/Vft1e2_$i.R
    awk 'NR>1 {print $7}' snp_sol > W
 done

cd ../
```