I'm trying to understand how the GCTA trait simulation works and am walking through the calculation for a simulated trait. I'm using the example dataset from a simulation run with 96 individuals.

Here is the GWAS simulation notes from GCTA:

"The phenotypes are simulated based on a set of real genotype data and a simple additive genetic model yj = sum(wijui) + ej, where wij = (xij - 2pi) / sqrt[2pi(1 - pi)] with xij being the number of reference alleles for the i-th causal variant of the j-th individual and pi being the frequency of the i-th causal variant, ui is the allelic effect of the i-th causal variant and ej is the residual effect generated from a normal distribution with mean of 0 and variance of var(sum(wij*ui))(1 / h2 - 1)."

I have an example dataset from a simulation run with 96 individuals. A simulated trait (`data/test_data/1_1_gamma_ce.96.a` was generated by the simulation pipeline with a heritability of 0.9.

```
# set path to the genotype matrix created for the test mapping panel
gt_m_path <- "data/test_data/ce.96.allout15_irrepressible.grosbeak_0.05_Genotype_Matrix.tsv"

# path to the file containing the causal variants
causal_var_path <- "data/test_data/1_1_gamma_ce.96.allout15_irrepressible.grosbeak_0.05/PYTHON_SIMULATE_

# path to the .par file for the simulation
par_path <- "data/test_data/1_1_gamma_ce.96.allout15_irrepressible.grosbeak_0.05/0.9/GCTA_SIMULATE_PHEN

# path to the simulated phenotype file
phen_path <- "data/test_data/1_1_gamma_ce.96.allout15_irrepressible.grosbeak_0.05/0.9/GCTA_SIMULATE_PHE
```

```
# load the causal variants
causal_var <- data.table::fread(
  causal_var_path,
  col.names = c("snpid", "effect")
) %>%
  tidyr::separate(snpid, into = c("chrom", "pos"), sep = ":", convert = TRUE)
head(causal_var)
```

```
##   chrom     pos   effect
## 1     1 4644600 2.489919
```

The causal variant selected is 1:4644600 and has a frequency of 0.0833333 and the effect size assigned to it from the gamma distribution is 2.48992. This is included in the par file generated by the GCTA trait simulation step of the pipeline.

```
# load the par file
par <- data.table::fread(par_path, header = TRUE, sep = "\t", data.table = FALSE)
head(par)
```

```
##         QTL RefAllele Frequency  Effect
## 1 1:4644600         C 0.0833333 2.48992
```

Load the genotype matrix

## Get the genotype matrix for the causal variant

```
cv_genos <- gt_m %>%
  dplyr::filter(
    CHROM == causal_var$chrom[1],
    POS == causal_var$pos[1]
  )
head(cv_genos)
```

```
##   CHROM     POS REF ALT BRC20263 CB4851 CB4853 CB4932 DL226 ECA1255 ECA1316
## 1     1 4644600   A   C       -1     -1     -1     -1    -1       -1       -1
##   ECA1901 ECA2522 ECA2527 ECA2533 ECA2549 ECA2559 ECA2583 ECA2603 ECA2676
## 1      -1      -1      -1      -1      -1      -1      -1      -1      -1
##   ECA592 ECA738 ED3005 ED3011 ED3046 EG4724 EG4946 GXW1 JT11398 JU1409 JU1440
## 1     -1     -1     -1     -1     -1     -1     -1   -1      -1     -1     -1
##   JU1652 JU1793 JU1934 JU2001 JU2250 JU2257 JU2576 JU2593 JU2619 JU2829 JU2838
## 1     -1     -1     -1     -1     -1     -1     -1     -1      1     -1     -1
##   JU2866 JU3128 JU3132 JU3137 JU3144 JU3166 JU3228 JU323 JU346 JU397 JU4047
## 1     -1     -1     -1     -1     -1      1     -1    -1    -1    -1     -1
##   JU751 JU778 KR314 MY1 MY18 MY2147 MY2212 MY2693 MY2713 MY795 MY920 NIC1107
## 1    -1    -1    -1  -1   -1     -1     -1     -1     -1    -1    -1      -1
##   NIC1700 NIC1773 NIC1782 NIC1783 NIC1786 NIC1794 NIC1809 NIC1810 NIC1832
## 1      -1      -1      -1      -1      -1      -1      -1      -1      -1
##   NIC199 NIC2 NIC255 NIC272 NIC274 NIC276 NIC522 NIC523 PS2025 QG2818 QG2823
## 1     -1   -1     -1      1     -1      1     -1     -1     -1     -1      1
##   QG4006 QG4012 QG4080 QG4151 QG4226 QG536 QG556 RC301 TWN2530 WN2001 WN2002
## 1     -1     -1     -1      1      1    -1    -1     1      -1     -1     -1
##   WN2033 WN2066 XZ1672 XZ1734
## 1     -1     -1     -1     -1
```

Drop the `CHOM`, `POS`, `REF` and `ALT` columns of the genotype matrix and create a new data frame by pivoting the genotype matrix long

```
# drop the CHROM, POS, REF and ALT columns
cv_genos <- cv_genos %>%
  dplyr::select(-CHROM, -POS, -REF, -ALT) %>%
  tidyr::pivot_longer(
    cols = everything(),
    names_to = "strain",
    values_to = "GT"
  )
head(cv_genos)
```

```
## # A tibble: 6 x 2
##   strain      GT
##   <chr>    <int>
## 1 BRC20263    -1
## 2 CB4851      -1
## 3 CB4853      -1
## 4 CB4932      -1
## 5 DL226       -1
## 6 ECA1255     -1
```

Based on the GT column we are going to create a new colum that shows the number of reference alleles for each strain. If a strain is -1 (homozygous REF) then the number of reference alleles is 0. If a strain is 1 (homozygous ALT) then the number of reference alleles is 2.

```
cv_genos <- cv_genos %>%
  dplyr::mutate(
    REF_ALLELES = dplyr::case_when(
      GT == -1 ~ 0,
      GT == 1 ~ 2,
      TRUE ~ NA_real_
    )
  )
head(cv_genos)
```

```
## # A tibble: 6 x 3
##   strain      GT REF_ALLELES
##   <chr>    <int>       <dbl>
## 1 BRC20263    -1           0
## 2 CB4851      -1           0
## 3 CB4853      -1           0
## 4 CB4932      -1           0
## 5 DL226       -1           0
## 6 ECA1255     -1           0
```

## Calculate the wij for all strains based on the number of reference alleles and the AF

Pull / set the variables for the calculation

```
# get the AF for the SNP of interest
p <- par %>%
  dplyr::pull(Frequency)
# get the effect size for the SNP of interest
u <- effect_size <- causal_var %>%
  dplyr::pull(effect)

h2 <- 0.9

print(glue::glue("AF: {p}"))
```

```
## AF: 0.0833333
```

```
print(glue::glue("Effect size: {u}"))
```

```
## Effect size: 2.48991852882817
```

```
print(glue::glue("Heritability: {h2}"))
```

```
## Heritability: 0.9
```

Now we can calculate the wij for all strains based on the number of reference alleles and the AF using the formula wij = (REF_ALLELES - 2 * p) / sqrt(2 * p * (1 - p))

```
# calculate the wij for all strains based on the number of reference alleles and the AF
cv_genos_wij <- cv_genos %>%
  dplyr::mutate(
    wij = (REF_ALLELES - 2 * p) / sqrt(2 * p * (1 - p))
  )
head(cv_genos_wij)
```

```
## # A tibble: 6 x 4
##   strain      GT REF_ALLELES    wij
##   <chr>    <int>       <dbl>  <dbl>
## 1 BRC20263    -1           0 -0.426
## 2 CB4851      -1           0 -0.426
## 3 CB4853      -1           0 -0.426
## 4 CB4932      -1           0 -0.426
## 5 DL226       -1           0 -0.426
## 6 ECA1255     -1           0 -0.426
```

Now we can calculate the genetic component for all strains based on the wij and the effect size using the formula gj = wij * u

```
# calculate the genetic component for all strains based on the wij and the effect size
cv_genos_gj <- cv_genos_wij %>%
  dplyr::mutate(
    gj = wij * u
  )
head(cv_genos_gj)
```

```
## # A tibble: 6 x 5
##   strain      GT REF_ALLELES    wij    gj
##   <chr>    <int>       <dbl>  <dbl> <dbl>
## 1 BRC20263    -1           0 -0.426 -1.06
## 2 CB4851      -1           0 -0.426 -1.06
## 3 CB4853      -1           0 -0.426 -1.06
## 4 CB4932      -1           0 -0.426 -1.06
## 5 DL226       -1           0 -0.426 -1.06
## 6 ECA1255     -1           0 -0.426 -1.06
```

Look at the genetic component for strains with the causal variant

```
# look at the genetic component for strains with the causal variant
cv_genos_gj %>%
  dplyr::filter(REF_ALLELES == 2) %>%
  head()
```

```
## # A tibble: 6 x 5
##   strain    GT REF_ALLELES   wij    gj
##   <chr>  <int>       <dbl> <dbl> <dbl>
## 1 JU2619     1           2  4.69  11.7
## 2 JU3166     1           2  4.69  11.7
```

```
## 3 NIC272     1           2  4.69  11.7
## 4 NIC276     1           2  4.69  11.7
## 5 QG2823     1           2  4.69  11.7
## 6 QG4151     1           2  4.69  11.7
```

## Calculate the residual effect

To get the residual effect we need to calculate the variance of the genetic component using the formula var(gj) = var(wij * u)

```
# calculate the variance of the genetic component
var_g <- var(cv_genos_gj$gj)
print(var_g)
```

```
## [1] 12.52991
```

Now we can calculate the residual effect using the formula var(e) = var(g) * (1 / h2 - 1)

```
# calculate the residual effect
var_e <- var_g * (1 / h2 - 1)
print(head(var_e))
```

```
## [1] 1.392213
```

Now we can sample from a normal distribution to get the residual effect using the formula ej = rnorm(n = 96, mean = 0, sd = sqrt(var_e))

```
# sample from a normal distribution to get the residual effect
set.seed(123)
ej <- rnorm(n = 96, mean = 0, sd = sqrt(var_e))
print(ej)
```

```
##  [1] -0.661316745 -0.271591155  1.839152005  0.083194302  0.152549258
##  [6]  2.023640460  0.543844513 -1.492671775 -0.810431810 -0.525845726
## [11]  1.444319294  0.424551736  0.472878479  0.130596813 -0.655848389
## [16]  2.108415570  0.587424021 -2.320452039  0.827544256 -0.557856308
## [21] -1.259947157 -0.257193086 -1.210603755 -0.860033756 -0.737496690
## [26] -1.990164135  0.988522168  0.180968097 -1.342911185  1.479402018
## [31]  0.503193908 -0.348160913  1.056177182  1.036127766  0.969400419
## [36]  0.812540802  0.653578834 -0.073050901 -0.361011641 -0.448925561
## [41] -0.819699056 -0.245325874 -1.493067186  2.559195762  1.425299211
## [46] -1.325178921 -0.475372105 -0.550616252  0.920296888 -0.098368877
## [51]  0.298895726 -0.033682904 -0.050583734  1.614841989 -0.266391830
## [56]  1.789314862 -1.827405294  0.689797790  0.146138170  0.254793899
## [61]  0.447944435 -0.592701776 -0.393158247 -1.201838048 -1.264628517
## [66]  0.358139688  0.528851938  0.062540777  1.088202358  2.418937094
## [71] -0.579377774 -2.724635860  1.186691575 -0.836800569 -0.811795520
## [76]  1.210092757 -0.336009529 -1.440349941  0.213923705 -0.163880776
## [81]  0.006801282  0.454600271 -0.437349397  0.760313145 -0.260156630
## [86]  0.391476364  1.294182915  0.513479593 -0.384573383  1.355501742
## [91]  1.172255638  0.647064854  0.281684485 -0.740879296  1.605461815
## [96] -0.708258635
```

Lets also look at the range of the residual effect

```
# look at the range of the residual effect
residual_effect_range  <- range(ej)
print(residual_effect_range)
```

```
## [1] -2.724636  2.559196
```

## Calculate the trait value for all strains

Now we can calculate the phenotype for all strains using the formula phen = gj + ej

```
# calculate the phenotype for all strains
cv_genos_phen <- cv_genos_gj %>%
  dplyr::mutate(
    phen = gj + ej
  )
head(cv_genos_phen)
```

```
## # A tibble: 6 x 6
##   strain      GT REF_ALLELES    wij    gj   phen
##   <chr>    <int>       <dbl>  <dbl> <dbl>  <dbl>
## 1 BRC20263    -1           0 -0.426 -1.06 -1.72
## 2 CB4851      -1           0 -0.426 -1.06 -1.33
## 3 CB4853      -1           0 -0.426 -1.06  0.777
## 4 CB4932      -1           0 -0.426 -1.06 -0.979
## 5 DL226       -1           0 -0.426 -1.06 -0.909
## 6 ECA1255     -1           0 -0.426 -1.06  0.962
```

Peak at the phenotype for strains with the causal variant

```
# peak at the phenotype for strains with the causal variant
cv_genos_phen %>%
  dplyr::filter(REF_ALLELES == 2) %>%
  head()
```

```
## # A tibble: 6 x 6
##   strain    GT REF_ALLELES   wij    gj  phen
##   <chr>  <int>       <dbl> <dbl> <dbl> <dbl>
## 1 JU2619     1           2  4.69  11.7  12.5
## 2 JU3166     1           2  4.69  11.7  14.2
## 3 NIC272     1           2  4.69  11.7  10.8
## 4 NIC276     1           2  4.69  11.7  12.9
## 5 QG2823     1           2  4.69  11.7  11.7
## 6 QG4151     1           2  4.69  11.7  11.4
```

## Compare the calculated phenotype to the simulated phenotype

And compare it to the simulated phenotype file

```r
# compare it to the simulated phenotype file
phen <- data.table::fread(
  phen_path,
  col.names = c("id", "strain", "simulated_pheno")
) %>%
  dplyr::select(-id)
head(phen)
```

```
##     strain simulated_pheno
##     <char>           <num>
## 1:  MY2713       -1.885160
## 2:   JU323        0.408951
## 3:  XZ1734       -1.909350
## 4:  QG4080       -1.851420
## 5:   DL226       -0.610534
## 6: ECA2533       -2.054670
```

Attach the phenotype to the genotype matrix

```r
# attach the phenotype to the genotype matrix
cv_genos_phen_check <- cv_genos_phen %>%
  dplyr::left_join(phen, by = "strain")
head(cv_genos_phen_check)
```

```
## # A tibble: 6 x 7
##   strain      GT REF_ALLELES    wij    gj   phen simulated_pheno
##   <chr>    <int>       <dbl>  <dbl> <dbl>  <dbl>           <dbl>
## 1 BRC20263    -1           0 -0.426 -1.06 -1.72           -0.581
## 2 CB4851      -1           0 -0.426 -1.06 -1.33            0.528
## 3 CB4853      -1           0 -0.426 -1.06  0.777          -0.896
## 4 CB4932      -1           0 -0.426 -1.06 -0.979          -1.15
## 5 DL226       -1           0 -0.426 -1.06 -0.909          -0.611
## 6 ECA1255     -1           0 -0.426 -1.06  0.962          -1.13
```

Check the strains with only alt alleles

```r
# check the strains with no reference alleles
cv_genos_phen_check %>%
  dplyr::filter(REF_ALLELES == 2)
```

```
## # A tibble: 8 x 7
##   strain    GT REF_ALLELES   wij    gj  phen simulated_pheno
##   <chr>  <int>       <dbl> <dbl> <dbl> <dbl>           <dbl>
## 1 JU2619     1           2  4.69  11.7  12.5            12.0
## 2 JU3166     1           2  4.69  11.7  14.2            13.0
## 3 NIC272     1           2  4.69  11.7  10.8            13.3
## 4 NIC276     1           2  4.69  11.7  12.9            11.3
## 5 QG2823     1           2  4.69  11.7  11.7            11.0
## 6 QG4151     1           2  4.69  11.7  11.4            12.5
## 7 QG4226     1           2  4.69  11.7  12.1            12.6
## 8 RC301      1           2  4.69  11.7  11.3            10.6
```

Lets make one final check that the difference between the simulated phenotype and the calculated phenotype is due to the residual effect The abs difference should be within the residual effect range since the residual effects are randomly assigned to each strain

```
cv_genos_phen_check <- cv_genos_phen_check %>%
  dplyr::mutate(
    abs_diff = abs(simulated_pheno - phen)
  ) %>%
  arrange(desc(abs_diff))

head(cv_genos_phen_check)
```

```
## # A tibble: 6 x 8
##   strain     GT REF_ALLELES    wij    gj    phen simulated_pheno abs_diff
##   <chr>   <int>       <dbl>  <dbl> <dbl>   <dbl>           <dbl>    <dbl>
## 1 JU1793     -1           0 -0.426 -1.06 -2.40             1.34     3.74
## 2 NIC1810    -1           0 -0.426 -1.06  0.0265          -3.66     3.69
## 3 WN2002     -1           0 -0.426 -1.06 -0.415           -3.61     3.20
## 4 JU346      -1           0 -0.426 -1.06 -1.54             1.55     3.09
## 5 JU3132     -1           0 -0.426 -1.06 -1.88             1.12     3.01
## 6 JU1934     -1           0 -0.426 -1.06  0.418           -2.55     2.97
```

Calculate the mean of the absolute difference between the simulated phenotype and the calculated phenotype

```
# calcualte the mean of the absolute difference between the simulated phenotype and the calculated phen
mean_abs_diff <- mean(cv_genos_phen_check$abs_diff)
print(glue::glue("Mean absolute difference: {mean_abs_diff}"))
```

```
## Mean absolute difference: 1.33665864689316
```

```
range_abs_diff <- range(cv_genos_phen_check$abs_diff)
print(glue::glue("Range of absolute difference: {range_abs_diff}"))
```

```
## Range of absolute difference: 0.0305444907603538
## Range of absolute difference: 3.74283578159334
```