	1	2	3	Σ
Benedikt Hopf Alireza Ketabdari				

Exercise Sheet Nr. 1

(Deadline November 09, 2021)

Problem 1

a) *GWAS* stands for *Genome-Wide Association Studies.* The purpose of *GWAS* is to find associations between the genotype and the phenotype of a person or in other words to find out how certain features of a person (say their risk for a certain disease) are associated with thier genome. One possible question could look like that:
Is a person with a genome that looks like that more or less likely to develop diabetes than others/than people are on average.

b) *SNPs* are places in the DNA where exactly one base-pair is different between (*Single Nucleotide Polymorphism*) different genomes (different genomes are the same at every point except those). Since there are four different bases used in the genome (*A, C, G, T*) there are four possibilities for a *SNP.* In practice there are usually only two versions that actually exist.

c) Allele describes a part of a chromosome, that varies between different people. Usually there is a mor common variant (major allele) and a less common variant (minor allele).

d) Sometimes two SNPs are correlated. That is, if one SNP has a certain value, the other one is very likely to also have a certain fixed value. Knowing that one does not have to know the value of every SNP as some SNPs can be inferred by the value of a linked SNP.

e) Regression describes a learning model, which has some continuous output (e.g. $\mathbb{R}$). Classification on the other hand has discrete output (e.g. $\{cat, dog, cow\}$).

The price of a house can be a regression problem, where the output would be the exact price (so some value in $\mathbb{R}^+$). One could also phrase it as a regression problem, where one would not predict the price exactly but just some categories (something like $\{< 100000, >= 100000\}$).

f) The significance level α is the maximum probability of the teststatistic $T(x)$ under the nullhypothesis H_0, where H_0 would be discarded. That is H_0 would be discarded if

$$P_{H_0}(T(X) \text{ more extreme than P(x)}) < \alpha$$

where $T(x)$ is the actual measured teststatistic and $T(X)$ is the random variable. The $p - value$ is the largest α such that H_0 is rejected:

$$p - value = \arg\inf_{\alpha} P_{H_0}(T(X) \text{ more extreme than P(x)}) < \alpha$$

The significance level α is chosen (often $\alpha = 0.05$), whereas the $p-value$ is calculated from the measurement. We accept alternative hypothesis (reject the null hypothesis) if $p - value < \alpha$.

g) Using a significance level α means, that even if the null hypothesis was correct, there is a chance of α (5%) that we sill do a measurement that leads us to reject the null hypothesis. So in expectation we should see a false rejection after $\frac{1}{\alpha}$ experiments. So if we test multiple hypotheses we would get a lot of false rejections. That can be dealt with in multiple ways:

1. The Bonferroni method: The significance level is devided by the number of tests. So when doing n tests a significance level of $\frac{\alpha}{n}$ is used as opposed to just α. This significantly reduces the chance of false rejections.
2. For independent hypotheses one can use the Benjamini-Hochberg Method. There on sorts the p-values (ascending) and selects rejects the null hypothesis as long as $p_j < \frac{j}{M}\alpha$ (where there are M $p-values$). Thereby one calls only those $p-values$ significant which are "surprisingly small".

h) Confounding factors are (unexpected) reasons other than the alternative hypotesis to find some correlation. For example if one has an experiment where in the case group there are mostly Europeans and in the control group mostly Africans, there is quite a good chance, that one will just find genetic differences between Europeans and Africans (which is not what one wants to find). EIGENSTRAT can account for such geographic factors, by performing PCA on the matrix of SNPs and people. Then the values are being corrected using tha first eigenvectors, which correspond quite well to geographic location. EIGENSTART can however not account for family structure and cryptic relatedness.

i) The Cochan-Armitage test considers three types of people:

1. Homozygous major allele: Both chromosomes have the major allele.
2. Heterozygous: One chromosome has the major and one the minor allele.
3. Homozygous minor allele: both chromosomes have the minor allele.

which are numbered 0, 1 and 2. Then the test tries to fit a linear model from those numbers to the associated disease risk. That works well is every minor allele increases the risk (or if every major one does). It can however not detect the case where only the heterozygous configuration is dangerous

j) Standard linear models assume the generative model $y = X\beta + \epsilon$, where y is the phenotypes, X the gene-matrix, β some weigths and ϵ some noise. This can cause problems, if there are similarities (e.g. family) between the differnt testet persons. Linear mixed models assume $y = X\beta + u + \epsilon$ where u encodes some additional effects (like family structure). This is achived using a so called kinship matrix which includes the pairwise genetic similarities between the persons. This is helpful because this way the model can correct for the fact, that the genes of the tested persons are not necessarily uncorrelated.

k) The FAST-LLM algorithm reduces computation time by using only a subset of SNPs, equally spaced over the entire genome. This is possible, because SNPs that are close to each other are often linked. Usually for the analysis one would need to do a spectral decomposition per SNP which has cubic complexity. Fast-LMM manages to get away with one spectral decomposition. There however is a trick if there are fewer SNPs than people in which case there are significant speedups due to (necessarily) linear dependence in the matrix. This allows for those faster computation times[1].

[1] http://www.cs.toronto.edu/~jenn/papers/FaST-LMM-2011.pdf pages 2 and 3

Problem 2

a) i) There are 1000 people, so 2000 alleles. Therefore we get:

$$\begin{aligned} n_A &= 2 \cdot 299 + 490 = 1088 \implies p_1 = \frac{1088}{2000} = 0.544 \\ n_G &= 2 \cdot 211 + 490 = 912 \implies p_2 = \frac{1088}{2000} = 0.456 \end{aligned}$$

ii) Using the formulas provided above:

$$\begin{aligned} p_{AA} = p_1^2 = 0.544^2 = 0.295936 &\implies n_{AA} = 1000 \cdot p_{AA} = 295.936 \approx 296 \\ p_{AG} = 2p_1p_2 = 2 \cdot 0.544 \cdot 0.456 = 0.496128 &\implies n_{AA} = 1000 \cdot p_{AA} = 496.128 \approx 496 \\ p_{AA} = p_2^2 = 0.456^2 = 0.207936 &\implies n_{AA} = 1000 \cdot p_{AA} = 207.936 \approx 208 \end{aligned}$$

iii) We will call the expected number of genotype i n_i and the actual measured number of that genotype $\tilde{n}_i$. The chi-sqared teststatistic is then[2]:

$$\begin{aligned} T(n) &= \sum_{i \in \{AA,AG,GG\}} \frac{(\tilde{n}_i - n_i)^2}{n_i} \\ &= \frac{(299 - 295.936)^2}{295.936} + \frac{(490 - 496.128)^2}{496.128} + \frac{(211 - 207.936)^2}{207.936} \\ &= 10.579737978201427 \approx 10.58 \end{aligned}$$

Since we have one degree of freedom (p_1, because p_2 is then defined by $p_1 + p_2 = 1$), $T(n)$ is approximately χ_1^2 verteilt. We therefore get:

$$\begin{aligned} p - value &\approx 1 - F_{\chi_1^2}(n) = 1 - 0.3039022735605503 = 0.6960977264394497 \\ &\approx 0.696 > 0.05 = \alpha \end{aligned}$$

Therefore we can not reject the nullhypothesis.

b) i) Usually during meiosis one gets one complete chromosome from each of the parents. However due to crossover, there is a chance to get a mixture of both chromosomes of each of the parents. If the way this happens is completely random (the parents chromosomes could combine in any way) there is linkage equilibrium. If however that distribution is uniformly random (that is some combinations are more likely than others and thus on can infer the prescence of one part of the chromosome from another part) this is called linkage disequilibrium.

ii) In order for this to be in linkage equilibrium we need:

$$p_{1,1} = a_1 \cdot b_1$$

iii) *Proof.* To show this we will prove that D = 0 if and only if there is linkage equilibrium.
If there is linkage equilibrium, that is the formula from ii) holds:

$$\begin{aligned} D &= p_{1,1}p_{2,2} - p_{1,2}p_{2,1} \\ &= a_1b_1a_2b_2 - a_1b_2a_2b_1 \\ &= a_1b_1a_2b_2 - a_1b_1a_2b_2 = 0 \end{aligned}$$

[2] This is taken from my Mathe 4 script

On the other hand from D = 0 follows:

$$\begin{aligned} 0 = D &= p_{1,1}p_{2,2} - p_{1,2}p_{2,1} \\ p_{1,1}p_{2,2} &= p_{1,2}p_{2,1} \end{aligned}$$

Let n be the size of the total population the we furthermore have:

$$\begin{aligned} n_{a_1} &= na_1 = n(p_{1,1} + p_{1,2}) \\ n_{a_2} &= na_2 = n(p_{2,1} + p_{2,2}) \\ n_{b_1} &= nb_1 = n(p_{1,1} + p_{2,1}) \\ n_{b_2} &= nb_2 = n(p_{1,2} + p_{2,2}) \end{aligned}$$

This imples:

$$\begin{aligned} a_1 &= p_{1,1} + p_{1,2} \\ a_1 - p_{1,2} &= p_{1,1} \end{aligned}$$

and

$$\begin{aligned} a_2 &= p_{2,1} + p_{2,2} \\ a_2 - p_{2,1} &= p_{2,2} \end{aligned}$$

Substituting into the above equation we get:

$$\begin{aligned} p_{1,1}p_{2,2} &= p_{1,2}p_{2,1} \\ (a_1 - p_{1,2})(a_2 - p_{2,1}) = a_1a_2 - a_1p_{2,1} - a_2p_{1,2} + p_{1,2}p_{2,1} &= p_{1,2}p_{2,1} \\ a_1a_2 - a_1p_{2,1} - a_2p_{1,2} &= 0 \end{aligned}$$

With $p_{2,1} = b_1 - p_{1,1}$ and $p_{1,2} = a_1 - p_{1,1}$

$$\begin{aligned} a_1a_2 - a_1b_1 + a_1p_{1,1} - a_2a_1 + a_2p_{1,1} &= 0 \\ -a_1b_1 + a_1p_{1,1} + a_2p_{1,1} &= 0 \\ a_1p_{1,1} + a_2p_{1,1} &= a_1b_1 \\ \underbrace{(a_1 + a_2)}_{=1} p_{1,1} &= a_1b_1 \\ p_{1,1} &= a_1b_1 \end{aligned}$$

Substituting in $a_1 = p_{1,1} + p_{1,2}$ we get

$$\begin{aligned} a_1 &= p_{1,1} + p_{1,2} \\ a_1 &= a_1b_1 + p_{1,2} \\ a_1 - a_1b_1 = a_1(1 - b_1) = a_1b_2 &= p_{1,2} \end{aligned}$$

Substituting in $b_1 = p_{1,1} + p_{2,1}$ we get

$$\begin{aligned} b_1 &= p_{1,1} + p_{2,1} \\ b_1 &= a_1b_1 + p_{2,1} \\ b_1 - a_1b_1 = b_1(1 - a_1) = a_2b_1 &= p_{2,1} \end{aligned}$$

and finally by substituting in $b_2 = p_{1,2} + p_{2,2}$ we get

$$\begin{aligned} b_2 &= p_{1,2} + p_{2,2} \\ b_2 &= a_1 b_2 + p_{2,2} \\ b_2 - a_1 b_2 = b_2(1 - a_1) = a_2 b_2 &= p_{2,1} \end{aligned}$$

which proofs equilibrium.

□

Task 3

See following pages (jupyter notebook).

Task3

November 9, 2021

1 Notebook for task 3

We will start with some basic imports

```
[1]: import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     from scipy.stats import chi2
```

1.1 Part (a)

in order for the `bash` commands to execute, the paths for `plink` and `BOLT-LMM` have to be set accordingly (see the respective commands).

1.1.1 Use plink to run a Cochan-Armitage test

```
[2]: !../../plink/plink --bfile ../data/plink --model trend-only --out 3a --adjust
```

```
PLINK v1.90b6.24 64-bit (6 Jun 2021)           www.cog-genomics.org/plink/1.9/
(C) 2005-2021 Shaun Purcell, Christopher Chang   GNU General Public License v3
Logging to 3a.log.
Options in effect:
  --adjust
  --bfile ../data/plink
  --model trend-only
  --out 3a

7905 MB RAM detected; reserving 3952 MB for main workspace.
1440616 variants loaded from .bim file.
619 people (305 males, 314 females) loaded from .fam.
619 phenotype values loaded from .fam.
Using 1 thread (no multithreaded calculations invoked).
Before main variant filters, 523 founders and 96 nonfounders present.
Calculating allele frequencies… 1011121314151617181920212223242526272829303132
3334353637383940414243444546474849505152535455565758596061626364656667686970717 2
7374757677787980818283848586878889909192939495969798 9 done.
Warning: 1879 het. haploid genotypes present (see 3a.hh ); many commands treat
these as missing.
```

```
Total genotyping rate is 0.99772.
1440616 variants and 619 people pass filters and QC.
Among remaining phenotypes, 119 are cases and 500 are controls.
Excluding 112 MT/haploid variants from --model analysis.
Writing --model report to 3a.model … 10111213141516171819202122232425262728293
03132333435363738394041424344454647484950515253545556575859606162636465666768697
07172737475767778798081828384858687888990919293949596979899done.
--adjust: Genomic inflation est. lambda (based on median chisq) = 3.99938.
10111213141516171819202122232425262728293031323334353637383940414243444546474849
50515253545556575859606162636465666768697071727374757677787980818283848586878889
90919293949596979899--adjust values (1440504 variants) written to
3a.model.trend.adjusted .
```

1.1.2 Translate the space-seperated output to tab-seperated output which pandas can parse

[3]:
```
!cat 3a.model | tr -s " " "\t" > 3a.tsv
```

1.1.3 Load into pandas and show the table

[4]:
```
table = pd.read_csv("3a.tsv", sep="\t")
```

[5]:
```
table.head()
```

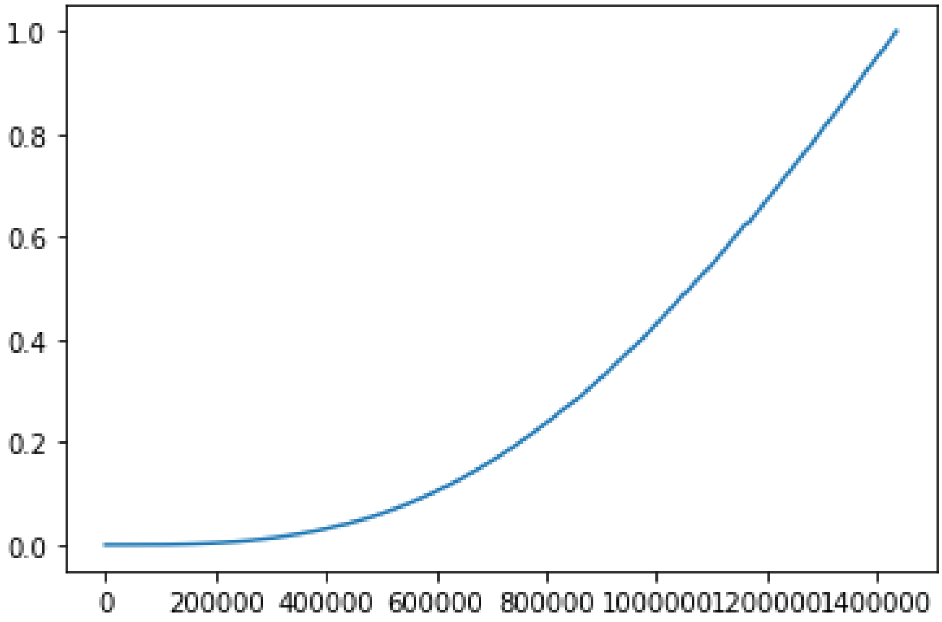
[5]:
```
   Unnamed: 0  CHR         SNP A1 A2   TEST    AFF    UNAFF    CHISQ   DF  \
0         NaN    1  rs10458597  T  C  TREND  4/230   13/973   0.1565  1.0
1         NaN    1   rs2185539  T  C  TREND  7/231   0/1000  29.7500  1.0
2         NaN    1  rs11240767  T  C  TREND 10/228   15/985   6.2060  1.0
3         NaN    1  rs12564807  G  A  TREND  0/238   0/1000      NaN  NaN
4         NaN    1   rs3131972  A  G  TREND 82/156  259/741   6.6180  1.0

              P
0  6.924000e-01
1  4.920000e-08
2  1.273000e-02
3           NaN
4  1.010000e-02
```

Here we can easily drop the *nan*-values

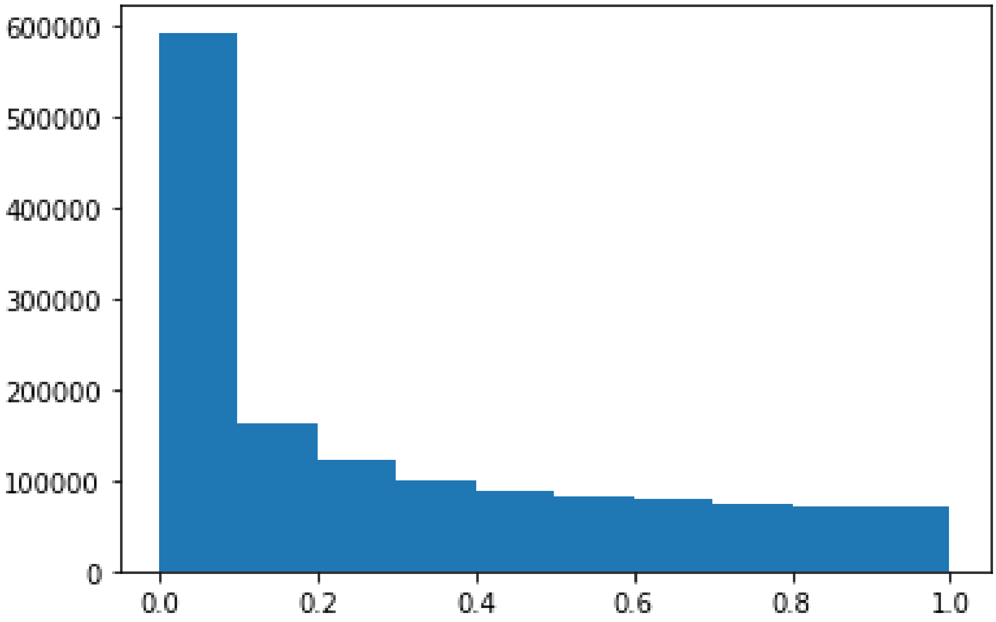
[6]:
```
ps = table["P"].dropna().to_numpy()
```

Also we do some plots to get an idea of the data

[7]:
```
plt.plot(sorted(ps))
plt.show()
```



[8]:
```
plt.hist(ps)
plt.show()
```

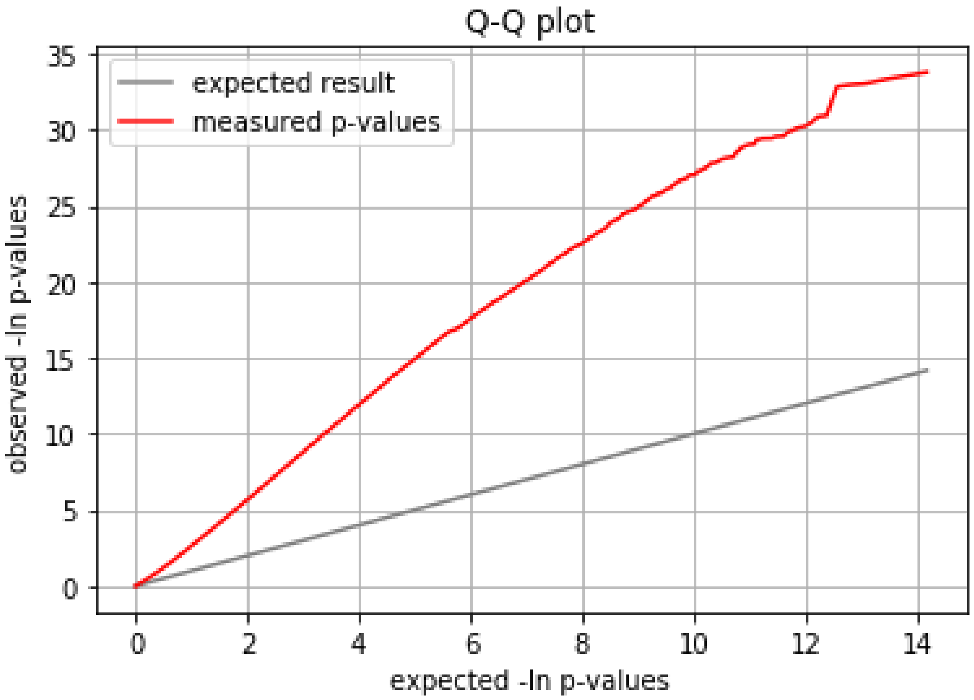


1.2 Q-Q plot

First we generate the expected p-values (uniform distributen => np.linspace), then do the required ln-transform and plot

[9]:

```
X = np.linspace(0,1,ps.size+1)[1:] # avoid to get a 0 since ln(0) is a problem
nln_X = -np.log(X)
nln_Y = -np.log(np.sort(ps))
plt.plot(nln_X, nln_X, color="grey", label="expected result")
plt.plot(nln_X, nln_Y, color="red", label="measured p-values")
plt.legend(loc="upper left")
plt.grid()
plt.xlabel("expected -ln p-values")
plt.ylabel("observed -ln p-values")
plt.title("Q-Q plot")
plt.show()
```



A Q-Q plot shows the measured p-values plotted against the expected ones. This should mostly be straight line like (y = x) with possibly some exception towards the very small p-values. In order to make it easier so see that area one uses a negative ln transform. This makes the small values more pronounced and also puts them in the top right corner. As we can see from the plot already this is not callibrated at all, since the red and grey lines do not look anythink alike.

This can also be seen from the lambda, which has ben calculated by the first plink command: > –adjust: Genomic inflation est. lambda (based on median chisq) = 3.99938.

which is significantly larger than 1

1.3 Part (b)

We do not have to change the command from a since we have already been performing the adjusted test. We have, however, to also use the adjusted output:

```
[10]: !cat 3a.model.trend.adjusted | tr -s " " "\t" > 3a.adjusted.tsv
```

```
[11]: table_b = pd.read_csv("3a.adjusted.tsv", sep="\t")
```

```
[12]: table_b.head()
```

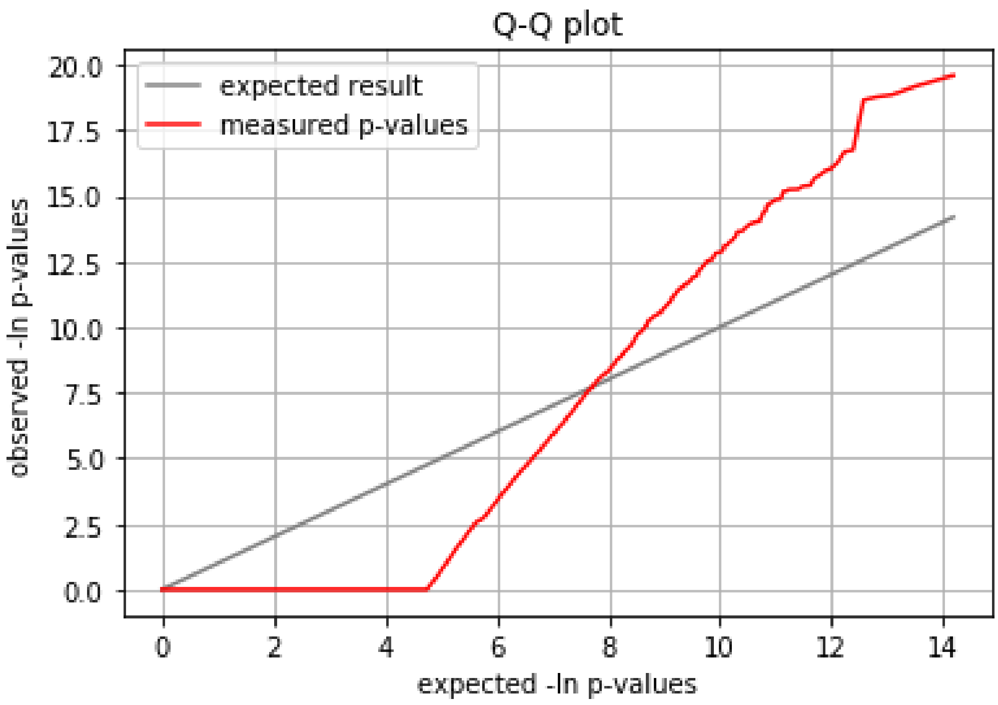
```
[12]:      Unnamed: 0         CHR           SNP     UNADJ            GC  \
      NaN           6  rs11962226  2.168000e-15  0.000073  3.123000e-09
      NaN           9   rs7031414  3.315000e-15  0.000082  4.775000e-09
      NaN           3   rs9311319  4.536000e-15  0.000089  6.534000e-09
      NaN           4  rs10007859  4.953000e-15  0.000091  7.135000e-09
      NaN           1  rs17130151  5.589000e-15  0.000094  8.050000e-09

                   BONF          HOLM      SIDAK_SS      SIDAK_SD        FDR_BH  \
      NaN  3.123000e-09  3.123000e-09  3.123000e-09  1.610000e-09  2.376000e-08
      NaN  4.775000e-09  4.775000e-09  4.775000e-09  1.610000e-09  2.376000e-08
      NaN  6.534000e-09  6.534000e-09  6.534000e-09  1.610000e-09  2.376000e-08
      NaN  7.135000e-09  7.135000e-09  7.135000e-09  1.610000e-09  2.376000e-08
      NaN  8.050000e-09  8.050000e-09  8.050000e-09  1.610000e-09  2.376000e-08

           FDR_BY
      NaN     NaN
      NaN     NaN
      NaN     NaN
      NaN     NaN
      NaN     NaN
```

Here one can see the *GC* (genomic control) column, which has the adjusted values.

```
[13]: ps_gc = table_b["GC"].dropna().to_numpy()
      ps_gc[:5]
```

```
[13]: array([3.123e-09, 4.775e-09, 6.534e-09, 7.135e-09, 8.050e-09])
```

```
[14]: X = np.linspace(0,1,ps_gc.size+1)[1:] # avoid to get a 0 since ln(0) is a␣
       ↪problem
      nln_X = -np.log(X)
      nln_Y = -np.log(np.sort(ps_gc))
      plt.plot(nln_X, nln_X, color="grey", label="expected result")
      plt.plot(nln_X, nln_Y, color="red", label="measured p-values")
      plt.legend(loc="upper left")
      plt.grid()
      plt.xlabel("expected -ln p-values")
      plt.ylabel("observed -ln p-values")
      plt.title("Q-Q plot")
      plt.show()
```



The plot still does not look good, but one can see, that the red line is now roughly centered with the grey one (it is above the grey one about as much as below). Still the red line does not follow the grey line as nicely as in the example in the slides.

1.4 Part (c)

here we have to provide the *gc* value to the *–adjust* command. This causes *plink* to use the *gc*-adjusted values in the formulas. The result we are then looking for is in the *BONF* column.

[15]:
```
!../../plink/plink --bfile ../data/plink --model trend-only --out 3c --adjust gc
```

```
PLINK v1.90b6.24 64-bit (6 Jun 2021)           www.cog-genomics.org/plink/1.9/
(C) 2005-2021 Shaun Purcell, Christopher Chang   GNU General Public License v3
Logging to 3c.log.
Options in effect:
  --adjust gc
  --bfile ../data/plink
  --model trend-only
  --out 3c

7905 MB RAM detected; reserving 3952 MB for main workspace.
1440616 variants loaded from .bim file.
619 people (305 males, 314 females) loaded from .fam.
619 phenotype values loaded from .fam.
Using 1 thread (no multithreaded calculations invoked).
Before main variant filters, 523 founders and 96 nonfounders present.
Calculating allele frequencies… 10111213141516171819202122232425262728293031
3233343536373839404142434445464748495051525354555657585960616263646566676869707172
73747576777879808182838485868788899091929394959697989 done.
Warning: 1879 het. haploid genotypes present (see 3c.hh ); many commands treat
these as missing.
Total genotyping rate is 0.99772.
1440616 variants and 619 people pass filters and QC.
Among remaining phenotypes, 119 are cases and 500 are controls.
Excluding 112 MT/haploid variants from --model analysis.
Writing --model report to 3c.model … 1011121314151617181920212223242526272829
3031323334353637383940414243444546474849505152535455565758596061626364656667686970
7172737475767778798081828384858687888990919293949596979899done.
--adjust: Genomic inflation est. lambda (based on median chisq) = 3.99938.
1011121314151617181920212223242526272829303132333435363738394041424344454647484950
5152535455565758596061626364656667686970717273747576777879808182838485868788899091
929394959697989899--adjust values (1440504 variants) written to
3c.model.trend.adjusted .
```

```
[16]: !cat 3c.model.trend.adjusted | tr -s " " "\t" > 3c.adjusted.tsv
```

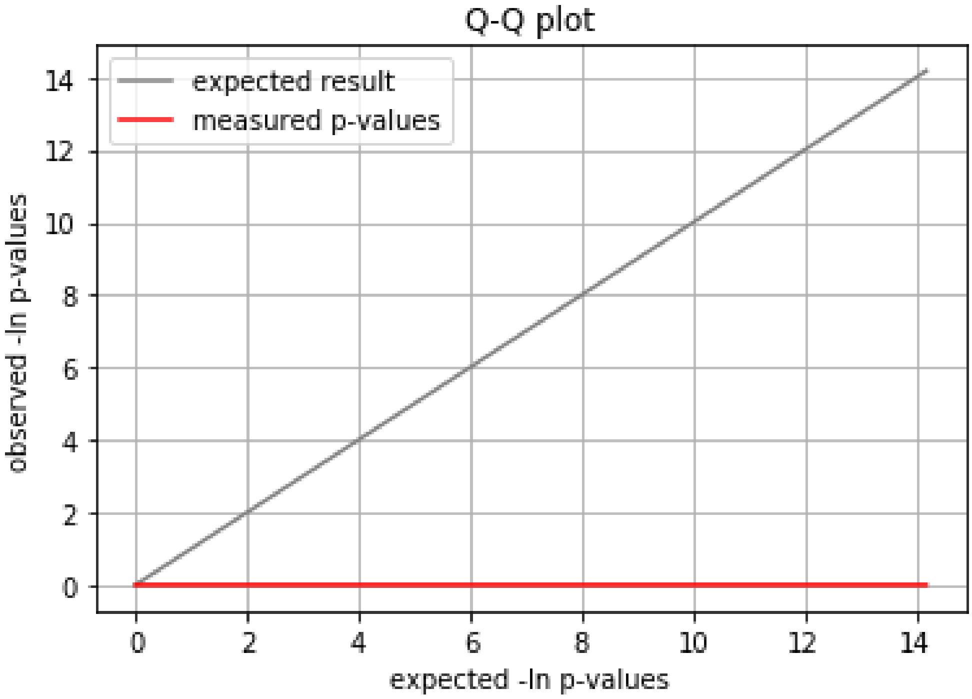
```
[17]: table_c = pd.read_csv("3c.adjusted.tsv", sep="\t")
```

```
[18]: table_c.head()
```

```
[18]:      Unnamed: 0          CHR            SNP     UNADJ  GC  BONF  HOLM  SIDAK_SS  \
     NaN           6  rs11962226  2.168000e-15  0.000073   1     1     1         1
     NaN           9   rs7031414  3.315000e-15  0.000082   1     1     1         1
     NaN           3   rs9311319  4.536000e-15  0.000089   1     1     1         1
     NaN           4  rs10007859  4.953000e-15  0.000091   1     1     1         1
     NaN           1  rs17130151  5.589000e-15  0.000094   1     1     1         1

          SIDAK_SD  FDR_BH  FDR_BY
     NaN    0.9957       1     NaN
     NaN    0.9957       1     NaN
     NaN    0.9957       1     NaN
     NaN    0.9957       1     NaN
     NaN    0.9957       1     NaN
```

```
[19]: ps_gc_bonf = table_c["GC"].dropna().to_numpy()
```

[20]:
```
X = np.linspace(0,1,ps_gc_bonf.size+1)[1:] # avoid to get a 0 since ln(0) is a␣
 ↪problem
nln_X = -np.log(X)
nln_Y = -np.log(np.sort(ps_gc_bonf))
plt.plot(nln_X, nln_X, color="grey", label="expected result")
plt.plot(nln_X, nln_Y, color="red", label="measured p-values")
plt.legend(loc="upper left")
plt.grid()
plt.xlabel("expected -ln p-values")
plt.ylabel("observed -ln p-values")
plt.title("Q-Q plot")
plt.show()
```



[21]:
```
print(np.sort(ps_gc_bonf)[:10])
```

```
[1 1 1 1 1 1 1 1 1 1]
```

We can see, that all the p-values have been *corrected* up to 1. One possible explanation is, that we have very uncalibrated data ($\lambda \approx 4$) and we do a lot of tests, so by the time both of those effects have been corrected, th results are just very uncertain (p-value of 1 is most uncertain).

1.5 Part (d)

This is probably not the proper way, to do it but it ended up giving reasonable reesults, so maybe it is not that far off after all.

The given data generates a problem, since there are chromosomes with numbers > 23 (and humans only have 23 chromosomes). So first we will note all the SNPs in chromosomes that are > 23. We cannot just delete them, since then there is a problem with incopatibilities (different lengths) between the `.bim` and `.bed` files

[22]:
```
!cat ../data/plink.bim | grep "^2[3-9]" | cut -f2 > 3d.exclude
```

Since we cannot delete the bad SNPs we rename their chromosomes to ones that are actually valid (this should not cause problems, since we will exclude them anyways).

[23]:
```
!cat ../data/plink.bim | sed "s/^2[3-9]/1/g" > plink.bim.23
```

Finally since there are too many SNPs for FastLMM to handle (it could do it, but there is a warning for more than 1 000 000 SNPs) we can prune some due to linkage disequilibrium (the idea is from the error message of FastLMM and the command from https://www.cog-genomics.org/plink/1.9/ld).

[24]:
```
!../../plink/plink -bfile ../data/plink --indep-pairwise 50 5 0.5
```

```
PLINK v1.90b6.24 64-bit (6 Jun 2021)          www.cog-genomics.org/plink/1.9/
(C) 2005-2021 Shaun Purcell, Christopher Chang   GNU General Public License v3
Logging to plink.log.
Options in effect:
  --bfile ../data/plink
  --indep-pairwise 50 5 0.5

7905 MB RAM detected; reserving 3952 MB for main workspace.
1440616 variants loaded from .bim file.
619 people (305 males, 314 females) loaded from .fam.
619 phenotype values loaded from .fam.
Using 1 thread (no multithreaded calculations invoked).
Before main variant filters, 523 founders and 96 nonfounders present.
Calculating allele frequencies… 1011121314151617181920212223242526272829303132
33343536373839404142434445464748495051525354555657585960616263646566676869707172
73747576777879808182838485868788899091929394959697989 done.
Warning: 1879 het. haploid genotypes present (see plink.hh ); many commands
treat these as missing.
Total genotyping rate is 0.99772.
1440616 variants and 619 people pass filters and QC.
Among remaining phenotypes, 119 are cases and 500 are controls.
Pruned 75937 variants from chromosome 1, leaving 40628.
Pruned 77042 variants from chromosome 2, leaving 39540.
Pruned 63102 variants from chromosome 3, leaving 33569.
Pruned 55824 variants from chromosome 4, leaving 30076.
Pruned 57509 variants from chromosome 5, leaving 30556.
Pruned 60339 variants from chromosome 6, leaving 31161.
Pruned 48616 variants from chromosome 7, leaving 26819.
Pruned 49402 variants from chromosome 8, leaving 25975.
Pruned 40620 variants from chromosome 9, leaving 23086.
Pruned 48077 variants from chromosome 10, leaving 25859.
Pruned 46835 variants from chromosome 11, leaving 24257.
Pruned 43987 variants from chromosome 12, leaving 24652.
Pruned 33434 variants from chromosome 13, leaving 18577.
Pruned 28920 variants from chromosome 14, leaving 16637.
Pruned 26414 variants from chromosome 15, leaving 15998.
Pruned 27265 variants from chromosome 16, leaving 17445.
Pruned 23179 variants from chromosome 17, leaving 15278.
Pruned 25598 variants from chromosome 18, leaving 15293.
Pruned 15191 variants from chromosome 19, leaving 11072.
Pruned 22576 variants from chromosome 20, leaving 13726.
Pruned 11920 variants from chromosome 21, leaving 7411.
Pruned 11918 variants from chromosome 22, leaving 8191.
Pruned 34937 variants from chromosome 23, leaving 15572.
Pruned 138 variants from chromosome 25, leaving 346.
Pruned 47 variants from chromosome 26, leaving 65.
Pruning complete.  928827 of 1440616 variants removed.
Marker lists written to plink.prune.in and plink.prune.out .
```

Now with some SNPs prunded (we are now down to about 500 000), we can run the BOLT_LMM command unsing the changes files from above (the `--LDscoresUseChip` again was given in an error message).

[25]:
```
!../../BOLT-LMM_v2.3.5/bolt --fam=../data/plink.fam --bim=plink.bim.23 --bed=../
↪data/plink.bed --phenoUseFam --lmm --statsFile 3d --LDscoresUseChip␣
↪--exclude 3d.exclude --exclude plink.prune.out --numThreads 8
```

```
                      +-----------------------------+
                      |                        ___  |
                      |   BOLT-LMM, v2.3.5   /_ /   |
                      |   March 20, 2021      /_/   |
                      |   Po-Ru Loh            //   |
                      |                        /    |
                      +-----------------------------+

Copyright (C) 2014-2021 Harvard University.
Distributed under the GNU GPLv3 open source license.

Compiled with USE_SSE: fast aligned memory access
Compiled with USE_MKL: Intel Math Kernel Library linear algebra
Boost version: 1_58

Command line options:

../../BOLT-LMM_v2.3.5/bolt \
    --fam=../data/plink.fam \
    --bim=plink.bim.23 \
    --bed=../data/plink.bed \
    --phenoUseFam \
    --lmm \
    --statsFile 3d \
    --LDscoresUseChip \
    --exclude 3d.exclude \
    --exclude plink.prune.out \
    --numThreads 8

Setting number of threads to 8
fam: ../data/plink.fam
bim(s): plink.bim.23
bed(s): ../data/plink.bed

=== Reading genotype data ===

Total indivs in PLINK data: Nbed = 619
Total indivs stored in memory: N = 619
Reading bim file #1: plink.bim.23
WARNING: Out-of-order snp in bim file: plink.bim.23
Line 1596:
1       AFFX-SNP_11906976__rs581070     0       4289059 C       G
WARNING: Out-of-order snp in bim file: plink.bim.23
Line 2223:
1       AFFX-SNP_6869948__rs10915322    0       4997214 A       G
WARNING: Out-of-order snp in bim file: plink.bim.23
Line 2273:
1       AFFX-SNP_4600__rs9439505        0       5093533 T       C
WARNING: Out-of-order snp in bim file: plink.bim.23
Line 2320:
1       AFFX-SNP_4992__rs7536712        0       5153524 T       C
WARNING: Out-of-order snp in bim file: plink.bim.23
Line 2445:
1       AFFX-SNP_9025902__rs4847565     0       5284188 G       A
WARNING: Total number of out-of-order snps in bim file: 2051
    Read 1440616 snps
Total snps in PLINK data: Mbed = 1440616
Reading exclude file (SNPs to exclude): 3d.exclude
Excluded 51105 SNP(s)
Reading exclude file (SNPs to exclude): plink.prune.out
Excluded 893705 SNP(s)

Breakdown of SNP pre-filtering results:
  495806 SNPs to include in model (i.e., GRM)
  0 additional non-GRM SNPs loaded
  944810 excluded SNPs
WARNING: No genetic map provided; using physical positions only
Allocating 495806 x 620/4 bytes to store genotypes
Reading genotypes and performing QC filtering on snps and indivs…
Reading bed file #1: ../data/plink.bed
```

```
    Expecting 223295480 (+3) bytes for 619 indivs, 1440616 snps
Total indivs after QC: 619
Total post-QC SNPs: M = 495806
  Variance component 1: 495806 post-QC SNPs (name: 'modelSnps')
Time for SnpData setup = 10.2737 sec

=== Reading phenotype and covariate data ===

Number of indivs with no missing phenotype(s) to use: 619
NOTE: Using all-1s vector (constant term) in addition to specified covariates
    Using quantitative covariate: CONST_ALL_ONES
Number of individuals used in analysis: Nused = 619
Singular values of covariate matrix:
    S[0] = 24.8797
Total covariate vectors: C = 1
Total independent covariate vectors: Cindep = 1

=== Initializing Bolt object: projecting and normalizing SNPs ===

Number of chroms with >= 1 good SNP: 22
Average norm of projected SNPs:           618.000000
Dimension of all-1s proj space (Nused-1): 618
Time for covariate data setup + Bolt initialization = 1.68237 sec

Phenotype 1:   N = 619   mean = 1.19225   std = 0.394384

=== Computing linear regression (LINREG) stats ===

Time for computing LINREG stats = 0.389594 sec

=== Estimating variance parameters ===

Using CGtol of 0.005 for this step
Using default number of random trials: 15 (for Nused = 619)

Estimating MC scaling f_REML at log(delta) = 1.09861, h2 = 0.25…
  Batch-solving 16 systems of equations using conjugate gradient iteration
  iter 1:  time=2.34  rNorms/orig: (0.6,2)  res2s: 1097.46..18.0129
  iter 2:  time=1.71  rNorms/orig: (0.1,0.5)  res2s: 1197.42..40.9796
  iter 3:  time=1.91  rNorms/orig: (0.03,0.2)  res2s: 1225.58..45.2024
  iter 4:  time=1.65  rNorms/orig: (0.02,0.04)  res2s: 1263.1..48.3579
  iter 5:  time=1.76  rNorms/orig: (0.003,0.008)  res2s: 1267.37..48.4496
  iter 6:  time=1.91  rNorms/orig: (0.0007,0.002)  res2s: 1267.71..48.4583
  Converged at iter 6: rNorms/orig all < CGtol=0.005
  Time breakdown: dgemm = 55.8%, memory/overhead = 44.2%
  MCscaling: logDelta = 1.10, h2 = 0.250, f = 0.203599

Estimating MC scaling f_REML at log(delta) = 0, h2 = 0.5…
```

```
  Batch-solving 16 systems of equations using conjugate gradient iteration
  iter 1:  time=1.73  rNorms/orig: (2,2)  res2s: 152.875..3.95465
  iter 2:  time=1.62  rNorms/orig: (0.3,1)  res2s: 242.219..15.0849
  iter 3:  time=1.73  rNorms/orig: (0.2,0.4)  res2s: 259.345..18.4906
  iter 4:  time=1.81  rNorms/orig: (0.07,0.2)  res2s: 299.292..23.5421
  iter 5:  time=1.81  rNorms/orig: (0.03,0.05)  res2s: 307.252..23.948
  iter 6:  time=1.76  rNorms/orig: (0.01,0.02)  res2s: 309.136..24.0482
  iter 7:  time=1.70  rNorms/orig: (0.004,0.007)  res2s: 309.566..24.069
  iter 8:  time=1.65  rNorms/orig: (0.001,0.002)  res2s: 309.623..24.0724
  Converged at iter 8: rNorms/orig all < CGtol=0.005
  Time breakdown: dgemm = 55.0%, memory/overhead = 45.0%
  MCscaling: logDelta = 0.00, h2 = 0.500, f = 0.0719073

Estimating MC scaling f_REML at log(delta) = -0.599873, h2 = 0.645627…
  Batch-solving 16 systems of equations using conjugate gradient iteration
  iter 1:  time=1.73  rNorms/orig: (2,2)  res2s: 39.9569..1.4435
  iter 2:  time=1.85  rNorms/orig: (0.4,1)  res2s: 85.6996..6.84329
  iter 3:  time=1.64  rNorms/orig: (0.3,0.6)  res2s: 95.6316..8.868
  iter 4:  time=1.69  rNorms/orig: (0.1,0.2)  res2s: 120.508..12.7935
  iter 5:  time=1.79  rNorms/orig: (0.05,0.09)  res2s: 126.586..13.2451
  iter 6:  time=1.69  rNorms/orig: (0.03,0.04)  res2s: 128.556..13.3987
  iter 7:  time=1.82  rNorms/orig: (0.01,0.02)  res2s: 129.301..13.4468
  iter 8:  time=1.64  rNorms/orig: (0.004,0.006)  res2s: 129.478..13.4596
  iter 9:  time=1.73  rNorms/orig: (0.002,0.002)  res2s: 129.509..13.4629
  Converged at iter 9: rNorms/orig all < CGtol=0.005
  Time breakdown: dgemm = 57.0%, memory/overhead = 43.0%
  MCscaling: logDelta = -0.60, h2 = 0.646, f = 0.0268059

Estimating MC scaling f_REML at log(delta) = -0.956406, h2 = 0.722402…
  Batch-solving 16 systems of equations using conjugate gradient iteration
  iter 1:  time=1.74  rNorms/orig: (2,3)  res2s: 17.4538..0.762534
  iter 2:  time=2.01  rNorms/orig: (0.4,1)  res2s: 44.3168..3.98776
  iter 3:  time=4.47  rNorms/orig: (0.4,0.8)  res2s: 50.7492..5.31301
  iter 4:  time=2.55  rNorms/orig: (0.1,0.2)  res2s: 67.2869..8.22528
  iter 5:  time=1.82  rNorms/orig: (0.07,0.1)  res2s: 71.7183..8.61819
  iter 6:  time=1.87  rNorms/orig: (0.04,0.05)  res2s: 73.2865..8.77167
  iter 7:  time=1.71  rNorms/orig: (0.02,0.03)  res2s: 74.0444..8.82938
  iter 8:  time=1.98  rNorms/orig: (0.008,0.01)  res2s: 74.2836..8.84875
  iter 9:  time=2.27  rNorms/orig: (0.003,0.005)  res2s: 74.3373..8.85495
  Converged at iter 9: rNorms/orig all < CGtol=0.005
  Time breakdown: dgemm = 49.3%, memory/overhead = 50.7%
  MCscaling: logDelta = -0.96, h2 = 0.722, f = 0.00817042

Estimating MC scaling f_REML at log(delta) = -1.11272, h2 = 0.752636…
  Batch-solving 16 systems of equations using conjugate gradient iteration
  iter 1:  time=1.46  rNorms/orig: (2,3)  res2s: 12.1377..0.57247
  iter 2:  time=1.50  rNorms/orig: (0.5,2)  res2s: 32.9609..3.10373
  iter 3:  time=1.49  rNorms/orig: (0.4,0.8)  res2s: 38.1615..4.17945
  iter 4:  time=1.58  rNorms/orig: (0.2,0.3)  res2s: 51.6553..6.65785
  iter 5:  time=1.49  rNorms/orig: (0.07,0.1)  res2s: 55.4047..7.01248
  iter 6:  time=1.89  rNorms/orig: (0.04,0.06)  res2s: 56.7528..7.15814
  iter 7:  time=1.50  rNorms/orig: (0.02,0.04)  res2s: 57.4706..7.21677
  iter 8:  time=1.46  rNorms/orig: (0.01,0.01)  res2s: 57.7236..7.23824
  iter 9:  time=1.54  rNorms/orig: (0.004,0.006)  res2s: 57.7858..7.24574
  iter 10:  time=1.51  rNorms/orig: (0.002,0.002)  res2s: 57.7985..7.24713
  Converged at iter 10: rNorms/orig all < CGtol=0.005
  Time breakdown: dgemm = 56.3%, memory/overhead = 43.7%
  MCscaling: logDelta = -1.11, h2 = 0.753, f = 0.00153066

Estimating MC scaling f_REML at log(delta) = -1.14876, h2 = 0.759284…
  Batch-solving 16 systems of equations using conjugate gradient iteration
  iter 1:  time=1.64  rNorms/orig: (2,3)  res2s: 11.165..0.535585
  iter 2:  time=1.61  rNorms/orig: (0.5,2)  res2s: 30.7718..2.92634
  iter 3:  time=1.36  rNorms/orig: (0.4,0.8)  res2s: 35.7152..3.94973
  iter 4:  time=1.55  rNorms/orig: (0.2,0.3)  res2s: 48.5667..6.3319
  iter 5:  time=1.46  rNorms/orig: (0.07,0.1)  res2s: 52.1667..6.6771
  iter 6:  time=1.41  rNorms/orig: (0.04,0.06)  res2s: 53.4634..6.82043
  iter 7:  time=1.45  rNorms/orig: (0.02,0.04)  res2s: 54.1688..6.87898
  iter 8:  time=1.37  rNorms/orig: (0.01,0.01)  res2s: 54.4236..6.90083
  iter 9:  time=1.47  rNorms/orig: (0.005,0.006)  res2s: 54.4874..6.90861
  iter 10:  time=1.50  rNorms/orig: (0.002,0.003)  res2s: 54.5007..6.91007
  Converged at iter 10: rNorms/orig all < CGtol=0.005
  Time breakdown: dgemm = 58.7%, memory/overhead = 41.3%
  MCscaling: logDelta = -1.15, h2 = 0.759, f = 0.000109975

Secant iteration for h2 estimation converged in 4 steps
Estimated (pseudo-)heritability: h2g = 0.759
To more precisely estimate variance parameters and estimate s.e., use --reml
Variance params: sigma^2_K = 0.116254, logDelta = -1.148757, f = 0.000109975

Time for fitting variance components = 98.566 sec

=== Computing mixed model assoc stats (inf. model) ===

Selected 30 SNPs for computation of prospective stat
Tried 30; threw out 0 with GRAMMAR chisq > 5
Assigning SNPs to 22 chunks for leave-out analysis
Each chunk is excluded when testing SNPs belonging to the chunk
  Batch-solving 52 systems of equations using conjugate gradient iteration
  iter 1:  time=2.73  rNorms/orig: (1,3)  res2s: 0.538387..1.11321
  iter 2:  time=2.98  rNorms/orig: (0.2,2)  res2s: 2.92479..25.5659
  iter 3:  time=3.05  rNorms/orig: (0.2,0.9)  res2s: 3.94232..28.904
  iter 4:  time=2.95  rNorms/orig: (0.02,0.3)  res2s: 6.31714..37.1859
  iter 5:  time=3.03  rNorms/orig: (0.01,0.1)  res2s: 6.65944..40.2549
  iter 6:  time=2.83  rNorms/orig: (0.006,0.06)  res2s: 6.8044..41.2866
  iter 7:  time=3.11  rNorms/orig: (0.002,0.03)  res2s: 6.86191..41.6184
  iter 8:  time=3.19  rNorms/orig: (0.0009,0.01)  res2s: 6.88329..41.755
  iter 9:  time=3.08  rNorms/orig: (0.0005,0.006)  res2s: 6.89117..41.7828
  iter 10:  time=3.22  rNorms/orig: (0.0002,0.002)  res2s: 6.89261..41.7888
  iter 11:  time=3.03  rNorms/orig: (0.0001,0.0009)  res2s: 6.89291..41.79
  iter 12:  time=3.08  rNorms/orig: (0.0002,0.002)  res2s: 6.89296..41.7903
  iter 13:  time=2.91  rNorms/orig: (4e-05,0.0008)  res2s: 6.89296..41.7902
  iter 14:  time=2.82  rNorms/orig: (2e-05,0.0002)  res2s: 6.89297..41.7903
  Converged at iter 14: rNorms/orig all < CGtol=0.0005
  Time breakdown: dgemm = 69.9%, memory/overhead = 30.1%

AvgPro: 0.828   AvgRetro: 0.592   Calibration: 1.398 (0.096)   (30 SNPs)
Ratio of medians: 1.212   Median of ratios: 1.298
WARNING: Calibration std error is high; consider increasing --numCalibSnps
         Using ratio of medians instead: 1.21167

Time for computing infinitesimal model assoc stats = 42.764 sec

=== Estimating chip LD Scores using 400 indivs ===


Time for estimating chip LD Scores = 2.20954 sec

WARNING: No LDscoresFile provided; using estimated LD among chip SNPs

=== Estimating mixture parameters by cross-validation ===

Setting maximum number of iterations to 250 for this step
Max CV folds to compute = 5 (to have > 10000 samples)

====> Starting CV fold 1 <====

NOTE: Using all-1s vector (constant term) in addition to specified covariates
    Using quantitative covariate: CONST_ALL_ONES
Number of individuals used in analysis: Nused = 495
Singular values of covariate matrix:
    S[0] = 22.2486
Total covariate vectors: C = 1
Total independent covariate vectors: Cindep = 1

=== Initializing Bolt object: projecting and normalizing SNPs ===

Number of chroms with >= 1 good SNP: 22
Average norm of projected SNPs:           494.000000
Dimension of all-1s proj space (Nused-1): 494
  Beginning variational Bayes
  iter 1:  time=5.55 for 18 active reps
  iter 2:  time=4.19 for 18 active reps  approxLL diffs: (209.55,297.81)
  iter 3:  time=3.61 for 18 active reps  approxLL diffs: (31.26,55.38)
  iter 4:  time=3.94 for 18 active reps  approxLL diffs: (12.37,17.69)
  iter 5:  time=3.89 for 18 active reps  approxLL diffs: (6.33,8.07)
  iter 6:  time=3.66 for 18 active reps  approxLL diffs: (3.77,4.35)
  iter 7:  time=3.84 for 18 active reps  approxLL diffs: (2.46,2.65)
  iter 8:  time=3.90 for 18 active reps  approxLL diffs: (1.72,1.79)
  iter 9:  time=3.68 for 18 active reps  approxLL diffs: (1.25,1.31)
  iter 10:  time=3.59 for 18 active reps  approxLL diffs: (0.91,0.99)
  iter 11:  time=3.55 for 18 active reps  approxLL diffs: (0.68,0.74)
  iter 12:  time=3.75 for 18 active reps  approxLL diffs: (0.53,0.57)
  iter 13:  time=3.92 for 18 active reps  approxLL diffs: (0.44,0.46)
  iter 14:  time=3.61 for 18 active reps  approxLL diffs: (0.37,0.39)
  iter 15:  time=3.55 for 18 active reps  approxLL diffs: (0.31,0.33)
  iter 16:  time=4.05 for 18 active reps  approxLL diffs: (0.27,0.28)
  iter 17:  time=4.14 for 18 active reps  approxLL diffs: (0.23,0.24)
  iter 18:  time=4.10 for 18 active reps  approxLL diffs: (0.20,0.21)
  iter 19:  time=3.74 for 18 active reps  approxLL diffs: (0.18,0.19)
  iter 20:  time=4.48 for 18 active reps  approxLL diffs: (0.16,0.17)
  iter 21:  time=3.83 for 18 active reps  approxLL diffs: (0.15,0.16)
  iter 22:  time=4.70 for 18 active reps  approxLL diffs: (0.13,0.15)
  iter 23:  time=3.76 for 18 active reps  approxLL diffs: (0.12,0.13)
  iter 24:  time=3.74 for 18 active reps  approxLL diffs: (0.11,0.12)
  iter 25:  time=3.74 for 18 active reps  approxLL diffs: (0.11,0.12)
  iter 26:  time=4.63 for 18 active reps  approxLL diffs: (0.10,0.11)
  iter 27:  time=3.98 for 18 active reps  approxLL diffs: (0.09,0.10)
  iter 28:  time=4.50 for 18 active reps  approxLL diffs: (0.09,0.10)
  iter 29:  time=5.13 for 18 active reps  approxLL diffs: (0.08,0.09)
  iter 30:  time=4.25 for 18 active reps  approxLL diffs: (0.08,0.08)
  iter 31:  time=3.87 for 18 active reps  approxLL diffs: (0.07,0.08)
  iter 32:  time=3.66 for 18 active reps  approxLL diffs: (0.07,0.08)
  iter 33:  time=3.61 for 18 active reps  approxLL diffs: (0.07,0.07)
  iter 34:  time=3.60 for 18 active reps  approxLL diffs: (0.06,0.07)
  iter 35:  time=3.65 for 18 active reps  approxLL diffs: (0.06,0.07)
  iter 36:  time=3.65 for 18 active reps  approxLL diffs: (0.06,0.06)
  iter 37:  time=3.50 for 18 active reps  approxLL diffs: (0.05,0.06)
  iter 38:  time=3.49 for 18 active reps  approxLL diffs: (0.05,0.06)
  iter 39:  time=3.39 for 18 active reps  approxLL diffs: (0.05,0.05)
  iter 40:  time=3.42 for 18 active reps  approxLL diffs: (0.05,0.05)
  iter 41:  time=3.52 for 18 active reps  approxLL diffs: (0.05,0.05)
  iter 42:  time=3.41 for 18 active reps  approxLL diffs: (0.04,0.05)
  iter 43:  time=3.34 for 18 active reps  approxLL diffs: (0.04,0.05)
  iter 44:  time=3.79 for 18 active reps  approxLL diffs: (0.04,0.04)
  iter 45:  time=3.57 for 18 active reps  approxLL diffs: (0.04,0.04)
  iter 46:  time=3.52 for 18 active reps  approxLL diffs: (0.04,0.04)
  iter 47:  time=3.72 for 18 active reps  approxLL diffs: (0.04,0.04)
  iter 48:  time=3.64 for 18 active reps  approxLL diffs: (0.04,0.04)
  iter 49:  time=3.70 for 18 active reps  approxLL diffs: (0.04,0.04)
  iter 50:  time=3.65 for 18 active reps  approxLL diffs: (0.03,0.04)
  iter 51:  time=3.59 for 18 active reps  approxLL diffs: (0.03,0.03)
  iter 52:  time=3.69 for 18 active reps  approxLL diffs: (0.03,0.03)
  iter 53:  time=3.65 for 18 active reps  approxLL diffs: (0.03,0.03)
  iter 54:  time=3.59 for 18 active reps  approxLL diffs: (0.03,0.03)
  iter 55:  time=3.64 for 18 active reps  approxLL diffs: (0.03,0.03)
  iter 56:  time=3.67 for 18 active reps  approxLL diffs: (0.03,0.03)
  iter 57:  time=3.63 for 18 active reps  approxLL diffs: (0.03,0.03)
  iter 58:  time=3.62 for 18 active reps  approxLL diffs: (0.03,0.03)
  iter 59:  time=3.49 for 18 active reps  approxLL diffs: (0.03,0.03)
  iter 60:  time=3.65 for 18 active reps  approxLL diffs: (0.02,0.03)
  iter 61:  time=3.62 for 18 active reps  approxLL diffs: (0.02,0.03)
  iter 62:  time=3.68 for 18 active reps  approxLL diffs: (0.02,0.03)
  iter 63:  time=3.74 for 18 active reps  approxLL diffs: (0.02,0.02)
  iter 64:  time=3.71 for 18 active reps  approxLL diffs: (0.02,0.02)
  iter 65:  time=3.69 for 18 active reps  approxLL diffs: (0.02,0.02)
  iter 66:  time=3.66 for 18 active reps  approxLL diffs: (0.02,0.02)
  iter 67:  time=3.68 for 18 active reps  approxLL diffs: (0.02,0.02)
  iter 68:  time=3.61 for 18 active reps  approxLL diffs: (0.02,0.02)
  iter 69:  time=3.57 for 18 active reps  approxLL diffs: (0.02,0.02)
  iter 70:  time=3.62 for 18 active reps  approxLL diffs: (0.02,0.02)
  iter 71:  time=3.93 for 18 active reps  approxLL diffs: (0.02,0.02)
  iter 72:  time=3.87 for 18 active reps  approxLL diffs: (0.02,0.02)
  iter 73:  time=3.84 for 18 active reps  approxLL diffs: (0.02,0.02)
  iter 74:  time=3.61 for 18 active reps  approxLL diffs: (0.02,0.02)
  iter 75:  time=3.67 for 18 active reps  approxLL diffs: (0.02,0.02)
  iter 76:  time=3.72 for 18 active reps  approxLL diffs: (0.02,0.02)
  iter 77:  time=4.15 for 18 active reps  approxLL diffs: (0.02,0.02)
  iter 78:  time=3.57 for 18 active reps  approxLL diffs: (0.02,0.02)
  iter 79:  time=3.79 for 18 active reps  approxLL diffs: (0.02,0.02)
  iter 80:  time=3.52 for 18 active reps  approxLL diffs: (0.02,0.02)
  iter 81:  time=3.68 for 18 active reps  approxLL diffs: (0.02,0.02)
  iter 82:  time=3.77 for 18 active reps  approxLL diffs: (0.02,0.02)
  iter 83:  time=3.62 for 18 active reps  approxLL diffs: (0.02,0.02)
  iter 84:  time=3.60 for 18 active reps  approxLL diffs: (0.02,0.02)
  iter 85:  time=3.70 for 18 active reps  approxLL diffs: (0.02,0.02)
  iter 86:  time=3.82 for 18 active reps  approxLL diffs: (0.01,0.02)
  iter 87:  time=4.78 for 18 active reps  approxLL diffs: (0.01,0.02)
  iter 88:  time=4.59 for 18 active reps  approxLL diffs: (0.01,0.01)
  iter 89:  time=5.28 for 18 active reps  approxLL diffs: (0.01,0.01)
  iter 90:  time=4.28 for 18 active reps  approxLL diffs: (0.01,0.01)
  iter 91:  time=4.56 for 18 active reps  approxLL diffs: (0.01,0.01)
  iter 92:  time=3.98 for 18 active reps  approxLL diffs: (0.01,0.01)
  iter 93:  time=4.24 for 18 active reps  approxLL diffs: (0.01,0.01)
  iter 94:  time=3.40 for 18 active reps  approxLL diffs: (0.01,0.01)
  iter 95:  time=3.38 for 18 active reps  approxLL diffs: (0.01,0.01)
  iter 96:  time=3.44 for 18 active reps  approxLL diffs: (0.01,0.01)
  iter 97:  time=3.51 for 18 active reps  approxLL diffs: (0.01,0.01)
  iter 98:  time=3.44 for 18 active reps  approxLL diffs: (0.01,0.01)
  iter 99:  time=3.38 for 18 active reps  approxLL diffs: (0.01,0.01)
```

```
  iter 100:  time=3.34 for 18 active reps  approxLL diffs: (0.01,0.01)
  iter 101:  time=3.30 for 18 active reps  approxLL diffs: (0.01,0.01)
  iter 102:  time=3.35 for 18 active reps  approxLL diffs: (0.01,0.01)
  iter 103:  time=3.36 for 18 active reps  approxLL diffs: (0.01,0.01)
  iter 104:  time=3.38 for 18 active reps  approxLL diffs: (0.01,0.01)
  iter 105:  time=3.43 for 18 active reps  approxLL diffs: (0.01,0.01)
  iter 106:  time=3.44 for 18 active reps  approxLL diffs: (0.01,0.01)
  iter 107:  time=3.29 for 18 active reps  approxLL diffs: (0.01,0.01)
  iter 108:  time=3.56 for 18 active reps  approxLL diffs: (0.01,0.01)
  iter 109:  time=3.74 for 18 active reps  approxLL diffs: (0.01,0.01)
  iter 110:  time=3.48 for 18 active reps  approxLL diffs: (0.01,0.01)
  iter 111:  time=3.58 for 18 active reps  approxLL diffs: (0.01,0.01)
  iter 112:  time=3.23 for 17 active reps  approxLL diffs: (0.01,0.01)
  iter 113:  time=0.67 for  1 active reps  approxLL diffs: (0.01,0.01)
  iter 114:  time=0.69 for  1 active reps  approxLL diffs: (0.01,0.01)
  Converged at iter 114: approxLL diffs each have been < LLtol=0.01
  Time breakdown: dgemm = 32.7%, memory/overhead = 67.3%
Computing predictions on left-out cross-validation fold
Time for computing predictions = 1.80839 sec

Average PVEs obtained by param pairs tested (high to low):
  f2=0.5, p=0.5: 0.301579
  f2=0.3, p=0.5: 0.301555
  f2=0.5, p=0.2: 0.301496
            …
 f2=0.1, p=0.01: 0.296609

Detailed CV fold results:
  Absolute prediction MSE baseline (covariates only): 0.146285
  Absolute prediction MSE using standard LMM:         0.102169
  Absolute prediction MSE, fold-best  f2=0.5, p=0.5:  0.102169
    Absolute pred MSE using   f2=0.5, p=0.5: 0.102169
    Absolute pred MSE using   f2=0.5, p=0.2: 0.102181
    Absolute pred MSE using   f2=0.5, p=0.1: 0.102207
    Absolute pred MSE using  f2=0.5, p=0.05: 0.102259
    Absolute pred MSE using  f2=0.5, p=0.02: 0.102405
    Absolute pred MSE using  f2=0.5, p=0.01: 0.102604
    Absolute pred MSE using   f2=0.3, p=0.5: 0.102172
    Absolute pred MSE using   f2=0.3, p=0.2: 0.102203
    Absolute pred MSE using   f2=0.3, p=0.1: 0.102254
    Absolute pred MSE using  f2=0.3, p=0.05: 0.102355
    Absolute pred MSE using  f2=0.3, p=0.02: 0.102610
    Absolute pred MSE using  f2=0.3, p=0.01: 0.102860
    Absolute pred MSE using   f2=0.1, p=0.5: 0.102183
    Absolute pred MSE using   f2=0.1, p=0.2: 0.102235
    Absolute pred MSE using   f2=0.1, p=0.1: 0.102320
    Absolute pred MSE using  f2=0.1, p=0.05: 0.102479
    Absolute pred MSE using  f2=0.1, p=0.02: 0.102806
    Absolute pred MSE using  f2=0.1, p=0.01: 0.102896

====> End CV fold 1: 18 remaining param pair(s) <====

Estimated proportion of variance explained using inf model: 0.302
Relative improvement in prediction MSE using non-inf model: 0.000

Exiting CV: non-inf model does not substantially improve prediction
Optimal mixture parameters according to CV: f2 = 0.5, p = 0.5
Bayesian non-infinitesimal model does not fit substantially better
=> Not computing non-inf assoc stats (to override, use --lmmForceNonInf)

Time for estimating mixture parameters = 426.859 sec

Calibration stats: mean and lambdaGC (over SNPs used in GRM)
  (note that both should be >1 because of polygenicity)
Mean BOLT_LMM_INF: 0.887406 (495806 good SNPs)   lambdaGC: 0.806821

=== Streaming genotypes to compute and write assoc stats at all SNPs ===

Time for streaming genotypes and writing output = 8.53651 sec

Total elapsed time for analysis = 591.281 sec
```

1.5.1 Analyis of the result in python

Now we can inport the generated file into pandas and have a look at it

[26]:
```
lmm_data = pd.read_csv("3d", sep="\t")
```

[27]:
```
lmm_data.head()
```

[27]:
```
          SNP  CHR      BP  GENPOS ALLELE1 ALLELE0    A1FREQ    F_MISS  \
0  rs10458597    1  554484       0       T       C  0.013934  0.014540
1   rs2185539    1  556738       0       T       C  0.005654  0.000000
2  rs11240767    1  718814       0       T       C  0.020194  0.000000
3   rs3131969    1  744045       0       A       G  0.277414  0.012924
4  rs12562034    1  758311       0       A       G  0.258592  0.012924

       BETA        SE  P_BOLT_LMM_INF
0 -0.006496  0.080937            0.94
1  0.194531  0.146021            0.18
2  0.004044  0.072609            0.96
3 -0.000252  0.022920            0.99
4 -0.022805  0.023099            0.32
```

For a quick check we can have a look at the smalles (strongest) p-values, which are pretty small so there is a chance this is good.

```
[28]: lmm_data.sort_values("P_BOLT_LMM_INF").head()
```

```
[28]:                SNP  CHR         BP  GENPOS ALLELE1 ALLELE0    A1FREQ  \
      130871  rs17042171    4  111927736       0       A       C  0.297254
      402721  rs17158372   15   81144053       0       G       A  0.019418
      410483  rs12444503   16   10696526       0       A       G  0.004854
      487969   rs8190314   22   16606771       0       A       G  0.012195
      291319   rs4253212   10   50348218       0       A       G  0.039580

                F_MISS      BETA        SE  P_BOLT_LMM_INF
      130871  0.000000  0.140726  0.022166    2.200000e-10
      402721  0.001616  0.453361  0.076621    3.300000e-09
      410483  0.001616  0.839470  0.157517    9.900000e-08
      487969  0.006462  0.528830  0.100227    1.300000e-07
      291319  0.000000  0.292594  0.055867    1.600000e-07
```

Since we already have all the data in Python, we just quickly do the Bonferroni correction here (i.e. multiply the p-values by the number of tests).

```
[29]: n = lmm_data.size
      print(n)
```

```
5453866
```

```
[30]: p_corrected = np.clip(lmm_data.P_BOLT_LMM_INF.to_numpy() * n, 0, 1)
```

```
[31]: for i, p in enumerate(np.sort(p_corrected)[:10]):
          print(f"The {i+1}-smallest p-value is {(p*100):2.1f}% and therefore {'' if␣
       ↪p < 0.05 else 'not '}significant")
```

```
The 1-smallest p-value is 0.1% and therefore significant
The 2-smallest p-value is 1.8% and therefore significant
The 3-smallest p-value is 54.0% and therefore not significant
The 4-smallest p-value is 70.9% and therefore not significant
The 5-smallest p-value is 87.3% and therefore not significant
The 6-smallest p-value is 100.0% and therefore not significant
The 7-smallest p-value is 100.0% and therefore not significant
The 8-smallest p-value is 100.0% and therefore not significant
The 9-smallest p-value is 100.0% and therefore not significant
The 10-smallest p-value is 100.0% and therefore not significant
```

These are the smallest p-values after correction. We can see, that the first two are smaller than 5%, so we have found two significant results.

1.5.2 Comparison

These p-values are much better than the ones from before. Before we had (after calibration and correction for multiple testing) all p-values 1. This is obviously useless. Before these correctionswe had so many very small p-values, that it is unrealistic, that this is correct.

Now we get two significant p-values, which are not overly small either, and everything else is much bigger and therefore insignificant. This appears much more reasonable.