Untitled

Contents

1 Simulation I. Prediction accuracy

1.1 Simulation scheme

Following Lee et al. simulation II to compare prediction accuracy in unrelated and related individuals by chosing the effective number of chromosome segments (M_{eff}) to be 50,000 or 10,000 respectively. Assume $h^2 = 0.5$ and life-time risk = $K = 0.1$ (need to change this for realistic simulation of ALS, but $K = 0.0025$ takes much longer to simulate).

1. Draw causal effect sizes from three different distributions:

- *large effect*. 50 Rare variants with large effects explaining 10% of the phenotypic variance. The causal allele-frequency $p \sim U(0.0001, 0.001)$ and per-variant $\beta = h^2_{SNP} \sim N(0, 0.1/20)$
- *small effect*. 1,000 Common variants with small effects explaining 40% of the phenotypic variance. Here $p \sim U(0.001, 0.5)$ and per-variant $\beta = h^2_{SNP} \sim N(0, 0.4/1000)$
- *null*. M_eff - 1,050 variants with no effect with $p \sim U(0.0001, 0.5)$

2. *Reference sample*. This set includes 4,000 individuals (N_{ref}) with proportion of cases (P) = 0.5.

- sample normalized genotype matrix $\mathbf{X}$ with M_eff columns reflecting the number of variants and N_{ref} rows.
- calculate genetic values as $G = \mathbf{X}\beta$
- sample residual values from $E \sim N(0, 1 - h^2)$
- calculate phenotypic values on the liability scale as $P = G + E$
- define cases as those exceeding the liability threshold T defined as $\Phi^{-1}(T)$.

3. Estimate SNP effects (b) in reference sample using PLINK.

4. *Target sample*. This set includes 2,000 individuals (N_{target}) with proportion of cases (P) = 0.5.

- same procedure as simulating reference sample.

5. Predict genetic values $\hat{g}$.

- $\hat{g} = \mathbf{X}\hat{b}$

6. Calculate AUC based genetic prediction.

1.2 Simulation code

```
sim_ref = function(M, N, K, Pcase,
                   pi_1, pi_2, pi_3,
```

```
                  h2_1, h2_2, h2_3,
                  p_1_min, p_2_min, p_3_min,
                  p_1_max, p_2_max, p_3_max){

  h2   = h2_1 + h2_2 + h2_3

  # simulate effect sizes and allele-frequencies
  b_1  = rnorm(pi_1, 0, sqrt(h2_1/pi_1))
  p_1  = runif(pi_1, p_1_min, p_1_max)
  b_2  = rnorm(pi_2, 0, sqrt(h2_2/pi_2))
  p_2  = runif(pi_2, p_2_min, p_2_max)
  b_3  = rnorm(pi_3, 0, sqrt(h2_3/pi_3))
  p_3  = runif(pi_3, p_3_min, p_3_max)

  b    = c(b_1, b_2, b_3)
  p    = c(p_1, p_2, p_3)

  # define number of cases and controls
  Ncase = N * Pcase
  Ncon  = N * (1 - Pcase)

  # define liabiliy threshold
  T = -qnorm(K)

  # simulate genotypes for reference set
  n_case_batch    = c()
  n_control_batch = c()
  ref             = data.frame()

  while (sum(n_case_batch) < Ncase || sum(n_control_batch) < Ncon) {
    Nb  = 10000
    X   = parallel::mcmapply(function(i) rbinom(Nb, 2, p[i]), 1:M)
    G   = scale(X) %*% b
    E   = rnorm(Nb, 0, sqrt(1 - h2))
    P   = G + E
    Pcc = rep(0, Nb) ; Pcc[P >= T] = 1
    d   = data.frame(P, G, E, X, Pcc)
    cases = d[d$Pcc == 1, ]
    cons  = d[d$Pcc == 0, ]
    cons  = cons[sample(nrow(cons), round(nrow(cases) / (Pcase / (1-Pcase)))), ]
    ref   = rbind(ref, cases, cons)
    n_case_batch = c(n_case_batch, nrow(cases))
    n_control_batch = c(n_control_batch, nrow(cons))
    print(paste0("Number of cases ", sum(n_case_batch)))
    print(paste0("Number of controls ", sum(n_control_batch)))
  }
  return(list("ref" = ref, "b" = b, "p" = p))
}
```

```
est_b = function(ref){

  x     = as.matrix(ref[,4:(M+3)])
  y     = as.vector(ref[,M+4])
  b_hat = Rfast::logistic_only(x, y, tol = 1e-09, b_values=T)[3,]
```

```
  b_na  = is.na(b_hat)

  if(sum(b_na) > 0) {
    cat("WARNING",
        sum(b_na),
        "variants with infinite effect estimates, set to null-effect\n")
    b_hat[b_na] = 0
  }

  return(b_hat)

}
```

```
sim_target = function(M, N, Pcase, h2, K, b, p){

  Ncase = N * Pcase
  Ncon  = N * (1 - Pcase)

  # define liabilitty threshold
  T = -qnorm(K)

  # simulate genotypes for reference set
  n_case_batch    = c()
  n_control_batch = c()
  target          = data.frame()

  while (sum(n_case_batch) < Ncase || sum(n_control_batch) < Ncon) {
    Nb  = 10000
    X   = parallel::mcmapply(function(i) rbinom(Nb, 2, p[i]), 1:M)
    G   = scale(X) %*% b
    E   = rnorm(Nb, 0, sqrt(1 - h2))
    P   = G + E
    Pcc = rep(0, Nb) ; Pcc[P >= T] = 1
    d   = data.frame(P, G, E, X, Pcc)
    cases  = d[d$Pcc == 1, ]
    cons  = d[d$Pcc == 0, ]
    cons = cons[sample(nrow(cons), round(nrow(cases) / (Pcase / (1-Pcase)))), ]
    target = rbind(target, cases, cons)
    n_case_batch <- c(n_case_batch, nrow(cases))
    n_control_batch <- c(n_control_batch, nrow(cons))
    print(paste0("Number of cases ", sum(n_case_batch)))
    print(paste0("Number of controls ", sum(n_control_batch)))
  }

  return(target)

}
```

```
est_g = function(target, b_hat){

  X_target = as.matrix(target[, 4:(length(b_hat)+3)])
  g_hat = X_target %*% b_hat

  return(as.vector(g_hat))
```

```
}
```

```
get_auc = function(Pcc, g_hat){

  roc_obj = pROC::roc(Pcc, g_hat)
  auc     = pROC::auc(roc_obj)

  return(auc)

}
```

```
get_r2 = function(P, g_hat){

  P_std      = scale(P)
  g_hat_std  = scale(g_hat)
  cov_P_ghat = cov(P_std, g_hat_std)
  r2l        = (cov_P_ghat / sqrt(var(P_std) * var(g_hat_std)))^2

  return(list("cov"=cov_P_ghat, "r2l"=r2l))

}
```

```
run_simulation = function(M, K,
                          N_ref, Pcase_ref,
                          N_target, Pcase_target,
                          pi_1, pi_2, pi_3,
                          h2_1, h2_2, h2_3,
                          p_1_min, p_2_min, p_3_min,
                          p_1_max, p_2_max, p_3_max){

  h2  = h2_1 + h2_2 + h2_3

  cat("1. simulate reference set\n")
  ref = sim_ref(M=M, N=N_ref, K=K, Pcase=Pcase_ref,
                pi_1=pi_1, pi_2=pi_2, pi_3=pi_3,
                h2_1=h2_1, h2_2=h2_2, h2_3=h2_3,
                p_1_min=p_1_min, p_2_min=p_2_min, p_3_min=p_3_min,
                p_1_max=p_1_max, p_2_max=p_2_max, p_3_max=p_3_min)

  cat("2. estimate SNP effects\n")
  b_hat = est_b(ref$ref)

  cat("3. simulate target set\n")
  target = sim_target(M=M, N=N_target, K=K, Pcase=Pcase_target,
                      h2=h2, b=ref$b, p=ref$p)

  cat("4. estimate genetic values for target set\n")
  g_hat_target = est_g(target=target, b_hat=b_hat)

  cat("5. calculate AUC.\n")
  auc = get_auc(Pcc=target$Pcc, g_hat=g_hat_target)

  cat("6. calculate covariance and r2 on the standardized liability scale.\n")
  cov_r2l    = get_r2(P=target$P, g_hat=g_hat_target)
```

```
  cov_P_ghat = cov_r2l$cov
  r2l        = cov_r2l$r2l

  return(list("auc"=auc, "cov_P_ghat"=cov_P_ghat, "r2l"=r2l))

}
```

1.3 Run simulations

Run the simulation for close relatives (M_{eff} = 10,000)

```
M = 10000
K = 0.1                 # disease life-time risk
N_ref     = 4000        # number of individuals in reference set
Pcase_ref = 0.5         # proportion of cases in reference set
N_target     = 2000     # number of individuals in target set
Pcase_target = 0.5      # proportion of individuals in target set
pi_1 = 50               # number of variants in large-effect bin
pi_2 = 1000             # number of variants in small-effect bin
pi_3 = M - pi_1 - pi_2  # (remaining) number of variants in null-effect bin
h2_1 = 0.1              # total h2 of variants in large-effect bin
h2_2 = 0.4              # total h2 of variants in small-effect bin
h2_3 = 0                # total h2 of variants in null-effect bin (0 by definition)
p_1_min = 0.001         # lower allele frequency limit of large-effect bin
p_1_max = 0.01          # upper allele frequency limit of large-effect bin
p_2_min = 0.01          # lower allele frequency limit of small-effect bin
p_2_max = 0.5           # upper allele frequency limit of small-effect bin
p_3_min = 0.001         # lower allele frequency limit of null-effect bin
p_3_max = 0.5           # upper allele frequency limit of null-effect bin

perf = run_simulation(M=M, K=K, N_ref=N_ref, Pcase_ref=Pcase_ref,
                      N_target=N_target, Pcase_target=Pcase_target,
                      pi_1=pi_1, pi_2=pi_2, pi_3=pi_3,
                      h2_1=h2_1, h2_2=h2_2, h2_3=h2_3,
                      p_1_min=p_1_min, p_2_min=p_2_min, p_3_min=p_3_min,
                      p_1_max=p_1_max, p_2_max=p_2_max, p_3_max=p_3_max)
```

```
## 1. simulate reference set
## [1] "Number of cases 1003"
## [1] "Number of controls 1003"
## [1] "Number of cases 1955"
## [1] "Number of controls 1955"
## [1] "Number of cases 2978"
## [1] "Number of controls 2978"
## 2. estimate SNP effects
## 3. simulate target set
## [1] "Number of cases 984"
## [1] "Number of controls 984"
## [1] "Number of cases 1939"
## [1] "Number of controls 1939"
## 4. estimate genetic values for target set
## 5. calculate AUC.
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
## 6. calculate covariance and r2 on the standardized liability scale.
```

```
cat("AUC =", round(perf$auc, 3), "\n")
```

```
## AUC = 0.657
```

```
cat("cov_P_ghat =", perf$cov_P_ghat, "\n")
```

```
## cov_P_ghat = 0.1951384
```

```
cat("r2_l =", perf$r2l, "\n")
```

```
## r2_l = 0.038079
```

Run the simulation for unrelated individuals of European ancestry (M_{eff} = 50,000)

```
M = 50000
K = 0.1                 # disease life-time risk
N_ref     = 4000        # number of individuals in reference set
Pcase_ref = 0.5         # proportion of cases in reference set
N_target     = 2000     # number of individuals in target set
Pcase_target = 0.5      # proportion of individuals in target set
pi_1 = 50               # number of variants in large-effect bin
pi_2 = 1000             # number of variants in small-effect bin
pi_3 = M - pi_1 - pi_2  # (remaining) number of variants in null-effect bin
h2_1 = 0.1              # total h2 of variants in large-effect bin
h2_2 = 0.4              # total h2 of variants in small-effect bin
h2_3 = 0                # total h2 of variants in null-effect bin (0 by definition)
p_1_min = 0.001         # lower allele frequency limit of large-effect bin
p_1_max = 0.01          # upper allele frequency limit of large-effect bin
p_2_min = 0.01          # lower allele frequency limit of small-effect bin
p_2_max = 0.5           # upper allele frequency limit of small-effect bin
p_3_min = 0.001         # lower allele frequency limit of null-effect bin
p_3_max = 0.5           # upper allele frequency limit of null-effect bin

perf = run_simulation(M=M, K=K, N_ref=N_ref, Pcase_ref=Pcase_ref,
                      N_target=N_target, Pcase_target=Pcase_target,
                      pi_1=pi_1, pi_2=pi_2, pi_3=pi_3,
                      h2_1=h2_1, h2_2=h2_2, h2_3=h2_3,
                      p_1_min=p_1_min, p_2_min=p_2_min, p_3_min=p_3_min,
                      p_1_max=p_1_max, p_2_max=p_2_max, p_3_max=p_3_max)
```

```
## 1. simulate reference set
## [1] "Number of cases 1029"
## [1] "Number of controls 1029"
## [1] "Number of cases 2084"
## [1] "Number of controls 2084"
## 2. estimate SNP effects
## WARNING 11 variants with infinite effect estimates, set to null-effect
## 3. simulate target set
## [1] "Number of cases 965"
## [1] "Number of controls 965"
## [1] "Number of cases 1969"
## [1] "Number of controls 1969"
## 4. estimate genetic values for target set
## 5. calculate AUC.
```

```
## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

## 6. calculate covariance and r2 on the standardized liability scale.
cat("AUC =", round(perf$auc, 3), "\n")
```

```
## AUC = 0.516
cat("cov_P_ghat =", perf$cov_P_ghat, "\n")
```

```
## cov_P_ghat = 0.02936111
cat("r2_l =", perf$r2l, "\n")
```

```
## r2_l = 0.0008620748
```

2 Simulation II. Power using predicted liability vs. case-control design

2.1 Simulation scheme

- The input parameters for this simulation are:
- covariance between (standardized) genetic profile score and phenotype on liability scale (obtained from simulation 1)
- Odds ratio for the measure (for example cortical thickness on MRI) on binary case-control scale (OR_{PA})
- Proportion of cases in sample from which this OR is obtained (K_P)
- Proportion of individuals with the abnormal test result (thresholded) (K_A)
- Sample size (N)
- Type-I error rate (α)
- Calculate correlation between measure and phenotype on liability scale using the tetrachoric correlation.
- Define variance-covariance matrix for (V):
- Phenotypic values on liability scale (P)
- Standardized genetic profile scores ($\hat{G}$)
- Standardized normal measure (A)

$$
\begin{aligned}
&\begin{pmatrix} P & \hat{G} & A \end{pmatrix} \sim N(M, V) \\
&M = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix} \\
&V = \begin{pmatrix} 1 & cov(P, \hat{G}) & rt_{PA} \\ & 1 & rt_{PA} \cdot cov(P, \hat{G}) \\ & & 1 \end{pmatrix}
\end{aligned}
$$

- Compare power to detect an associationi between measure and outcome on case-control and predicted liability scale.

2.2 Simulation Code

From the OR_{PA}, K_P and K_A create the 2x2 table. Source: Stack Exchange.

```
get_cont_tbl = function(Kx, Ky, ORxy, eps=1e-15){

  # this function derives the 2x2 contingency table based on
  # the prevalence and odds ratio of two binary traits
  # the entries of the 2x2 table is defined as:
  # b = Kx - a
  # c = Ky - a
```

```
  # d = 1 + a - Ky - Kx
  # a = a^2 + a*(Kx + Ky - 2* a) + ORxy*(Kx*Ky - (Kx + Ky)*a + a^2)
  # a can be solved from the quadratic:
  # (ORxy - 1)*a^2 + [(Kx+Ky)*(1-ORxy)-1]*a + ORxy*Kx*Ky

  a  = ORxy - 1
  b  = (Kx + Ky) * (1 - ORxy) - 1
  c_ = ORxy * Kx * Ky

  if (abs(a) < eps) {
    z = -c_ / b
  } else {
    d = b^2 - 4 * a * c_
    if (d < eps^2) {
      s = 0
    } else {
      s = c(-1, 1)
    }
    z = (-b + s*sqrt(max(0,d))) / (2*a)
  }
  y = vapply(z, function(a) zapsmall(matrix(c(a, Ky-a, Kx-a, 1+a-Ky-Kx), 2, 2,)),
             matrix(0.0, 2, 2))
  i = apply(y, 3, function(u) all(u >= 0))
  return(y[,,i])
}
```

Calculate the tetrachoric correlation $r_{tc,PA}$:

```
get_rtc = function(tbl){

  rtc = polycor::polychor(tbl)

  return(rtc)

}
```

Define the variance-covariance matrix (V) for phenotype on liability scale (P), genetic profile score ($\hat{G}$) and measure A:

```
define_V = function(cov_P_ghat, rtc_PA){

  V = matrix(c(1, cov_P_ghat, rtc_PA,
               cov_P_ghat, 1, rtc_PA* cov_P_ghat,
               rtc_PA, rtc_PA*cov_P_ghat, 1), nrow=3)

  return(V)

}
```

Simulate a study of size (N) with P, $\hat{G}$ and A and test associations:

```
sim_study = function(N, V, K_P, K_A){

  T_P = -qnorm(K_P)
  T_A = -qnorm(K_A)
```

```
  S = MASS::mvrnorm(N, c(0,0,0), V)
  colnames(S) = c("P", "G_hat", "A")
  Pcc = rep(0, N) ; Pcc[S[,"P"] >= T_P] = 1

  # association on case-control scale
  m1 = glm(Pcc ~ S[,"A"], family=binomial)
  p1 = coef(summary(m1))[2,4]

  # association on predicted liability scale
  m2 = lm(S[,"G_hat"] ~ S[,"A"])
  p2 = coef(summary(m2))[2,4]

  return(c(p1, p2))
}
```

Power simulations:

```
get_power = function(cov_P_ghat,
                     OR_PA, K_P, K_A,
                     R, a, N){

  # R = number of repeated simulations to obtain power
  # a = alpha
  # N = sample size

  tbl = get_cont_tbl(Kx=K_A, Ky=K_P, ORxy=OR_PA, eps=1e-15)

  rtc = get_rtc(tbl=tbl)

  V   = define_V(cov_P_ghat=cov_P_ghat, rtc_PA=rtc)

  ps  = parallel::mcmapply(function(i) sim_study(N=N, V=V, K_P=K_P, K_A=K_A), 1:R)

  power_cc = sum(ps[1,] < a) / R
  power_gl = sum(ps[2,] < a) / R


  return(list("power_cc" = power_cc, "power_gl" = power_gl))

}
```

2.3 Run simulations

Will have to pick more realistic parameterssim_study(N=N, V=V, K_P=K_P, K_A=K_A)

```
cov_P_ghat = 0.25   # covariance obtained from simulation I.
OR_PA      = 1.3    # OR binary measure on phenotype on case-control scale
K_P        = 1/300  # prevalence of disease in sample (here population)
K_A        = 0.1    # proportion of "positive" measure A
N          = 10000
R          = 1000
a          = 0.05

powers = get_power(cov_P_ghat=cov_P_ghat, OR_PA=OR_PA,
                   K_P=K_P, K_A=K_A, R=R, a=a, N=N)
```

```r
cat("Power for case-control design: ", powers$power_cc, "\n")
```

```
## Power for case-control design:  0.126
```

```r
cat("Power for predicted genetic liability design: ", powers$power_gl, "\n")
```

```
## Power for predicted genetic liability design:  0.204
```