In [ ]:
```
#Author: 坚定的唯物主义鼠鼠

# 用脚本来做SNP_scan, 拒绝使用phenoscanner

# 1:47 2023/5/9

library(TwoSampleMR)

a1="ieu-a-2"
a2="ieu-a-7"
```

In [ ]:
```
exposure_data = extract_instruments(a1,clump=TRUE,r2=0.01,kb=1e7)  # 根据暴露提取SNP
```

In [ ]:
```
exposure_data$SNP   #这里能够列出所有与暴露相关的SNP

length(exposure_data$SNP)  #一共有87个
```

'rs543874' · 'rs11165643' · 'rs977747' · 'rs657452' · 'rs7531118' · 'rs17391694' · 'rs7551507' · 'rs7550711' · 'rs10182181' · 'rs6713510' · 'rs4671328' · 'rs11126666' · 'rs2890652' · 'rs13021737' · 'rs1016287' · 'rs1460676' · 'rs7599312' · 'rs11692326' · 'rs1516725' · 'rs2365389' · 'rs3849570' · 'rs13078960' · 'rs6804842' · 'rs16851483' · 'rs11727676' · 'rs13107325' · 'rs13130484' · 'rs17001654' · 'rs7715256' · 'rs2112347' · 'rs6457796' · 'rs9374842' · 'rs943005' · 'rs2465031' · 'rs13201877' · 'rs13191362' · 'rs3800229' · 'rs1167827' · 'rs2060604' · 'rs10733682' · 'rs6477694' · 'rs2183825' · 'rs1928295' · 'rs4740619' · 'rs7903146' · 'rs7899106' · 'rs12220375' · 'rs17094222' · 'rs3817334' · 'rs12286929' · 'rs10840100' · 'rs11030104' · 'rs2176598' · 'rs11057405' · 'rs7138803' · 'rs9579083' · 'rs12429545' · 'rs1441264' · 'rs9540493' · 'rs7144011' · 'rs10132280' · 'rs12885454' · 'rs13329567' · 'rs3736485' · 'rs7164727' · 'rs749767' · 'rs2080454' · 'rs758747' · 'rs879620' · 'rs3888190' · 'rs12446632' · 'rs1421085' · 'rs11074446' · 'rs9914578' · 'rs12940622' · 'rs1000940' · 'rs7243357' · 'rs17066842' · 'rs891389' · 'rs6567160' · 'rs2075650' · 'rs11672660' · 'rs3810291' · 'rs14810' · 'rs17724992' · 'rs6091540' · 'rs2836754'

87

In [ ]:
```
exposure_data$pval.exposure  # 这里是暴露的SNP的p值

print("--------------------")

print(sort(exposure_data$pval.exposure))  #我们将p值进行升序排列, 这样就可以看到最大的p值是 4.97794e-08 , 即所有的SNP都与暴露显著相关的
```

2.28718e-40 · 1.43384e-13 · 2.18198e-08 · 2.12324e-13 · 1.88018e-28 · 5.68604e-09 · 1.11892e-15 · 5.05941e-14 · 8.07049e-26 · 1.97401e-08 · 2.77402e-09 · 1.32401e-09 · 1.242e-08 · 5.43876e-54 · 4.35512e-12 · 4.97794e-08 · 4.73042e-11 · 4.80397e-08 · 1.39412e-24 · 1.34602e-10 · 1.93299e-08 · 1.42298e-14 · 8.01604e-10 · 1.84999e-10 · 6.24698e-09 · 1.0639e-12 · 8.01124e-41 · 5.03095e-09 · 8.85095e-09 · 1.9602e-17 · 2.53501e-10 · 7.19797e-09 · 4.52376e-31 · 2.711e-08 · 4.28499e-08 · 1.09199e-09 · 4.95005e-08 · 1.97501e-10 · 9.46019e-12 · 2.45499e-10 · 1.70498e-08 · 2.2228e-14 · 4.31797e-10 · 6.35594e-09 · 1.10306e-12 · 1.269e-08 · 1.758e-09 · 2.18625e-11 · 1.16788e-17 · 5.44252e-13 · 6.66653e-12 · 6.65733e-30 · 3.46897e-08 · 1.22e-08 · 5.11446e-26 · 1.426e-10 · 3.1521e-13 · 2.95903e-08 · 3.95203e-09 · 6.04505e-15 · 1.40088e-11 · 9.09285e-11 · 1.52616e-18 · 4.52397e-08 · 3.91498e-09 · 8.17523e-11 · 8.60399e-09 · 1.50598e-10 · 3.93904e-10 · 3.45382e-25 · 1.80884e-19 · 2.1677e-158 · 6.07198e-10 · 2.072e-08 · 3.63597e-10 · 1.81201e-08 · 9.13503e-09 · 8.98669e-14 · 1.617e-08 · 6.68344e-59 · 3.209e-09 · 7.91043e-19 · 6.35331e-16 · 1.923e-08 · 7.787e-09 · 2.13801e-08 · 1.60498e-08

```
[1] "-------------------"
 [1] 2.16770e-158 6.68344e-59  5.43876e-54  8.01124e-41  2.28718e-40
 [6]  4.52376e-31  6.65733e-30  1.88018e-28  5.11446e-26  8.07049e-26
[11]  3.45382e-25  1.39412e-24  1.80884e-19  7.91043e-19  1.52616e-18
[16]  1.16788e-17  1.96020e-17  6.35331e-16  1.11892e-15  6.04505e-15
[21]  1.42298e-14  2.22280e-14  5.05941e-14  8.98669e-14  1.43384e-13
[26]  2.12324e-13  3.15210e-13  5.44252e-13  1.06390e-12  1.10306e-12
[31]  4.35512e-12  6.66653e-12  9.46019e-12  1.40088e-11  2.18625e-11
[36]  4.73042e-11  8.17523e-11  9.09285e-11  1.34602e-10  1.42600e-10
[41]  1.50598e-10  1.84999e-10  1.97501e-10  2.45499e-10  2.53501e-10
[46]  3.63597e-10  3.93904e-10  4.31797e-10  6.07198e-10  8.01604e-10
[51]  1.09199e-09  1.32401e-09  1.75800e-09  2.77402e-09  3.20900e-09
[56]  3.91498e-09  3.95203e-09  5.03095e-09  5.68604e-09  6.24698e-09
[61]  6.35594e-09  7.19797e-09  7.78700e-09  8.60399e-09  8.85095e-09
[66]  9.13503e-09  1.22000e-08  1.24200e-08  1.26900e-08  1.60498e-08
[71]  1.61700e-08  1.70498e-08  1.81201e-08  1.92300e-08  1.93299e-08
[76]  1.97401e-08  2.07200e-08  2.13801e-08  2.18198e-08  2.71100e-08
[81]  2.95903e-08  3.46897e-08  4.28499e-08  4.52397e-08  4.80397e-08
[86]  4.95005e-08  4.97794e-08
```

In [ ]:

```
#---------------------------------------------------------------
```

In [ ]:

```
outcome_data = extract_outcome_data(snps=exposure_data$SNP,outcomes=a2)  # 在暴露的SNP中提取结局SNP（就是从中筛一遍）    （参数snps就是暴露的
```

```
Extracting data for 87 SNP(s) from 1 GWAS(s)
```

In [ ]:

```
outcome_data$SNP  #这里能够列出所有与结局相关的SNP

length(outcome_data$SNP)  #一共有87个
```

'rs977747' · 'rs7550711' · 'rs9374842' · 'rs4740619' · 'rs2112347' · 'rs3817334' · 'rs10840100' · 'rs11030104' · 'rs13021737' · 'rs11057405' · 'rs7144011' · 'rs12429545' · 'rs3888190' · 'rs7164727' · 'rs2080454' · 'rs14810' · 'rs12940622' · 'rs6091540' · 'rs1460676' · 'rs17391694' · 'rs7531118' · 'rs17001654' · 'rs13107325' · 'rs13130484' · 'rs6804842' · 'rs1167827' · 'rs7243357' · 'rs6567160' · 'rs17724992' · 'rs2075650' · 'rs3800229' · 'rs13191362' · 'rs2060604' · 'rs2465031' · 'rs7899106' · 'rs12885454' · 'rs3736485' · 'rs2836754' · 'rs7599312' · 'rs2365389' · 'rs13078960' · 'rs16851483' · 'rs10132280' · 'rs9540493' · 'rs1928295' · 'rs17094222' · 'rs7715256' · 'rs17066842' · 'rs12446632' · 'rs1421085' · 'rs3810291' · 'rs13201877' · 'rs7551507' · 'rs943005' · 'rs6457796' · 'rs7138803' · 'rs13329567' · 'rs758747' · 'rs11126666' · 'rs657452' · 'rs6713510' · 'rs4671328' · 'rs11727676' · 'rs11692326' · 'rs7903146' · 'rs12220375' · 'rs879620' · 'rs11074446' · 'rs1000940' · 'rs9579083' · 'rs6477694' · 'rs10733682' · 'rs891389' · 'rs749767' · 'rs1441264' · 'rs543874' · 'rs10182181' · 'rs2890652' · 'rs3849570' · 'rs11165643' · 'rs2183825' · 'rs1516725' · 'rs1016287' · 'rs12286929' · 'rs2176598' · 'rs9914578' · 'rs11672660'

87

In [ ]:

```
outcome_data$pval.outcome  # 这里是结局的SNP的p值

print("--------------------")
print( sort(outcome_data$pval.outcome) ) #我们将p值进行升序排列，这样就可以看到最小的p值是 4.55995e-08
```

0.148375 · 0.113631 · 0.201203 · 0.479131 · 0.544364 · 0.970312 · 0.125286 · 0.0117704 · 0.00292772 · 0.473392 · 0.0759189 · 0.411649 · 0.0578163 · 0.0271356 · 0.148963 · 0.814436 · 0.0852531 · 0.816488 · 0.756986 · 0.102851 · 0.0693745 · 0.166657 · 0.765341 · 0.000915103 · 0.283329 · 0.238789 · 0.730937 · 4.55995e-08 · 0.173778 · 1.61002e-06 · 0.693753 · 0.0162062 · 0.0355992 · 0.417679 · 0.770715 · 0.13374 · 0.868067 · 0.417181 · 0.0172556 · 0.11893 · 0.112632 · 0.381063 · 0.253816 · 0.538962 · 0.468643 · 0.759408 · 0.805256 · 0.0773019 · 0.547849 · 0.00206509 · 0.0177048 · 0.0373672 · 0.831668 · 0.129728 · 0.00152268 · 0.388225 · 0.18562 · 0.581946 · 0.020879 · 0.413747 · 7.10003e-05 · 0.67886 · 0.071241 · 0.0285089 · 0.001448 · 1.43999e-07 · 0.762578 · 0.0338563 · 0.139538 · 0.0554383 · 0.516634 · 0.633525 · 0.013128 · 0.556351 · 0.597832 · 0.517029 · 0.0486486 · 0.76784 · 0.00333472 · 0.579105 · 0.0238276 · 0.552521 · 0.182778 · 0.848298 · 0.0191686 · 0.0002009 · 0.0019162

```
[1] "--------------------"
 [1] 4.55995e-08 1.43999e-07 1.61002e-06 7.10003e-05 2.00900e-04 9.15103e-04
 [7] 1.44800e-03 1.52268e-03 1.91620e-03 2.06509e-03 2.92772e-03 3.33472e-03
[13] 1.17704e-02 1.31280e-02 1.62062e-02 1.72556e-02 1.77048e-02 1.91686e-02
[19] 2.08790e-02 2.38276e-02 2.71356e-02 2.85089e-02 3.38563e-02 3.55992e-02
[25] 3.73672e-02 4.86486e-02 5.54383e-02 5.78163e-02 6.93745e-02 7.12410e-02
[31] 7.59189e-02 7.73019e-02 8.52531e-02 1.02851e-01 1.12632e-01 1.13631e-01
[37] 1.18930e-01 1.25286e-01 1.29728e-01 1.33740e-01 1.39538e-01 1.48375e-01
[43] 1.48963e-01 1.66657e-01 1.73778e-01 1.82778e-01 1.85620e-01 2.01203e-01
[49] 2.38789e-01 2.53816e-01 2.83329e-01 3.81063e-01 3.88225e-01 4.11649e-01
[55] 4.13747e-01 4.17181e-01 4.17679e-01 4.68643e-01 4.73392e-01 4.79131e-01
[61] 5.16634e-01 5.17029e-01 5.38962e-01 5.44364e-01 5.47849e-01 5.52521e-01
[67] 5.56351e-01 5.79105e-01 5.81946e-01 5.97832e-01 6.33525e-01 6.78860e-01
[73] 6.93753e-01 7.30937e-01 7.56986e-01 7.59408e-01 7.62578e-01 7.65341e-01
[79] 7.67840e-01 7.70715e-01 8.05256e-01 8.14436e-01 8.16488e-01 8.31668e-01
[85] 8.48298e-01 8.68067e-01 9.70312e-01
```

In [ ]:

```
#然后我们将结局的SNP的p值小于0.05的SNP提取出来（即与结局相关的SNP），这些就是我们要去除的SNP
```

```r
SNP_should_OUT=outcome_data$SNP[ which(outcome_data$pval.outcome< 0.05 ) ]  #这里是结局的SNP的p值小于0.05的SNP,一共有26个

length(SNP_should_OUT)  # 26个

print(SNP_should_OUT)  #这些就是我们要去除的SNP
```

```
26
 [1] "rs11030104" "rs13021737" "rs7164727"  "rs13130484" "rs6567160"
 [6] "rs2075650"  "rs13191362" "rs2060604"  "rs7599312"  "rs1421085"
[11] "rs3810291"  "rs13201877" "rs6457796"  "rs11126666" "rs6713510"
[16] "rs11692326" "rs7903146"  "rs12220375" "rs11074446" "rs891389"
[21] "rs10182181" "rs3849570"  "rs2183825"  "rs2176598"  "rs9914578"
[26] "rs11672660"
```

总结： 这个方法是利用了TwoSampleMR的计算缓存，缓存中有与暴露或结局相关的SNP及其pval。利用这个缓存，我们就能直接计算出SNP分别与暴露和结局的pval，并进行一个直接筛选。就不需要通过手动的方式去用phenoscanner筛选了。

省流：用脚本来做SNP_scan, 拒绝使用phenoscanner

反思：目前还不清楚SNP与结局变量的pval应该为多少才算是显著。 但这个筛选的脚本能够提供极大的便利，帮助我们剔除SNP，减少计算量。