Appendix C: Implementations in R and SAS

In this chapter, the used R and SAS codes are given.

C.1 R Code: Data Generation for Simulation Study

The following R codes describe the data generation for the external data sets and the genetic association study data sets for the simulation study of the six European countries sample for Study Design 1, 2 and 3. The data generation of the Swiss sample for Study Design 1 was done analogously.

Simulation of the six European countries sample

```
## 1.) Read in POPRES genotype data set and subject
   identification per country
nrun=100
set.seed(13022018)
IT_UK_FR_DE_PRT_ES_geno=read.table("POPRES_IT_UK_FR_DE_PRT_ES_
   final_R.raw", header=T)
IT_SUBJID=read.table("IT_SUBJID.txt", header=F)
UK_SUBJID=read.table("UK_SUBJID.txt", header=F)
FR_SUBJID=read.table("FR_SUBJID.txt", header=F)
DE_SUBJID=read.table("DE_SUBJID.txt", header=F)
PRT_SUBJID=read.table("PRT_SUBJID.txt", header=F)
```

```
ES_SUBJID=read.table("ES_SUBJID.txt", header=F)
Italian_complete<-IT_SUBJID[,1]
British_complete<-UK_SUBJID[,1]
French_complete<-FR_SUBJID[,1]
German_complete<-DE_SUBJID[,1]
Portuguese_complete<-PRT_SUBJID[,1]
Spanish_complete<-ES_SUBJID[,1]
Italian<-Italian_complete[which(Italian_complete %in% IT_UK_FR_
   DE_PRT_ES_geno$FID)]
British<-British_complete[which(British_complete %in% IT_UK_FR_
   DE_PRT_ES_geno$FID)]
French<-French_complete[which(French_complete %in% IT_UK_FR_DE_
   PRT_ES_geno$FID)]
German<-German_complete[which(German_complete %in% IT_UK_FR_DE_
   PRT_ES_geno$FID)]
Portuguese<-Portuguese_complete[which(Portuguese_complete %in%
   IT_UK_FR_DE_PRT_ES_geno$FID)]
Spanish<-Spanish_complete[which(Spanish_complete %in% IT_UK_FR_
   DE_PRT_ES_geno$FID)]
together<-c(Italian,British,French,German,Portuguese,Spanish)
IT_UK_FR_DE_PRT_ES_geno_order<-IT_UK_FR_DE_PRT_ES_geno[match(
   together, IT_UK_FR_DE_PRT_ES_geno$FID),]
IT_UK_FR_DE_PRT_ES_geno_SNP<-IT_UK_FR_DE_PRT_ES_geno_order[,7:
   ncol(IT_UK_FR_DE_PRT_ES_geno_order)]
rownames(IT_UK_FR_DE_PRT_ES_geno_SNP)<-c(1:nrow(IT_UK_FR_DE_PRT
   _ES_geno_SNP))
IT_ID<-c(1:length(Italian))
UK_ID<-c((length(Italian)+1):(length(Italian)+length(British)))
FR_ID<-c((length(Italian)+length(British)+1):(length(Italian)+
   length(British)+length(French)))
DE_ID<-c((length(Italian)+length(British)+length(French)+1):(
   length(Italian)+length(British)+length(French)+length(German)
   ))
PRT_ID<-c((length(Italian)+length(British)+length(French)+
   length(German)+1):(length(Italian)+length(British)+length(
   French)+length(German)+length(Portuguese)))
ES_ID<-c((length(Italian)+length(British)+length(French)+length
   (German)+length(Portuguese)+1):(length(Italian)+length(
   British)+length(French)+length(German)+length(Portuguese)+
   length(Spanish)))
n_TOTAL<-252
n_percountry<-42
n_cases_TOTAL<-126
case<-c(rep(1,n_cases_TOTAL),rep(0,n_cases_TOTAL))

## 2.) Create external data sets
```

```
training_IT<-replicate(nrun,sample(IT_ID,n_percountry, replace=
   F))
training_UK<-replicate(nrun,sample(UK_ID,n_percountry, replace=
   F))
training_FR<-replicate(nrun,sample(FR_ID,n_percountry, replace=
   F))
training_DE<-replicate(nrun,sample(DE_ID,n_percountry, replace=
   F))
training_PRT<-replicate(nrun,sample(PRT_ID,n_percountry,
   replace=F))
training_ES<-replicate(nrun,sample(ES_ID,n_percountry, replace=
   F))
training_SNP_complete<-list(NA)
for(i in 1:nrun) {
training_SNP_complete[[i]]<-data.frame(IT_UK_FR_DE_PRT_ES_geno_
   SNP[c(training_IT[,i],training_UK[,i],training_FR[,i],
   training_DE[,i],training_PRT[,i],training_ES[,i]),],
   stringsAsFactors = F)}

## 3.) Create genetic association study data sets
test_IT<-apply(training_IT,2, function(x) {sample(setdiff(IT_ID
   , x),n_percountry, replace=F)})
test_UK<-apply(training_UK,2, function(x) {sample(setdiff(UK_ID
   , x),n_percountry, replace=F)})
test_FR<-apply(training_FR,2, function(x) {sample(setdiff(FR_ID
   , x),n_percountry, replace=F)})
test_DE<-apply(training_DE,2, function(x) {sample(setdiff(DE_ID
   , x),n_percountry, replace=F)})
test_PRT<-apply(training_PRT,2, function(x) {sample(setdiff(PRT
   _ID, x),n_percountry, replace=F)})
test_ES<-apply(training_ES,2, function(x) {sample(setdiff(ES_ID
   , x),n_percountry, replace=F)})
test_SNP_complete<-list(NA)
for(i in 1:nrun) {test_SNP_complete[[i]]<-data.frame(IT_UK_FR_
   DE_PRT_ES_geno_SNP[c(test_IT[,i],test_UK[,i],test_FR[,i],test
   _DE[,i],test_PRT[,i],test_ES[,i]),],stringsAsFactors = F)}

## 4.) Final MAF check
MAF_training_SNP_complete<-list(NA)
MAF_training_SNP_complete_1<-list(NA)
for (i in 1:nrun){MAF_training_SNP_complete[[i]]<-(apply(
   training_SNP_complete[[i]]==1,2,sum,na.rm=TRUE)+2*apply(
   training_SNP_complete[[i]]==2,2,sum,na.rm=TRUE))/(nrow(
   training_SNP_complete[[i]])*2)
MAF_training_SNP_complete_1[[i]]<-names(which(MAF_training_SNP_
   complete[[i]]<0.01))}
```

```
MAF_training_SNP_complete_1_list<-unique(unlist(MAF_training_
   SNP_complete_1))
MAF_test_SNP_complete<-list(NA)
MAF_test_SNP_complete_1<-list(NA)
for (i in 1:nrun){MAF_test_SNP_complete[[i]]<-(apply(test_SNP_
   complete[[i]]==1,2,sum,na.rm=TRUE)+2*apply(test_SNP_complete
   [[i]]==2,2,sum,na.rm=TRUE))/(nrow(test_SNP_complete[[i]])*2)
MAF_test_SNP_complete_1[[i]]<-names(which(MAF_test_SNP_complete
   [[i]]<0.01))}
MAF_test_SNP_complete_1_list<-unique(unlist(MAF_test_SNP_
   complete_1))
MAF_training_test_SNP_complete_1_list<-unique(c(MAF_training_
   SNP_complete_1_list,MAF_test_SNP_complete_1_list))
IT_UK_FR_DE_PRT_ES_geno_SNP_MAF<-IT_UK_FR_DE_PRT_ES_geno_SNP[,!
   colnames(IT_UK_FR_DE_PRT_ES_geno_SNP) %in% MAF_training_test_
   SNP_complete_1_list]

## 5.) Simulate external data sets for Study Design 1
n_cases_percountry_reference<-21
country_reference<-c(rep("Italy",n_cases_percountry_reference),
   rep("UnitedKingdom",n_cases_percountry_reference),rep("France
   ",n_cases_percountry_reference),rep("Germany",n_cases_
   percountry_reference),rep("Portugal",n_cases_percountry_
   reference),rep("Spain",n_cases_percountry_reference),rep("
   Italy",n_cases_percountry_reference),rep("UnitedKingdom",n_
   cases_percountry_reference),rep("France",n_cases_percountry_
   reference),rep("Germany",n_cases_percountry_reference),rep("
   Portugal",n_cases_percountry_reference),rep("Spain",n_cases_
   percountry_reference))
training_cases_IT_reference<-training_IT[1:n_cases_percountry_
   reference,]
training_controls_IT_reference<-training_IT[(n_cases_percountry
   _reference+1):n_percountry,]
training_cases_UK_reference<-training_UK[1:n_cases_percountry_
   reference,]
training_controls_UK_reference<-training_UK[(n_cases_percountry
   _reference+1):n_percountry,]
training_cases_FR_reference<-training_FR[1:n_cases_percountry_
   reference,]
training_controls_FR_reference<-training_FR[(n_cases_percountry
   _reference+1):n_percountry,]
training_cases_DE_reference<-training_DE[1:n_cases_percountry_
   reference,]
training_controls_DE_reference<-training_DE[(n_cases_percountry
   _reference+1):n_percountry,]
training_cases_PRT_reference<-training_PRT[1:n_cases_percountry
   _reference,]
```

```
training_controls_PRT_reference<-training_PRT[(n_cases_
   percountry_reference+1):n_percountry,]
training_cases_ES_reference<-training_ES[1:n_cases_percountry_
   reference,]
training_controls_ES_reference<-training_ES[(n_cases_percountry
   _reference+1):n_percountry,]
training_data_reference<-list(NA)
training_data_reference_new<-list(NA)
training_data_reference_new_new<-list(NA)
for(i in 1:nrun) {training_data_reference[[i]]<-data.frame(case
   ,country_reference,IT_UK_FR_DE_PRT_ES_geno_SNP_MAF[c(training
   _cases_IT_reference[,i],training_cases_UK_reference[,i],
   training_cases_FR_reference[,i],training_cases_DE_reference[,
   i],training_cases_PRT_reference[,i],training_cases_ES_
   reference[,i],training_controls_IT_reference[,i],training_
   controls_UK_reference[,i],training_controls_FR_reference[,i],
   training_controls_DE_reference[,i],training_controls_PRT_
   reference[,i],training_controls_ES_reference[,i]),],
   stringsAsFactors = F)
training_data_reference_new[[i]]<-data.frame("row_num"=rownames
   (training_data_reference[[i]]),training_data_reference[[i]],
   stringsAsFactors = F)
training_data_reference_new_new[[i]]<-training_data_reference_
   new[[i]][order(as.numeric(training_data_reference_new[[i
   ]][,1])),c(2:ncol(training_data_reference_new[[i]]))]}
training_save_reference <- for(i in 1:nrun) {write.table(data.
   frame(training_data_reference_new_new[[i]], stringsAsFactors
   = F), file=paste("training_cases_controls_IT_UK_FR_DE_PRT_ES_
   simulated_reference_",i,".txt", sep=""), row.names=F, col.
   names=T)}
training_data_frame_reference<-do.call(rbind, training_data_
   reference_new_new)
id_reference<-rep(c(1:nrow(training_data_reference_new_new
   [[1]])),nrun)
run_reference<-rep(1:nrun, each=nrow(training_data_reference_
   new_new[[1]]))
training_data_frame_complete_reference<-data.frame(run_
   reference,id_reference,training_data_frame_reference,
   stringsAsFactors = F)
training_save_csv_reference <- write.table(training_data_frame_
   complete_reference, file="training_cases_controls_IT_UK_FR_DE
   _PRT_ES_simulated_reference_100.csv", row.names=FALSE, na="",
   col.names=FALSE, sep=",")

## 6.) Simulate genetic association study data sets for Study
   Design 1
```

```
test_cases_IT_reference<-test_IT[1:n_cases_percountry_reference
   ,]
test_controls_IT_reference<-test_IT[(n_cases_percountry_
   reference+1):n_percountry,]
test_cases_UK_reference<-test_UK[1:n_cases_percountry_reference
   ,]
test_controls_UK_reference<-test_UK[(n_cases_percountry_
   reference+1):n_percountry,]
test_cases_FR_reference<-test_FR[1:n_cases_percountry_reference
   ,]
test_controls_FR_reference<-test_FR[(n_cases_percountry_
   reference+1):n_percountry,]
test_cases_DE_reference<-test_DE[1:n_cases_percountry_reference
   ,]
test_controls_DE_reference<-test_DE[(n_cases_percountry_
   reference+1):n_percountry,]
test_cases_PRT_reference<-test_PRT[1:n_cases_percountry_
   reference,]
test_controls_PRT_reference<-test_PRT[(n_cases_percountry_
   reference+1):n_percountry,]
test_cases_ES_reference<-test_ES[1:n_cases_percountry_reference
   ,]
test_controls_ES_reference<-test_ES[(n_cases_percountry_
   reference+1):n_percountry,]
test_data_reference<-list(NA)
test_data_reference_new<-list(NA)
test_data_reference_new_new<-list(NA)
for(i in 1:nrun) {test_data_reference[[i]]<-data.frame(case,
   country_reference,IT_UK_FR_DE_PRT_ES_geno_SNP_MAF[c(test_
   cases_IT_reference[,i],test_cases_UK_reference[,i],test_cases
   _FR_reference[,i],test_cases_DE_reference[,i],test_cases_PRT_
   reference[,i],test_cases_ES_reference[,i],test_controls_IT_
   reference[,i],test_controls_UK_reference[,i],test_controls_FR
   _reference[,i],test_controls_DE_reference[,i],test_controls_
   PRT_reference[,i],test_controls_ES_reference[,i]),],
   stringsAsFactors = F)
test_data_reference_new[[i]]<-data.frame("row_num"=rownames(
   test_data_reference[[i]]),test_data_reference[[i]],
   stringsAsFactors = F)
test_data_reference_new_new[[i]]<-test_data_reference_new[[i]][
   order(as.numeric(test_data_reference_new[[i]][,1])),c(2:ncol(
   test_data_reference_new[[i]]))]}
test_save_reference <- for(i in 1:nrun) {write.table(data.frame
   (test_data_reference_new_new[[i]], stringsAsFactors = F),
   file=paste("test_cases_controls_IT_UK_FR_DE_PRT_ES_simulated_
   reference_",i,".txt", sep=""), row.names=F, col.names=T)}
```

```
test_data_frame_reference<-do.call(rbind, test_data_reference_
   new_new)
id_reference<-rep(c(1:nrow(test_data_reference_new_new[[1]])),
   nrun)
run_reference<-rep(1:nrun, each=nrow(test_data_reference_new_
   new[[1]]))
test_data_frame_complete_reference<-data.frame(run_reference,id
   _reference,test_data_frame_reference, stringsAsFactors = F)
test_save_csv_reference <- write.table(test_data_frame_complete
   _reference, file="test_cases_controls_IT_UK_FR_DE_PRT_ES_
   simulated_reference_100.csv", row.names=FALSE, na="",col.
   names=FALSE, sep=",")

## 7.) Simulate external data sets for Study Design 3
n_cases_percountry_most_extreme<-42
country_most_extreme<-c(rep("Italy",n_cases_percountry_most_
   extreme),rep("Spain",n_cases_percountry_most_extreme),rep("
   Portugal",n_cases_percountry_most_extreme),rep("UnitedKingdom
   ",n_cases_percountry_most_extreme),rep("Germany",n_cases_
   percountry_most_extreme),rep("France",n_cases_percountry_most
   _extreme))
training_cases_IT_most_extreme<-training_IT[1:n_cases_
   percountry_most_extreme,]
training_cases_ES_most_extreme<-training_ES[1:n_cases_
   percountry_most_extreme,]
training_cases_PRT_most_extreme<-training_PRT[1:n_cases_
   percountry_most_extreme,]
training_controls_UK_most_extreme<-training_UK[1:n_cases_
   percountry_most_extreme,]
training_controls_DE_most_extreme<-training_DE[1:n_cases_
   percountry_most_extreme,]
training_controls_FR_most_extreme<-training_FR[1:n_cases_
   percountry_most_extreme,]
training_data_most_extreme<-list(NA)
training_data_most_extreme_new<-list(NA)
training_data_most_extreme_new_new<-list(NA)
for(i in 1:nrun) {training_data_most_extreme[[i]]<-data.frame(
   case,country_most_extreme,IT_UK_FR_DE_PRT_ES_geno_SNP_MAF[c(
   training_cases_IT_most_extreme[,i],training_cases_ES_most_
   extreme[,i],training_cases_PRT_most_extreme[,i],training_
   controls_UK_most_extreme[,i],training_controls_DE_most_
   extreme[,i],training_controls_FR_most_extreme[,i]),],
   stringsAsFactors = F)
training_data_most_extreme_new[[i]]<-data.frame("row_num"=
   rownames(training_data_most_extreme[[i]]),training_data_most_
   extreme[[i]],stringsAsFactors = F)
```

```
training_data_most_extreme_new_new[[i]]<-training_data_most_
   extreme_new[[i]][order(as.numeric(training_data_most_extreme_
   new[[i]][,1])),c(2:ncol(training_data_most_extreme_new[[i]]))
   ]}
training_save_most_extreme <- for(i in 1:nrun) {write.table(
   data.frame(training_data_most_extreme_new_new[[i]],
   stringsAsFactors = F), file=paste("training_cases_controls_IT
   _UK_FR_DE_PRT_ES_simulated_most_extreme_",i,".txt", sep=""),
   row.names=F, col.names=T)}
training_data_frame_most_extreme<-do.call(rbind, training_data_
   most_extreme_new_new)
id_most_extreme<-rep(c(1:nrow(training_data_most_extreme_new_
   new[[1]])),nrun)
run_most_extreme<-rep(1:nrun, each=nrow(training_data_most_
   extreme_new_new[[1]]))
training_data_frame_complete_most_extreme<-data.frame(run_most_
   extreme,id_most_extreme,training_data_frame_most_extreme,
   stringsAsFactors = F)
training_save_csv_most_extreme <- write.table(training_data_
   frame_complete_most_extreme, file="training_cases_controls_IT
   _UK_FR_DE_PRT_ES_simulated_most_extreme_100.csv", row.names=
   FALSE, na="",col.names=FALSE, sep=",")

## 8.) Simulate genetic association study data sets for Study
   Design 3
test_cases_IT_most_extreme<-test_IT[1:n_cases_percountry_most_
   extreme,]
test_cases_ES_most_extreme<-test_ES[1:n_cases_percountry_most_
   extreme,]
test_cases_PRT_most_extreme<-test_PRT[1:n_cases_percountry_most
   _extreme,]
test_controls_UK_most_extreme<-test_UK[1:n_cases_percountry_
   most_extreme,]
test_controls_DE_most_extreme<-test_DE[1:n_cases_percountry_
   most_extreme,]
test_controls_FR_most_extreme<-test_FR[1:n_cases_percountry_
   most_extreme,]
test_data_most_extreme<-list(NA)
test_data_most_extreme_new<-list(NA)
test_data_most_extreme_new_new<-list(NA)
for(i in 1:nrun) {test_data_most_extreme[[i]]<-data.frame(case,
   country_most_extreme,IT_UK_FR_DE_PRT_ES_geno_SNP_MAF[c(test_
   cases_IT_most_extreme[,i],test_cases_ES_most_extreme[,i],test
   _cases_PRT_most_extreme[,i],test_controls_UK_most_extreme[,i
   ],test_controls_DE_most_extreme[,i],test_controls_FR_most_
   extreme[,i]),],stringsAsFactors = F)
```

```
test_data_most_extreme_new[[i]]<-data.frame("row_num"=rownames(
   test_data_most_extreme[[i]]),test_data_most_extreme[[i]],
   stringsAsFactors = F)
test_data_most_extreme_new_new[[i]]<-test_data_most_extreme_new
   [[i]][order(as.numeric(test_data_most_extreme_new[[i]][,1])),
   c(2:ncol(test_data_most_extreme_new[[i]]))]}
test_save_most_extreme <- for(i in 1:nrun) {write.table(data.
   frame(test_data_most_extreme_new_new[[i]], stringsAsFactors =
    F), file=paste("test_cases_controls_IT_UK_FR_DE_PRT_ES_
   simulated_most_extreme_",i,".txt", sep=""), row.names=F, col.
   names=T)}
test_data_frame_most_extreme<-do.call(rbind, test_data_most_
   extreme_new_new)
id_most_extreme<-rep(c(1:nrow(test_data_most_extreme_new_new
   [[1]])),nrun)
run_most_extreme<-rep(1:nrun, each=nrow(test_data_most_extreme_
   new_new[[1]]))
test_data_frame_complete_most_extreme<-data.frame(run_most_
   extreme,id_most_extreme,test_data_frame_most_extreme,
   stringsAsFactors = F)
test_save_csv_most_extreme <- write.table(test_data_frame_
   complete_most_extreme, file="test_cases_controls_IT_UK_FR_DE_
   PRT_ES_simulated_most_extreme_100.csv", row.names=FALSE, na="
   ",col.names=FALSE, sep=",")

## 9.) Simulate external data sets for Study Design 2
n_cases_Italy_extreme<-23
n_controls_Italy_extreme<-19
n_cases_UnitedKingdom_extreme<-19
n_controls_UnitedKingdom_extreme<-23
n_cases_France_extreme<-19
n_controls_France_extreme<-23
n_cases_Germany_extreme<-19
n_controls_Germany_extreme<-23
n_cases_Portugal_extreme<-23
n_controls_Portugal_extreme<-19
n_cases_Spain_extreme<-23
n_controls_Spain_extreme<-19
country_extreme<-c(rep("Italy",n_cases_Italy_extreme),rep("
   UnitedKingdom",n_cases_UnitedKingdom_extreme),rep("France",n_
   cases_France_extreme),rep("Germany",n_cases_Germany_extreme),
   rep("Portugal",n_cases_Portugal_extreme),rep("Spain",n_cases_
   Spain_extreme),rep("Italy",n_controls_Italy_extreme),rep("
   UnitedKingdom",n_controls_UnitedKingdom_extreme),rep("France"
   ,n_controls_France_extreme),rep("Germany",n_controls_Germany_
   extreme),rep("Portugal",n_controls_Portugal_extreme),rep("
   Spain",n_controls_Spain_extreme))
```

```
training_cases_IT_extreme<-training_IT[1:n_cases_Italy_extreme
   ,]
training_controls_IT_extreme<-training_IT[(n_cases_Italy_
   extreme+1):n_percountry,]
training_cases_UK_extreme<-training_UK[1:n_cases_UnitedKingdom_
   extreme,]
training_controls_UK_extreme<-training_UK[(n_cases_
   UnitedKingdom_extreme+1):n_percountry,]
training_cases_FR_extreme<-training_FR[1:n_cases_France_extreme
   ,]
training_controls_FR_extreme<-training_FR[(n_cases_France_
   extreme+1):n_percountry,]
training_cases_DE_extreme<-training_DE[1:n_cases_Germany_
   extreme,]
training_controls_DE_extreme<-training_DE[(n_cases_Germany_
   extreme+1):n_percountry,]
training_cases_PRT_extreme<-training_PRT[1:n_cases_Portugal_
   extreme,]
training_controls_PRT_extreme<-training_PRT[(n_cases_Portugal_
   extreme+1):n_percountry,]
training_cases_ES_extreme<-training_ES[1:n_cases_Spain_extreme
   ,]
training_controls_ES_extreme<-training_ES[(n_cases_Spain_
   extreme+1):n_percountry,]
training_data_extreme<-list(NA)
training_data_extreme_new<-list(NA)
training_data_extreme_new_new<-list(NA)
for(i in 1:nrun) {training_data_extreme[[i]]<-data.frame(case,
   country_extreme,IT_UK_FR_DE_PRT_ES_geno_SNP_MAF[c(training_
   cases_IT_extreme[,i],training_cases_UK_extreme[,i],training_
   cases_FR_extreme[,i],training_cases_DE_extreme[,i],training_
   cases_PRT_extreme[,i],training_cases_ES_extreme[,i],training_
   controls_IT_extreme[,i],training_controls_UK_extreme[,i],
   training_controls_FR_extreme[,i],training_controls_DE_extreme
   [,i],training_controls_PRT_extreme[,i],training_controls_ES_
   extreme[,i]),],stringsAsFactors = F)
training_data_extreme_new[[i]]<-data.frame("row_num"=rownames(
   training_data_extreme[[i]]),training_data_extreme[[i]],
   stringsAsFactors = F)
training_data_extreme_new_new[[i]]<-training_data_extreme_new[[
   i]][order(as.numeric(training_data_extreme_new[[i]][,1])),c
   (2:ncol(training_data_extreme_new[[i]]))]}
training_save_extreme <- for(i in 1:nrun) {write.table(data.
   frame(training_data_extreme_new_new[[i]], stringsAsFactors =
   F), file=paste("training_cases_controls_IT_UK_FR_DE_PRT_ES_
   simulated_extreme_",i,".txt", sep=""), row.names=F, col.names
   =T)}
```

```
training_data_frame_extreme<-do.call(rbind, training_data_
   extreme_new_new)
id_extreme<-rep(c(1:nrow(training_data_extreme_new_new[[1]])),
   nrun)
run_extreme<-rep(1:nrun, each=nrow(training_data_extreme_new_
   new[[1]]))
training_data_frame_complete_extreme<-data.frame(run_extreme,id
   _extreme,training_data_frame_extreme, stringsAsFactors = F)
training_save_csv_extreme <- write.table(training_data_frame_
   complete_extreme, file="training_cases_controls_IT_UK_FR_DE_
   PRT_ES_simulated_extreme_100.csv", row.names=FALSE, na="",col
   .names=FALSE, sep=",")

## 10.) Simulate genetic association study data sets for Study
   Design 2
test_cases_IT_extreme<-test_IT[1:n_cases_Italy_extreme,]
test_controls_IT_extreme<-test_IT[(n_cases_Italy_extreme+1):n_
   percountry,]
test_cases_UK_extreme<-test_UK[1:n_cases_UnitedKingdom_extreme
   ,]
test_controls_UK_extreme<-test_UK[(n_cases_UnitedKingdom_
   extreme+1):n_percountry,]
test_cases_FR_extreme<-test_FR[1:n_cases_France_extreme,]
test_controls_FR_extreme<-test_FR[(n_cases_France_extreme+1):n_
   percountry,]
test_cases_DE_extreme<-test_DE[1:n_cases_Germany_extreme,]
test_controls_DE_extreme<-test_DE[(n_cases_Germany_extreme+1):n
   _percountry,]
test_cases_PRT_extreme<-test_PRT[1:n_cases_Portugal_extreme,]
test_controls_PRT_extreme<-test_PRT[(n_cases_Portugal_extreme
   +1):n_percountry,]
test_cases_ES_extreme<-test_ES[1:n_cases_Spain_extreme,]
test_controls_ES_extreme<-test_ES[(n_cases_Spain_extreme+1):n_
   percountry,]
test_data_extreme<-list(NA)
test_data_extreme_new<-list(NA)
test_data_extreme_new_new<-list(NA)
for(i in 1:nrun) {test_data_extreme[[i]]<-data.frame(case,
   country_extreme,IT_UK_FR_DE_PRT_ES_geno_SNP_MAF[c(test_cases_
   IT_extreme[,i],test_cases_UK_extreme[,i],test_cases_FR_
   extreme[,i],test_cases_DE_extreme[,i],test_cases_PRT_extreme
   [,i],test_cases_ES_extreme[,i],test_controls_IT_extreme[,i],
   test_controls_UK_extreme[,i],test_controls_FR_extreme[,i],
   test_controls_DE_extreme[,i],test_controls_PRT_extreme[,i],
   test_controls_ES_extreme[,i]),],stringsAsFactors = F)
```

```
test_data_extreme_new[[i]]<-data.frame("row_num"=rownames(test_
   data_extreme[[i]]),test_data_extreme[[i]],stringsAsFactors =
   F)
test_data_extreme_new_new[[i]]<-test_data_extreme_new[[i]][
   order(as.numeric(test_data_extreme_new[[i]][,1])),c(2:ncol(
   test_data_extreme_new[[i]]))]}
test_save_extreme <- for(i in 1:nrun) {write.table(data.frame(
   test_data_extreme_new_new[[i]], stringsAsFactors = F), file=
   paste("test_cases_controls_IT_UK_FR_DE_PRT_ES_simulated_
   extreme_",i,".txt", sep=""), row.names=F, col.names=T)}
test_data_frame_extreme<-do.call(rbind, test_data_extreme_new_
   new)
id_extreme<-rep(c(1:nrow(test_data_extreme_new_new[[1]])),nrun)
run_extreme<-rep(1:nrun, each=nrow(test_data_extreme_new_new
   [[1]]))
test_data_frame_complete_extreme<-data.frame(run_extreme,id_
   extreme,test_data_frame_extreme, stringsAsFactors = F)
test_save_csv_extreme <- write.table(test_data_frame_complete_
   extreme, file="test_cases_controls_IT_UK_FR_DE_PRT_ES_
   simulated_extreme_100.csv", row.names=FALSE, na="",col.names=
   FALSE, sep=",")
```

C.2 R Code: Principal Component Analysis

The following R codes describe the PCA based on the external data sets necessary for the identification of F_{ST}- and PC-AIMs and the PCA based on the genetic association study data sets for population stratification adjustment with PCs in the simulated study designs. The R Codes for the six European countries sample are shown. The PCA based on the Swiss sample was conducted analogously and is therefore not displayed.

Classical Principal Component Analysis of the six European countries sample

```
## 1.) Read in simulated external data sets and genetic
   association study data sets
nrun=100
set.seed(13022018)
training_case_files = list.files(pattern = 'training_cases_
   controls_IT_UK_FR_DE_PRT_ES_simulated_reference_.*.txt')
library(gtools)
training_case_files <- mixedsort(training_case_files)
training_case_control_data = lapply(training_case_files, read.
   table, header=T)
```

```
test_case_files = list.files(pattern = 'test_cases_controls_IT_
   UK_FR_DE_PRT_ES_simulated_reference_.*.txt')
library(gtools)
test_case_files <- mixedsort(test_case_files)
test_case_control_data = lapply(test_case_files, read.table,
   header=T)
training_PCA_data<-list(NA)
test_PCA_data<-list(NA)
for (i in 1:nrun){ training_PCA_data[[i]]<-training_case_
   control_data[[i]][,c(3:ncol(training_case_control_data[[i]]))
   ]
test_PCA_data[[i]]<-test_case_control_data[[i]][,c(3:ncol(test_
   case_control_data[[i]]))]}

## 2.) EIGENSTRAT code for classical PCA conduction
##EIGENSTRAT
eigenstrat<-function(geno){
  nMis<-rowSums(is.na(geno))
  geno<-geno[nMis==0,]
  avg<-rowSums(geno)/ncol(geno)
  keep<-avg!=0&avg!=2
  avg<-avg[keep]
  geno<-geno[keep,]
  snp<-nrow(geno)
  ind<-ncol(geno)

  freq<-avg/2
  M <- (geno-avg)/sqrt(freq*(1-freq))
  X<-t(M)%*%as.matrix(M)
  X<-X/(sum(diag(X))/(snp-1))
  E<-eigen(X)
  mu<-(sqrt(snp-1)+sqrt(ind))^2/snp
  sigma<-(sqrt(snp-1)+sqrt(ind))/snp*(1/sqrt(snp-1)+1/sqrt(ind)
     )^(1/3)
  E$TW<-(E$values[1]*ind/sum(E$values)-mu)/sigma
  E$mu<-mu
  E$sigma<-sigma
  class(E)<-"eigenstrat"
  E
  return(E$vectors[,1:10])}

## 3.) Classical PCA based on external data sets
training_eig_ind_PC1_PC10<-list(NA)
for(i in 1:nrun){training_eig_ind_PC1_PC10[[i]]<-eigenstrat(t(
   training_PCA_data[[i]]))}
PC_names<-c("PC1","PC2","PC3","PC4","PC5","PC6","PC7","PC8","
   PC9","PC10")
```

```
training_eig_ind_PC1_PC10_names<-list(NA)
for (i in 1:nrun){training_eig_ind_PC1_PC10_names[[i]]<-
   training_eig_ind_PC1_PC10[[i]]
colnames(training_eig_ind_PC1_PC10_names[[i]])<-PC_names}
for(i in 1:nrun) {write.table(data.frame(training_eig_ind_PC1_
   PC10_names[[i]], stringsAsFactors = F), file=paste("training_
   st_ind_PC1_PC10_simulated_reference_IT_UK_FR_DE_PRT_ES_run_",
   i,".txt", sep=""), row.names=F, col.names=F)}

## 4.) Classical PCA based on genetic association study data
   sets
test_test_eig_ind_PC1_PC10<-list(NA)
for(i in 1:nrun){test_test_eig_ind_PC1_PC10[[i]]<-eigenstrat(t(
   test_PCA_data[[i]]))}
PC_names<-c("PC1","PC2","PC3","PC4","PC5","PC6","PC7","PC8","
   PC9","PC10")
test_test_eig_ind_PC1_PC10_names<-list(NA)
for (i in 1:nrun){test_test_eig_ind_PC1_PC10_names[[i]]<-test_
   test_eig_ind_PC1_PC10[[i]]
colnames(test_test_eig_ind_PC1_PC10_names[[i]])<-PC_names}
for(i in 1:nrun) {write.table(data.frame(test_test_eig_ind_PC1_
   PC10_names[[i]], stringsAsFactors = F), file=paste("test_test
   _st_ind_PC1_PC10_simulated_reference_IT_UK_FR_DE_PRT_ES_run_"
   ,i,".txt", sep=""), row.names=F, col.names=F)}

## 5.) Save identified PCs based on genetic association study
   data sets with genetic association study data sets
test_test_snp_cases_controls_PC1_PC10<-list(NA)
for(i in 1:nrun){test_test_snp_cases_controls_PC1_PC10[[i]]<-
   data.frame(test_case_control_data[[i]],test_test_eig_ind_PC1_
   PC10_names[[i]], stringsAsFactors = F)}
test_test_case_control_PC_data_frame<-do.call(rbind, test_test_
   snp_cases_controls_PC1_PC10)
id<-rep(c(1:nrow(test_case_control_data[[1]])),nrun)
run<-rep(1:nrun, each=nrow(test_case_control_data[[1]]))
test_test_case_control_PC_data_frame_complete<-data.frame(run,
   id,test_test_case_control_PC_data_frame, stringsAsFactors = F
   )
test_test_case_control_PC_data_1<-test_test_case_control_PC_
   data_frame_complete[test_test_case_control_PC_data_frame_
   complete$run %in% c(1:10),]
# Data samples 2 to 10 were conducted analogously.
test_test_cases_controls_PC_save_1_csv <- write.table(test_test
   _case_control_PC_data_1, file="test_test_cases_controls_PC1_
   PC10_IT_UK_FR_DE_PRT_ES_simulated_reference_100_1.csv", row.
   names=FALSE, na="",col.names=FALSE, sep=",")
# Data samples 2 to 10 were saved analogously.
```

Robust Principal Component Analysis of the six European countries sample

```
## 1.) Read in simulated external data sets and genetic
   association study data sets
nrun=100
set.seed(13022018)
training_case_files = list.files(pattern = 'training_cases_
   controls_IT_UK_FR_DE_PRT_ES_simulated_reference_.*.txt')
library(gtools)
training_case_files <- mixedsort(training_case_files)
training_case_control_data = lapply(training_case_files, read.
   table, header=T)
test_case_files = list.files(pattern = 'test_cases_controls_IT_
   UK_FR_DE_PRT_ES_simulated_reference_.*.txt')
library(gtools)
test_case_files <- mixedsort(test_case_files)
test_case_control_data = lapply(test_case_files, read.table,
   header=T)
training_PCA_data<-list(NA)
test_PCA_data<-list(NA)
for (i in 1:nrun){training_PCA_data[[i]]<-training_case_control
   _data[[i]][,c(3:ncol(training_case_control_data[[i]]))]
test_PCA_data[[i]]<-test_case_control_data[[i]][,c(3:ncol(test_
   case_control_data[[i]]))]}

## 2.) R libraries for robust PCA conduction
library(robustbase)
library(rrcov)

## 3.) Robust PCA based on external data sets
training_ro_ind_PC1_PC10_names<-list(NA)
for(i in 1:nrun){training_ro_ind_PC1_PC10_names[[i]]<-PcaHubert
   (na.omit(t(training_PCA_data[[i]])))@loadings[,c(1:10)]}
for(i in 1:nrun) {write.table(data.frame(training_ro_ind_PC1_
   PC10_names[[i]], stringsAsFactors = F), file=paste("training_
   ro_ind_PC1_PC10_simulated_reference_IT_UK_FR_DE_PRT_ES_run_",
   i,".txt", sep=""), row.names=F, col.names=F)}

## 4.) Robust PCA based on genetic association study data sets
test_test_ro_ind_PC1_PC10_names<-list(NA)
for(i in 1:nrun){test_test_ro_ind_PC1_PC10_names[[i]]<-
   PcaHubert(na.omit(t(test_PCA_data[[i]])))@loadings[,c(1:10)]}
for(i in 1:nrun) {write.table(data.frame(test_test_ro_ind_PC1_
   PC10_names[[i]], stringsAsFactors = F), file=paste("test_test
```

```
   _ro_ind_PC1_PC10_simulated_reference_IT_UK_FR_DE_PRT_ES_run_"
   ,i,".txt", sep=""), row.names=F, col.names=F)}

## 5.) Save identified PCs based on genetic association study
   data sets with genetic association study data sets
   analogously to classical PCs
```

C.3 R Code: F-Statistic Ancestry-Informative Markers

The following R codes describe the identification of the classical F_{ST}-AIMs based on classical PC1 of the external data sets of the six European countries sample. The identification of the robust F_{ST}-AIMs as well as the identification of the classical and robust F_{ST}-AIMs based on the Swiss sample was conducted analogously and is therefore not displayed.

Classical F-Statistic Ancestry-Informative Markers of the six European countries sample

```
## 1.) Read in simulated external data sets and genetic
   association study data sets
nrun=100
set.seed(13022018)
training_case_files = list.files(pattern = 'training_cases_
   controls_IT_UK_FR_DE_PRT_ES_simulated_reference_.*.txt')
library(gtools)
training_case_files <- mixedsort(training_case_files)
training_case_control_data = lapply(training_case_files, read.
   table, header=T)
training_case_control_data_SNPsonly<-list(NA)
for (i in 1:nrun){training_case_control_data_SNPsonly[[i]]<-
   training_case_control_data[[i]][,c(3:ncol(training_case_
   control_data[[i]]))]}
test_case_files = list.files(pattern = 'test_cases_controls_IT_
   UK_FR_DE_PRT_ES_simulated_reference_.*.txt')
library(gtools)
test_case_files <- mixedsort(test_case_files)
test_case_control_data = lapply(test_case_files, read.table,
   header=T)

## 2.) Read in classical PCs based on simulated external data
   sets
st_PC_files = list.files(pattern = 'training_st_ind_PC1_PC10_
   simulated_reference_IT_UK_FR_DE_PRT_ES_run_.*.txt')
library(gtools)
st_PC_files <- mixedsort(st_PC_files)
```

```
st_PC_data = lapply(st_PC_files, read.table, header=F)

## 3.) Pick classical PC1 only
st_PC_data_PC1<-list(NA)
for (i in 1: nrun){st_PC_data_PC1[[i]] = st_PC_data[[i]][,1]}
library(rlang)
library(tibble)
library(hierfstat)

## 4.) Identify extreme individuals along classical PC1 and
   calculate the FST of each genetic variant
ex_5_l_st_PCA<-list(NA)
ex_5_r_st_PCA<-list(NA)
ex_case_control_data<-list(NA)
st_PCA<-list(NA)
st_PCA_1<-list(NA)
st_PCA_FST<-list(NA)
st_PCA_FST_sort<-list(NA)
st_PCA_FST_sort_all<-list(NA)
st_PCA_FST_sort_overview<-list(NA)
st_PCA_FST_top10<-list(NA)
for(i in 1:nrun)
{ex_5_l_st_PCA[[i]]<-which(st_PC_data_PC1[[i]]<quantile(st_PC_
   data_PC1[[i]], c(0.05)))
ex_5_r_st_PCA[[i]]<-which(st_PC_data_PC1[[i]]>quantile(st_PC_
   data_PC1[[i]], c(0.95)))
ex_case_control_data[[i]]<-training_case_control_data_SNPsonly
   [[i]][c(ex_5_l_st_PCA[[i]],ex_5_r_st_PCA[[i]]),]
st_PCA[[i]]<-data.frame(c(rep(1,length(ex_5_l_st_PCA[[i]])),
   rep(0,length(ex_5_r_st_PCA[[i]]))),ex_case_control_data[[i]],
    stringsAsFactors=FALSE)
st_PCA_FST[[i]]<-(wc(st_PCA[[i]])$per.loc)$FST
st_PCA_FST_sort[[i]]<-sort(st_PCA_FST[[i]], decreasing=T, na.
   last=T)
st_PCA_FST_sort_all[[i]]<-data.frame(st_PCA_FST_sort[[i]],
   stringsAsFactors=FALSE)
st_PCA_FST_sort_overview[[i]]<-data.frame(colnames(training_
   case_control_data_SNPsonly[[i]])[as.numeric(rownames(st_PCA_
   FST_sort_all[[i]]))],st_PCA_FST_sort_all[[i]][,1],
   stringsAsFactors=FALSE)
st_PCA_FST_top10[[i]]<-st_PCA_FST_sort_overview[[i]][c(1:10),]}
for(i in 1:nrun) {write.table(data.frame(st_PCA_FST_top10[[i]],
    stringsAsFactors = F), file=paste("training_st_ind_FSTAIM1_
   FSTAIM10_simulated_reference_IT_UK_FR_DE_PRT_ES_run_",i,".txt
   ", sep=""), row.names=F, col.names=F)}
```

```
## 5.) Save identified genetic variants with genetic
   association study data sets
test_snp_cases_controls_AIM1_AIM10<-list(NA)
for(i in 1:nrun){test_snp_cases_controls_AIM1_AIM10[[i]]<-data.
   frame(test_case_control_data[[i]],test_case_control_data[[i
   ]][,c(st_PCA_FST_top10[[i]][,1])], stringsAsFactors = F)
colnames(test_snp_cases_controls_AIM1_AIM10[[i]])[(ncol(test_
   case_control_data[[1]])+1):(ncol(test_case_control_data[[1]])
   +10)]<-c("AIM1","AIM2","AIM3","AIM4","AIM5","AIM6","AIM7","
   AIM8","AIM9","AIM10")}
test_case_control_AIM_data_frame<-do.call(rbind, test_snp_cases
   _controls_AIM1_AIM10)
id<-rep(c(1:nrow(test_case_control_data[[1]])),nrun)
run<-rep(1:nrun, each=nrow(test_case_control_data[[1]]))
test_case_control_AIM_data_frame_complete<-data.frame(run,id,
   test_case_control_AIM_data_frame, stringsAsFactors = F)
test_case_control_AIM_data_1<-test_case_control_AIM_data_frame_
   complete[test_case_control_AIM_data_frame_complete$run %in% c
   (1:10),]
# Data samples 2 to 10 were conducted analogously.
test_cases_controls_AIM_save_1_csv <- write.table(test_case_
   control_AIM_data_1, file="test_cases_controls_FSTAIM1_
   FSTAIM10_IT_UK_FR_DE_PRT_ES_simulated_reference_100_1.csv",
   row.names=FALSE, na="",col.names=FALSE, sep=",")
# Data samples 2 to 10 were saved analogously.
```

C.4 R Code: Principal Component Ancestry-Informative Markers

The following R codes describe the identification of the classical *PC*-AIMs based on classical PC1 of the external data sets of the six European countries sample. The identification of the robust *PC*-AIMs as well as the identification of the classical and robust *PC*-AIMs based on the Swiss sample was conducted analogously and is therefore not displayed.

Classical Principal Component Ancestry-Informative Markers of the six European countries sample

```
## 1.) Read in simulated external data sets and genetic
   association study data sets
nrun=100
set.seed(13022018)
training_case_files = list.files(pattern = 'training_cases_
   controls_IT_UK_FR_DE_PRT_ES_simulated_reference_.*.txt')
library(gtools)
```

```
training_case_files <- mixedsort(training_case_files)
training_case_control_data = lapply(training_case_files, read.
   table, header=T)
training_case_control_data_SNPsonly<-list(NA)
for (i in 1:nrun){training_case_control_data_SNPsonly[[i]]<-
   training_case_control_data[[i]][,c(3:ncol(training_case_
   control_data[[i]]))]}
test_case_files = list.files(pattern = 'test_cases_controls_IT_
   UK_FR_DE_PRT_ES_simulated_reference_.*.txt')
library(gtools)
test_case_files <- mixedsort(test_case_files)
test_case_control_data = lapply(test_case_files, read.table,
   header=T)

## 2.) Read in classical PCs based on simulated external data
   sets
st_PC_files = list.files(pattern = 'training_st_ind_PC1_PC10_
   simulated_reference_IT_UK_FR_DE_PRT_ES_run_.*.txt')
library(gtools)
st_PC_files <- mixedsort(st_PC_files)
st_PC_data = lapply(st_PC_files, read.table, header=F)

## 3.) Pick classical PC1 only
st_PC_data_names<-list(NA)
for (i in 1: nrun){st_PC_data_names[[i]] = st_PC_data[[i]][,1]}
snp_cases_controls_PC1_PC10<-list(NA)
for(i in 1:nrun){snp_cases_controls_PC1_PC10[[i]]<-data.frame(
   training_case_control_data[[i]],st_PC_data_names[[i]],
   stringsAsFactors = F)
colnames(snp_cases_controls_PC1_PC10[[i]])<-c(colnames(training
   _case_control_data[[i]]),"PC1")}

## 4.) Run ANOVA analysis
PCAIMS_st_PC_function_pval_complete_list<-matrix(NA,1,ncol(
   training_case_control_data_SNPsonly[[1]]))
PCAIMS_st_PC_function_pval_complete <- rep(list(PCAIMS_st_PC_
   function_pval_complete_list),nrun)
PCAIMS_st_PC_function_pval_complete_sort<-list(NA)
PCAIMS_st_PC_sort_all<-list(NA)
PCAIMS_st_PC_sort_overview<-list(NA)
PCAIMS_st_PC_top10<-list(NA)
for(i in 1:nrun)
{PCAIMS_st_PC_function<- function(mi){
PCAIMS_st_PC<-aov(PC1 ~ mi, data=snp_cases_controls_PC1_PC10[[i
   ]])
```

```
PCAIMS_st_PC_function_pval = tryCatch({summary(PCAIMS_st_PC)
   [[1]][["Pr(>F)"]][[1]]}, warning = function(w) {NA}, error =
   function(e) {NA})
if (is.na(PCAIMS_st_PC_function_pval)=="TRUE") {return_pval <-
   NA} else {return_pval <- PCAIMS_st_PC_function_pval}
return(c(as.numeric(return_pval)))}
PCAIMS_st_PC_function_pval_complete[[i]]<-apply(snp_cases_
   controls_PC1_PC10[[i]][,c(3:ncol(training_case_control_data[[
   i]]))], 2, PCAIMS_st_PC_function)
PCAIMS_st_PC_function_pval_complete_sort[[i]]<-sort(PCAIMS_st_
   PC_function_pval_complete[[i]], decreasing=F, na.last=T)
PCAIMS_st_PC_sort_all[[i]]<-data.frame(PCAIMS_st_PC_function_
   pval_complete_sort[[i]], stringsAsFactors=FALSE)
PCAIMS_st_PC_sort_overview[[i]]<-data.frame(rownames(PCAIMS_st_
   PC_sort_all[[i]]),PCAIMS_st_PC_sort_all[[i]][,1],
   stringsAsFactors=FALSE)
PCAIMS_st_PC_top10[[i]]<-PCAIMS_st_PC_sort_overview[[i]][c
   (1:10),]}
for(i in 1:nrun) {write.table(data.frame(PCAIMS_st_PC_top10[[i
   ]], stringsAsFactors = F), file=paste("training_st_ind_PCAIM1
   _PCAIM10_simulated_reference_IT_UK_FR_DE_PRT_ES_run_",i,".txt
   ", sep=""), row.names=F, col.names=F)}

## 5.) Save identified genetic variants with genetic
   association study data sets
test_snp_cases_controls_AIM1_AIM10<-list(NA)
for(i in 1:nrun)
{test_snp_cases_controls_AIM1_AIM10[[i]]<-data.frame(test_case_
   control_data[[i]],test_case_control_data[[i]][,c(PCAIMS_st_PC
   _top10[[i]][,1])], stringsAsFactors = F)
colnames(test_snp_cases_controls_AIM1_AIM10[[i]])[(ncol(test_
   case_control_data[[1]])+1):(ncol(test_case_control_data[[1]])
   +10)]<-c("AIM1","AIM2","AIM3","AIM4","AIM5","AIM6","AIM7","
   AIM8","AIM9","AIM10")}
test_case_control_AIM_data_frame<-do.call(rbind, test_snp_cases
   _controls_AIM1_AIM10)
id<-rep(c(1:nrow(test_case_control_data[[1]])),nrun)
run<-rep(1:nrun, each=nrow(test_case_control_data[[1]]))
test_case_control_AIM_data_frame_complete<-data.frame(run,id,
   test_case_control_AIM_data_frame, stringsAsFactors = F)
test_case_control_AIM_data_1<-test_case_control_AIM_data_frame_
   complete[test_case_control_AIM_data_frame_complete$run %in% c
   (1:10),]
# Data samples 2 to 10 were conducted analogously.
test_cases_controls_AIM_save_1_csv <- write.table(test_case_
   control_AIM_data_1, file="test_cases_controls_PCAIM1_PCAIM10_
```

```
   IT_UK_FR_DE_PRT_ES_simulated_reference_100_1.csv", row.names=
   FALSE, na="",col.names=FALSE, sep=",")
# Data samples 2 to 10 were saved analogously.
```

C.5 R Code: Weighted Loading Ancestry-Informative Markers

The following R codes describe the identification of the classical and robust WL-AIMs based on classical PC1 of the external data sets of the six European countries sample. The identification of the classical and robust WL-AIMs based on the Swiss sample was conducted analogously and is therefore not displayed.

Classical Weighted Loading Ancestry-Informative Markers of the six European countries sample

```
## 1.) Read in simulated external data sets and genetic
   association study data sets
nrun=100
set.seed(13022018)
training_case_files = list.files(pattern = 'training_cases_
   controls_IT_UK_FR_DE_PRT_ES_simulated_reference_.*.txt')
library(gtools)
training_case_files <- mixedsort(training_case_files)
training_case_control_data = lapply(training_case_files, read.
   table, header=T)
test_case_files = list.files(pattern = 'test_cases_controls_IT_
   UK_FR_DE_PRT_ES_simulated_reference_.*.txt')
library(gtools)
test_case_files <- mixedsort(test_case_files)
test_case_control_data = lapply(test_case_files, read.table,
   header=T)
training_PCA_data<-list(NA)
for (i in 1:nrun){training_PCA_data[[i]]<-training_case_control
   _data[[i]][,c(3:ncol(training_case_control_data[[i]]))]}
PCA_data<-list(NA)
for (i in 1:nrun){PCA_data[[i]]<-training_PCA_data[[i]][,
   complete.cases(t(training_PCA_data[[i]]))]}

## 2.) EIGENSTRAT code for classical PCA conduction
eigenstrat<-function(geno){
  nMis<-rowSums(is.na(geno))
  geno<-geno[nMis==0,]
  avg<-rowSums(geno)/ncol(geno)
  keep<-avg!=0&avg!=2
```

```
  avg<-avg[keep]
  geno<-geno[keep,]
  snp<-nrow(geno)
  ind<-ncol(geno)

  freq<-avg/2
  M <- (geno-avg)/sqrt(freq*(1-freq))
  X<-t(M)%*%as.matrix(M)
  X<-X/(sum(diag(X))/(snp-1))
  E<-eigen(X)
  mu<-(sqrt(snp-1)+sqrt(ind))^2/snp
  sigma<-(sqrt(snp-1)+sqrt(ind))/snp*(1/sqrt(snp-1)+1/sqrt(ind)
     )^(1/3)
  E$TW<-(E$values[1]*ind/sum(E$values)-mu)/sigma
  E$mu<-mu
  E$sigma<-sigma
  class(E)<-"eigenstrat"
  E
  return(E$vectors[,1:10])}

## 3.) Classical PCA on genetic variants based on external data
    sets
eig_snp_PC1_PC10<-list(NA)
for(i in 1:nrun){eig_snp_PC1_PC10[[i]]<-eigenstrat(t(na.omit(t(
   PCA_data[[i]]))))}
PC_names<-c("PC1","PC2","PC3","PC4","PC5","PC6","PC7","PC8","
   PC9","PC10")
eig_snp_PC1_PC10_names<-list(NA)
for (i in 1:nrun){eig_snp_PC1_PC10_names[[i]]<-eig_snp_PC1_PC10
   [[i]]
colnames(eig_snp_PC1_PC10_names[[i]])<-PC_names}

## 4.) Pick classical PC1 only
eig_snp_PC1<-list(NA)
for(i in 1:nrun){eig_snp_PC1[[i]]<-eig_snp_PC1_PC10_names[[i
   ]][,1]}

## 5.) Calculate weighted loading per genetic variant
eig_snp_PC1_names_aload<-list(NA)
eig_snp_PC1_names_aload <- lapply(eig_snp_PC1, function(x) {abs
   (x)})
max_loading<-list(NA)
eig_snp_PC1_weight<-list(NA)
for (i in 1:nrun){max_loading[[i]]<-which(eig_snp_PC1_names_
   aload[[i]]== max(eig_snp_PC1_names_aload[[i]], na.rm=T))[1]
eig_snp_PC1_weight[[i]]<-1/eig_snp_PC1_names_aload[[i]][max_
   loading[[i]]]*eig_snp_PC1_names_aload[[i]]}
```

```
eig_snp_PC1_weight_sort<-list(NA)
eig_snp_PC1_weight_sort_all<-list(NA)
eig_snp_PC1_weight_sort_overview<-list(NA)
eig_snp_PC1_weight_top10<-list(NA)
eig_snp_PC1_weight_top10_weight<-list(NA)
eig_snp_PC1_weight_sort_overview_test<-list(NA)
eig_snp_PC1_weight_top10_test<-list(NA)
for(i in 1:nrun)
{eig_snp_PC1_weight_sort[[i]]<-sort(eig_snp_PC1_weight[[i]],
   decreasing=T, na.last=NA, index.return=T)
eig_snp_PC1_weight_top10_weight[[i]]<-data.frame(colnames(PCA_
   data[[i]])[eig_snp_PC1_weight_sort[[i]]$ix][1:10],eig_snp_PC1
   _weight_sort[[i]]$x[1:10], stringsAsFactors = F)
eig_snp_PC1_weight_sort_overview[[i]]<-data.frame(training_case
   _control_data[[i]][,colnames(PCA_data[[i]])[eig_snp_PC1_
   weight_sort[[i]]$ix]], stringsAsFactors=FALSE)
eig_snp_PC1_weight_top10[[i]]<-eig_snp_PC1_weight_sort_overview
   [[i]][,c(1:10)]
eig_snp_PC1_weight_sort_overview_test[[i]]<-data.frame(test_
   case_control_data[[i]][,colnames(PCA_data[[i]])[eig_snp_PC1_
   weight_sort[[i]]$ix]], stringsAsFactors=FALSE)
eig_snp_PC1_weight_top10_test[[i]]<-eig_snp_PC1_weight_sort_
   overview_test[[i]][,c(1:10)]}
for(i in 1:nrun) {write.table(eig_snp_PC1_weight_top10_weight[[
   i]], file=paste("training_st_snp_PCAIMSNP1_PCAIMSNP10_
   simulated_reference_IT_UK_FR_DE_PRT_ES_run_",i,".txt", sep=""
   ), row.names=F, col.names=F)}

## 5.) Save identified genetic variants with genetic
   association study data sets
test_snp_cases_controls_AIM1_AIM10<-list(NA)
for(i in 1:nrun){test_snp_cases_controls_AIM1_AIM10[[i]]<-data.
   frame(test_case_control_data[[i]],eig_snp_PC1_weight_top10_
   test[[i]], stringsAsFactors = F)
colnames(test_snp_cases_controls_AIM1_AIM10[[i]])[(ncol(test_
   case_control_data[[1]])+1):(ncol(test_case_control_data[[1]])
   +10)]<-c("AIM1","AIM2","AIM3","AIM4","AIM5","AIM6","AIM7","
   AIM8","AIM9","AIM10")}
test_case_control_AIM_data_frame<-do.call(rbind, test_snp_cases
   _controls_AIM1_AIM10)
id<-rep(c(1:nrow(test_case_control_data[[1]])),nrun)
run<-rep(1:nrun, each=nrow(test_case_control_data[[1]]))
test_case_control_AIM_data_frame_complete<-data.frame(run,id,
   test_case_control_AIM_data_frame, stringsAsFactors = F)
test_case_control_AIM_data_1<-test_case_control_AIM_data_frame_
   complete[test_case_control_AIM_data_frame_complete$run %in% c
   (1:10),]
```

```
# Data samples were conducted analogously.
test_cases_controls_AIM_save_1_csv <- write.table(test_case_
   control_AIM_data_1, file="test_cases_controls_PCAIMSNP1_
   PCAIMSNP10_IT_UK_FR_DE_PRT_ES_simulated_reference_100_1.csv",
    row.names=FALSE, na="",col.names=FALSE, sep=",")
# Data samples were saved analogously.
```

Robust Weighted Loading Ancestry-Informative Markers of the six European countries sample

```
## 1.) Read in simulated external data sets and genetic
   association study data sets
nrun=100
set.seed(13022018)
training_case_files = list.files(pattern = 'training_cases_
   controls_IT_UK_FR_DE_PRT_ES_simulated_reference_.*.txt')
library(gtools)
training_case_files <- mixedsort(training_case_files)
training_case_control_data = lapply(training_case_files, read.
   table, header=T)
test_case_files = list.files(pattern = 'test_cases_controls_IT_
   UK_FR_DE_PRT_ES_simulated_reference_.*.txt')
library(gtools)
test_case_files <- mixedsort(test_case_files)
test_case_control_data = lapply(test_case_files, read.table,
   header=T)
training_PCA_data<-list(NA)
for (i in 1:nrun){training_PCA_data[[i]]<-training_case_control
   _data[[i]][,c(3:ncol(training_case_control_data[[i]]))]}
PCA_data<-list(NA)
for (i in 1:nrun){PCA_data[[i]]<-training_PCA_data[[i]][,
   complete.cases(t(training_PCA_data[[i]]))]}

## 2.) Robust PCA on genetic variants based on external data
   sets
library(robustbase)
library(rrcov)
ro_snp_PC1_PC10_names<-list(NA)
for(i in 1:nrun){ro_snp_PC1_PC10_names[[i]]<-PcaHubert(t(na.
   omit(t(PCA_data[[i]]))))@loadings[,c(1:10)]}

## 3.) Pick robust PC1 only
ro_snp_PC1<-list(NA)
for(i in 1:nrun){ro_snp_PC1[[i]]<-ro_snp_PC1_PC10_names[[i
   ]][,1]}

## 4.) Calculate weighted loading per genetic variant
```

```
ro_snp_PC1_names_aload<-list(NA)
ro_snp_PC1_names_aload <- lapply(ro_snp_PC1, function(x) {abs(x
   )})
max_loading<-list(NA)
ro_snp_PC1_weight<-list(NA)
for (i in 1:nrun){max_loading[[i]]<-which(ro_snp_PC1_names_
   aload[[i]]== max(ro_snp_PC1_names_aload[[i]], na.rm=T))[1]
ro_snp_PC1_weight[[i]]<-1/ro_snp_PC1_names_aload[[i]][max_
   loading[[i]]]*ro_snp_PC1_names_aload[[i]]}
ro_snp_PC1_weight_sort<-list(NA)
ro_snp_PC1_weight_sort_all<-list(NA)
ro_snp_PC1_weight_sort_overview<-list(NA)
ro_snp_PC1_weight_top10<-list(NA)
ro_snp_PC1_weight_top10_weight<-list(NA)
ro_snp_PC1_weight_sort_overview_test<-list(NA)
ro_snp_PC1_weight_top10_test<-list(NA)
for(i in 1:nrun){ro_snp_PC1_weight_sort[[i]]<-sort(ro_snp_PC1_
   weight[[i]], decreasing=T, na.last=NA, index.return=T)
ro_snp_PC1_weight_top10_weight[[i]]<-data.frame(colnames(PCA_
   data[[i]])[ro_snp_PC1_weight_sort[[i]]$ix][1:10],ro_snp_PC1_
   weight_sort[[i]]$x[1:10], stringsAsFactors = F)
ro_snp_PC1_weight_sort_overview[[i]]<-data.frame(training_case_
   control_data[[i]][,colnames(PCA_data[[i]])[ro_snp_PC1_weight_
   sort[[i]]$ix]], stringsAsFactors=FALSE)
ro_snp_PC1_weight_top10[[i]]<-ro_snp_PC1_weight_sort_overview[[
   i]][,c(1:10)]
ro_snp_PC1_weight_sort_overview_test[[i]]<-data.frame(test_case
   _control_data[[i]][,colnames(PCA_data[[i]])[ro_snp_PC1_weight
   _sort[[i]]$ix]], stringsAsFactors=FALSE)ro_snp_PC1_weight_
   top10_test[[i]]<-ro_snp_PC1_weight_sort_overview_test[[i]][,c
   (1:10)]}
for(i in 1:nrun) {write.table(ro_snp_PC1_weight_top10_weight[[i
   ]], file=paste("training_ro_snp_PCAIMSNP1_PCAIMSNP10_
   simulated_reference_IT_UK_FR_DE_PRT_ES_run_",i,".txt", sep=""
   ), row.names=F, col.names=F)}

## 5.) Save identified genetic variants with genetic
   association study data sets
test_snp_cases_controls_AIM1_AIM10<-list(NA)
for(i in 1:nrun){
test_snp_cases_controls_AIM1_AIM10[[i]]<-data.frame(test_case_
   control_data[[i]],ro_snp_PC1_weight_top10_test[[i]],
   stringsAsFactors = F)
colnames(test_snp_cases_controls_AIM1_AIM10[[i]])[(ncol(test_
   case_control_data[[1]])+1):(ncol(test_case_control_data[[1]])
   +10)]<-c("AIM1","AIM2","AIM3","AIM4","AIM5","AIM6","AIM7","
   AIM8","AIM9","AIM10")}
```

```
test_case_control_AIM_data_frame<-do.call(rbind, test_snp_cases
   _controls_AIM1_AIM10)
id<-rep(c(1:nrow(test_case_control_data[[1]])),nrun)
run<-rep(1:nrun, each=nrow(test_case_control_data[[1]]))
test_case_control_AIM_data_frame_complete<-data.frame(run,id,
   test_case_control_AIM_data_frame, stringsAsFactors = F)
test_case_control_AIM_data_1<-test_case_control_AIM_data_frame_
   complete[test_case_control_AIM_data_frame_complete$run %in% c
   (1:10),]
# Data samples 2 to 10 were conducted analogously.
test_cases_controls_AIM_save_1_csv <- write.table(test_case_
   control_AIM_data_1, file="test_cases_controls_ro_PCAIMSNP1_
   PCAIMSNP10_IT_UK_FR_DE_PRT_ES_simulated_reference_100_1.csv",
    row.names=FALSE, na="",col.names=FALSE, sep=",")
# Data samples 2 to 10 were saved analogously.
```

C.6 R Code: Informative Ancestry-Informative Markers

The following R codes describe the identification of the *IN*-AIMs of the external data sets of the six European countries sample.

Informative Ancestry-Informative Markers of the six European countries sample

```
## 1.) Read in simulated external data sets and genetic
   association study data sets
nrun=100
set.seed(13022018)
training_case_files = list.files(pattern = 'training_cases_
   controls_IT_UK_FR_DE_PRT_ES_simulated_reference_.*.txt')
library(gtools)
training_case_files <- mixedsort(training_case_files)
training_case_control_data = lapply(training_case_files, read.
   table, header=T)
training_case_control_data_SNPsonly<-list(NA)
training_case_control_data_IT_SNP<-list(NA)
training_case_control_data_UK_SNP<-list(NA)
training_case_control_data_FR_SNP<-list(NA)
training_case_control_data_DE_SNP<-list(NA)
training_case_control_data_PRT_SNP<-list(NA)
training_case_control_data_ES_SNP<-list(NA)
for (i in 1:nrun)
{training_case_control_data_SNPsonly[[i]]<-training_case_
   control_data[[i]][,c(3:ncol(training_case_control_data[[i]]))
   ]
```

```
training_case_control_data_IT_SNP[[i]]<-subset(training_case_
   control_data[[i]], country_reference=="Italy", select=c(3:
   ncol(training_case_control_data[[i]])))
training_case_control_data_UK_SNP[[i]]<-subset(training_case_
   control_data[[i]], country_reference=="UnitedKingdom", select
   =c(3:ncol(training_case_control_data[[i]])))
training_case_control_data_FR_SNP[[i]]<-subset(training_case_
   control_data[[i]], country_reference=="France", select=c(3:
   ncol(training_case_control_data[[i]])))
training_case_control_data_DE_SNP[[i]]<-subset(training_case_
   control_data[[i]], country_reference=="Germany", select=c(3:
   ncol(training_case_control_data[[i]])))
training_case_control_data_PRT_SNP[[i]]<-subset(training_case_
   control_data[[i]], country_reference=="Portugal", select=c(3:
   ncol(training_case_control_data[[i]])))
training_case_control_data_ES_SNP[[i]]<-subset(training_case_
   control_data[[i]], country_reference=="Spain", select=c(3:
   ncol(training_case_control_data[[i]])))}
test_case_files = list.files(pattern = 'test_cases_controls_IT_
   UK_FR_DE_PRT_ES_simulated_reference_.*.txt')
library(gtools)
test_case_files <- mixedsort(test_case_files)
test_case_control_data = lapply(test_case_files, read.table,
   header=T)

## 2.) Calculate informativeness of genetic variants
n_pop<-6
p_IT_allele1<-list(NA)
p_IT_allele2<-list(NA)
p_UK_allele1<-list(NA)
p_UK_allele2<-list(NA)
p_FR_allele1<-list(NA)
p_FR_allele2<-list(NA)
p_DE_allele1<-list(NA)
p_DE_allele2<-list(NA)
p_PRT_allele1<-list(NA)
p_PRT_allele2<-list(NA)
p_ES_allele1<-list(NA)
p_ES_allele2<-list(NA)
p_IT_UK_FR_DE_PRT_ES_allele1<-list(NA)
p_IT_UK_FR_DE_PRT_ES_allele2<-list(NA)
In_IT_UK_FR_DE_PRT_ES<-list(NA)
In_IT_UK_FR_DE_PRT_ES_sort<-list(NA)
In_IT_UK_FR_DE_PRT_ES_overview<-list(NA)
INAIM_IT_UK_FR_DE_PRT_ES_top<-list(NA)
INAIM_IT_UK_FR_DE_PRT_ES_top_test<-list(NA)
for (i in 1:nrun)
```

```
{#Italy
p_IT_allele1_function<-function(x){(2*length(which(x==0))+
   length(which(x==1)))/(2*(length(which(x==0))+length(which(x
   ==1))+length(which(x==2))))}
p_IT_allele2_function<-function(x){(2*length(which(x==2))+
   length(which(x==1)))/(2*(length(which(x==0))+length(which(x
   ==1))+length(which(x==2))))}
p_IT_allele1[[i]]<-apply(training_case_control_data_IT_SNP[[i
   ]],2,p_IT_allele1_function)
p_IT_allele2[[i]]<-apply(training_case_control_data_IT_SNP[[i
   ]],2,p_IT_allele2_function)
#UnitedKingdom
p_UK_allele1_function<-function(x){(2*length(which(x==0))+
   length(which(x==1)))/(2*(length(which(x==0))+length(which(x
   ==1))+length(which(x==2))))}
p_UK_allele2_function<-function(x){(2*length(which(x==2))+
   length(which(x==1)))/(2*(length(which(x==0))+length(which(x
   ==1))+length(which(x==2))))}
p_UK_allele1[[i]]<-apply(training_case_control_data_UK_SNP[[i
   ]],2,p_UK_allele1_function)
p_UK_allele2[[i]]<-apply(training_case_control_data_UK_SNP[[i
   ]],2,p_UK_allele2_function)
#France
p_FR_allele1_function<-function(x){(2*length(which(x==0))+
   length(which(x==1)))/(2*(length(which(x==0))+length(which(x
   ==1))+length(which(x==2))))}
p_FR_allele2_function<-function(x){(2*length(which(x==2))+
   length(which(x==1)))/(2*(length(which(x==0))+length(which(x
   ==1))+length(which(x==2))))}
p_FR_allele1[[i]]<-apply(training_case_control_data_FR_SNP[[i
   ]],2,p_FR_allele1_function)
p_FR_allele2[[i]]<-apply(training_case_control_data_FR_SNP[[i
   ]],2,p_FR_allele2_function)
#Germany
p_DE_allele1_function<-function(x){(2*length(which(x==0))+
   length(which(x==1)))/(2*(length(which(x==0))+length(which(x
   ==1))+length(which(x==2))))}
p_DE_allele2_function<-function(x){(2*length(which(x==2))+
   length(which(x==1)))/(2*(length(which(x==0))+length(which(x
   ==1))+length(which(x==2))))}
p_DE_allele1[[i]]<-apply(training_case_control_data_DE_SNP[[i
   ]],2,p_DE_allele1_function)
p_DE_allele2[[i]]<-apply(training_case_control_data_DE_SNP[[i
   ]],2,p_DE_allele2_function)
#Portugal
```

```
p_PRT_allele1_function<-function(x){(2*length(which(x==0))+
   length(which(x==1)))/(2*(length(which(x==0))+length(which(x
   ==1))+length(which(x==2))))}
p_PRT_allele2_function<-function(x){(2*length(which(x==2))+
   length(which(x==1)))/(2*(length(which(x==0))+length(which(x
   ==1))+length(which(x==2))))}
p_PRT_allele1[[i]]<-apply(training_case_control_data_PRT_SNP[[i
   ]],2,p_PRT_allele1_function)
p_PRT_allele2[[i]]<-apply(training_case_control_data_PRT_SNP[[i
   ]],2,p_PRT_allele2_function)
#Spain
p_ES_allele1_function<-function(x){(2*length(which(x==0))+
   length(which(x==1)))/(2*(length(which(x==0))+length(which(x
   ==1))+length(which(x==2))))}
p_ES_allele2_function<-function(x){(2*length(which(x==2))+
   length(which(x==1)))/(2*(length(which(x==0))+length(which(x
   ==1))+length(which(x==2))))}
p_ES_allele1[[i]]<-apply(training_case_control_data_ES_SNP[[i
   ]],2,p_ES_allele1_function)
p_ES_allele2[[i]]<-apply(training_case_control_data_ES_SNP[[i
   ]],2,p_ES_allele2_function)
#together
p_IT_UK_FR_DE_PRT_ES_allele1[[i]]<-p_IT_allele1[[i]]/n_pop+p_UK
   _allele1[[i]]/n_pop+p_FR_allele1[[i]]/n_pop+p_DE_allele1[[i]]
   /n_pop+p_PRT_allele1[[i]]/n_pop+p_ES_allele1[[i]]/n_pop
p_IT_UK_FR_DE_PRT_ES_allele2[[i]]<-p_IT_allele2[[i]]/n_pop+p_UK
   _allele2[[i]]/n_pop+p_FR_allele2[[i]]/n_pop+p_DE_allele2[[i]]
   /n_pop+p_PRT_allele2[[i]]/n_pop+p_ES_allele2[[i]]/n_pop
log_function<-function(x){ifelse(x==0,0,log(x))}
In_IT_UK_FR_DE_PRT_ES[[i]]<- (-p_IT_UK_FR_DE_PRT_ES_allele1[[i
   ]]*log_function(p_IT_UK_FR_DE_PRT_ES_allele1[[i]])+(p_IT_
   allele1[[i]]/n_pop*log_function(p_IT_allele1[[i]])) + (p_UK_
   allele1[[i]]/n_pop*log_function(p_UK_allele1[[i]])) +(p_FR_
   allele1[[i]]/n_pop*log_function(p_FR_allele1[[i]])) +(p_DE_
   allele1[[i]]/n_pop*log_function(p_DE_allele1[[i]]))+(p_PRT_
   allele1[[i]]/n_pop*log_function(p_PRT_allele1[[i]]))+(p_ES_
   allele1[[i]]/n_pop*log_function(p_ES_allele1[[i]])))+ (-p_IT_
   UK_FR_DE_PRT_ES_allele2[[i]]*log_function(p_IT_UK_FR_DE_PRT_
   ES_allele2[[i]])+ (p_IT_allele2[[i]]/n_pop*log_function(p_IT_
   allele2[[i]])) + (p_UK_allele2[[i]]/n_pop*log_function(p_UK_
   allele2[[i]]))+ (p_FR_allele2[[i]]/n_pop*log_function(p_FR_
   allele2[[i]]))+ (p_DE_allele2[[i]]/n_pop*log_function(p_DE_
   allele2[[i]]))+ (p_PRT_allele2[[i]]/n_pop*log_function(p_PRT_
   allele2[[i]]))+ (p_ES_allele2[[i]]/n_pop*log_function(p_ES_
   allele2[[i]])))
In_IT_UK_FR_DE_PRT_ES_sort[[i]]<-sort(In_IT_UK_FR_DE_PRT_ES[[i
   ]],decreasing=T, na.last=NA)
```

```
In_IT_UK_FR_DE_PRT_ES_overview[[i]]<-data.frame(names(In_IT_UK_
   FR_DE_PRT_ES_sort[[i]]),In_IT_UK_FR_DE_PRT_ES_sort[[i]],
   stringsAsFactors=FALSE)
colnames(In_IT_UK_FR_DE_PRT_ES_overview[[i]])<-c("rsID","
   information_content")
INAIM_IT_UK_FR_DE_PRT_ES_top[[i]]<-training_case_control_data[[
   i]][,In_IT_UK_FR_DE_PRT_ES_overview[[i]][1:10,1]]
INAIM_IT_UK_FR_DE_PRT_ES_top_test[[i]]<-test_case_control_data
   [[i]][,In_IT_UK_FR_DE_PRT_ES_overview[[i]][1:10,1]]}
In_IT_UK_FR_DE_PRT_ES_overview_save <- for(i in 1:nrun) {write.
   table(In_IT_UK_FR_DE_PRT_ES_overview[[i]][1:10,], file=paste(
   "training_INAIM1_INAIM10_simulated_reference_IT_UK_FR_DE_PRT_
   ES_",i,".txt", sep=""), row.names=F, col.names=F)}

## 5.) Save identified genetic variants with genetic
   association study data sets
test_snp_cases_controls_AIM1_AIM10<-list(NA)
for(i in 1:nrun)
{test_snp_cases_controls_AIM1_AIM10[[i]]<-data.frame(test_case_
   control_data[[i]],INAIM_IT_UK_FR_DE_PRT_ES_top_test[[i]],
   stringsAsFactors = F)
colnames(test_snp_cases_controls_AIM1_AIM10[[i]])[(ncol(test_
   case_control_data[[1]])+1):(ncol(test_case_control_data[[1]])
   +10)]<-c("AIM1","AIM2","AIM3","AIM4","AIM5","AIM6","AIM7","
   AIM8","AIM9","AIM10")}
test_case_control_AIM_data_frame<-do.call(rbind, test_snp_cases
   _controls_AIM1_AIM10)
id<-rep(c(1:nrow(test_case_control_data[[1]])),nrun)
run<-rep(1:nrun, each=nrow(test_case_control_data[[1]]))
test_case_control_AIM_data_frame_complete<-data.frame(run,id,
   test_case_control_AIM_data_frame, stringsAsFactors = F)
test_case_control_AIM_data_1<-test_case_control_AIM_data_frame_
   complete[test_case_control_AIM_data_frame_complete$run %in% c
   (1:10),]
# Data samples 2 to 10 were conducted analogously.
test_cases_controls_AIM_save_1_csv <- write.table(test_case_
   control_AIM_data_1, file="test_cases_controls_INAIM1_INAIM10_
   IT_UK_FR_DE_PRT_ES_simulated_reference_100_1.csv", row.names=
   FALSE, na="",col.names=FALSE, sep=",")
# Data samples 2 to 10 were saved analogously.
```

C.7 SAS Code: Simulated Genetic Association Studies

In the following, the SAS codes for regression analyses including the top ten classical PCs per iteration (first ten iterations are displayed; split of iterations necessary due to limited SAS

working memory) of the simulated genetic association studies based on the six European countries for Study Design 1 are displayed. The SAS codes for further iterations, study designs, adjustment methods and the SAS codes for the genetic association studies based on the Swiss sample were conducted analogously and are therefore not displayed.

Genetic association studies of the six European countries sample for Study Design 1 with classical PCs adjustment

```
*1.) Read in genetic association study data sets with top ten
   PCs;
proc import datafile="test_cases_controls_PC1_PC10_IT_UK_FR_DE_
   PRT_ES_simulated_reference_100_1.csv"
     out=sim_ref
     dbms=csv
     replace;
     getnames=no;
run;

*2.) Prepare data set;
data sim_ref_var (keep=run id case country var5-var12839 pc1-
   pc10);
set sim_ref (keep=var1-var12849
rename=(var12840-var12849=pc1-pc10));
run=var1;
id=var2;
case=var3;
country=var4;
run;
proc transpose data=sim_ref_var out=sim_ref_var_trans prefix=
   var;
by run id case country pc1-pc10;
var var5-var12839;
run;
proc sort data=sim_ref_var_trans;
by run _NAME_;
run;

*3.) Run logistic regression model;
%MACRO LOGREG;
ods exclude all;
proc logistic data=sim_ref_var_trans;
by run _NAME_;
class case country  / param=glm;
model case (EVENT="1")= var1 country pc1-pc&I/ clodds=pl;
ods output ParameterEstimates = pvalchi&I;
ods output ConvergenceStatus = conv&I;
```

```
run;
ods exclude off;

*4.) Extract p-values;
data pvalchi_new&I;
set pvalchi&I (keep= run _NAME_ Variable WaldChiSq ProbChiSq);
where (Variable = "var1");
run;
%MEND LOGREG;
%MACRO LOOP;
%DO I=1 %TO 10;
%LOGREG;
%END;
%MEND LOOP;
%LOOP;

*5.) Save p-values and information on model convergence
data pvalchi_complete;
set pvalchi_new1-pvalchi_new10;
by run _NAME_ Variable;
run;
data conv_complete;
set conv1-conv10;
by run _NAME_;
run;
proc export data=pvalchi_complete
outfile='pval_chisq_cc2_CV_test_reference_IT_UK_FR_DE_PRT_ES_1.
   csv'
dbms=csv
replace;
run;
proc export data=conv_complete
outfile='conv_cc2_CV_test_reference_IT_UK_FR_DE_PRT_ES_1.csv'
dbms=csv
replace;
run;
```

C.8 R Code: Assessment of Type I Error Rate

The following R codes describe the assessment of the type I error rates of the simulated genetic association studies adjusted by the classical PC1 based on the six European countries for Study Design 1. The R codes for further classical PCs, study designs and adjustment

methods and the R codes for the assessment of the type I error rates of the genetic association study based on the Swiss sample were conducted analogously and are therefore not displayed.

Type I error rate of the six European countries sample for Study Design 1 with classical PC1 adjustment

```
## 1.) Read in simulated genetic association study data sets
nrun=100
set.seed(13022018)
case_files = list.files(pattern = 'test_cases_controls_IT_UK_FR
   _DE_PRT_ES_simulated_reference_.*.txt')
library(gtools)
case_files <- mixedsort(case_files)
case_control_data = lapply(case_files, read.table, header=T)

## 2.) Read in p-values and information on model convergence
cc2_pval_chisq_results_files = list.files(pattern = 'pval_chisq
   _cc2_CV_test_test_reference_IT_UK_FR_DE_PRT_ES_.*.csv')
library(gtools)
cc2_pval_chisq_results_files <- mixedsort(cc2_pval_chisq_
   results_files)
cc2_pval_chisq_results_tables = lapply(cc2_pval_chisq_results_
   files, read.csv, header=T)
cc2_pval_chisq_results <- do.call(rbind , cc2_pval_chisq_
   results_tables)
cc2_conv_files = list.files(pattern = 'conv_cc2_CV_test_test_
   reference_IT_UK_FR_DE_PRT_ES_.*.csv')
library(gtools)
cc2_conv_files <- mixedsort(cc2_conv_files)
cc2_conv_tables = lapply(cc2_conv_files, read.csv, header=T)
cc2_conv <- do.call(rbind , cc2_conv_tables)
cc2_pval_chisq_results$Variable<-c(rep(c("PC1","PC2","PC3","PC4
   ","PC5","PC6","PC7","PC8","PC9","PC10"),length(unique(cc2_
   pval_chisq_results$X_NAME_))*nrun))
cc2_conv$Variable<-c(rep(c("PC1","PC2","PC3","PC4","PC5","PC6",
   "PC7","PC8","PC9","PC10"),length(unique(cc2_pval_chisq_
   results$X_NAME_))*nrun))

## 3.) Consider only p-values when model convergence criteria
   was fullfilled
cc2_results<-merge(cc2_pval_chisq_results,cc2_conv, by=c("X_
   NAME_","run", "Variable"), all = F)
cc2_results$pval<-ifelse(cc2_results$ProbChiSq=="<.0001"
   ,0.0001,ifelse(cc2_results$ProbChiSq=="" | cc2_results$Reason
   !="Convergence␣criterion␣(GCONV=1E-8)␣satisfied." ,NA,as.
   numeric(levels(cc2_results$ProbChiSq))[cc2_results$ProbChiSq
   ]))
```

```
cc2_results$chi<-ifelse(cc2_results$WaldChiSq=="NA" | cc2_
   results$Reason!="Convergence␣criterion␣(GCONV=1E-8)␣satisfied
   ." ,NA,cc2_results$WaldChiSq)

## 4.) Merge p-values and respective genetic variants
   information
SNP_name<-data.frame(c(rep(1:nrun,each=12835)),c(rep(paste("VAR
   ",5:12839,sep=""),nrun)),c(rep(colnames(case_control_data
   [[1]])[3:ncol(case_control_data[[1]])],nrun)),
   stringsAsFactors = F)
colnames(SNP_name)<-c("run","X_NAME_","SNPID")
cc2_results_merge<-merge(cc2_results,SNP_name, by=c("X_NAME_","
   run"), all = F)
cc2_results_final<-cc2_results_merge[,c("run","SNPID","Variable
   ","pval","chi")]
cc2_pval_chisq_subset<-list(NA)
cc2_pval_chisq<-list(NA)
cc2_pval_chisq_subset_PC1<-list(NA)
cc2_pval_chisq_PC1<-list(NA)
for (i in 1:nrun){cc2_pval_chisq_subset[[i]]<-subset(cc2_
   results_final, run==i)
cc2_pval_chisq_subset_PC1[[i]]<-subset(cc2_pval_chisq_subset[[i
   ]], Variable=="PC1")
cc2_pval_chisq_PC1[[i]]<-cc2_pval_chisq_subset_PC1[[i]][match(
   SNP_name$SNPID[1:12835], cc2_pval_chisq_subset_PC1[[i]]$SNPID
   ),]}

## 5.) Calculate MAF for MAF groups determination
MAF<-list(NA)
for (i in 1:nrun)
{MAF[[i]]<-(apply(case_control_data[[i]][,c(3:ncol(case_control
   _data[[i]]))]==1,2,sum,na.rm=TRUE)+2*apply(case_control_data
   [[i]][,c(3:ncol(case_control_data[[i]]))]==2,2,sum,na.rm=TRUE
   ))/(nrow(case_control_data[[i]])*2)}

## 6.) Type I error rates for classical PC1 adjustment
type1_error_PC1<-NULL
for (i in 1:nrun){type1_error_PC1[i]<-length(which(cc2_pval_
   chisq_PC1[[i]]$pval<=0.05))/sum(is.na(cc2_pval_chisq_PC1[[i]]
   $pval)==F)}
type1_error_average_PC1<- round(sum(type1_error_PC1, na.rm=T)/
   nrun,4)
type1_error_lower_PC1<-round(type1_error_average_PC1 - qt
   (0.975, df=nrun-1) * sd(type1_error_PC1, na.rm=T)/sqrt(nrun)
   ,4)
```

```
type1_error_upper_PC1<-round(type1_error_average_PC1 + qt
   (0.975, df=nrun-1) * sd(type1_error_PC1, na.rm=T)/sqrt(nrun)
   ,4)
type1_error_CI_PC1<-c(paste("[",type1_error_lower_PC1,",",type1
   _error_upper_PC1,"]",sep = ""))
type1_error_CI_results_PC1<-data.frame(type1_error_average_PC1,
   type1_error_CI_PC1, stringsAsFactors = F)
write.table(type1_error_CI_results_PC1, file="type1_error_CV_
   test_test_reference_IT_UK_FR_DE_PRT_ES_PC1.txt", row.names=F,
    col.names=T)

## 7.) Type I error rates for classical PC1 adjustment
   dependent on MAF group 1
MAF_adj<-lapply(MAF,function(x){x<-ifelse(x<=0.5,x,1-x)})
MAF1_PC1<-list(NA)
type1_error_MAF1_PC1<-NULL
for(i in 1:nrun){MAF1_PC1[[i]]<-cc2_pval_chisq_PC1[[i]][c(which
   (MAF_adj[[i]]<=0.05)),]$pval
type1_error_MAF1_PC1[i]<-length(which(MAF1_PC1[[i]]<=0.05))/sum
   (is.na(MAF1_PC1[[i]])==F)}
type1_error_MAF1_average_PC1<- round(sum(type1_error_MAF1_PC1,
   na.rm=T)/nrun,4)
type1_error_MAF1_lower_PC1<-round(type1_error_MAF1_average_PC1
   - qt(0.975, df=nrun-1) * sd(type1_error_MAF1_PC1, na.rm=T)/
   sqrt(nrun),4)
type1_error_MAF1_upper_PC1<-round(type1_error_MAF1_average_PC1
   + qt(0.975, df=nrun-1) * sd(type1_error_MAF1_PC1, na.rm=T)/
   sqrt(nrun),4)
type1_error_MAF1_CI_PC1<-c(paste("[",type1_error_MAF1_lower_PC1
   ,",",type1_error_MAF1_upper_PC1,"]",sep = ""))

## 8.) Type I error rates for classical PC1 adjustment
   dependent on MAF group 2
MAF2_PC1<-list(NA)
type1_error_MAF2_PC1<-NULL
for(i in 1:nrun){MAF2_PC1[[i]]<-cc2_pval_chisq_PC1[[i]][c(which
   (MAF_adj[[i]]>0.05 & MAF_adj[[i]]<=0.20)),]$pval
type1_error_MAF2_PC1[i]<-length(which(MAF2_PC1[[i]]<=0.05))/sum
   (is.na(MAF2_PC1[[i]])==F)}
type1_error_MAF2_average_PC1<- round(sum(type1_error_MAF2_PC1,
   na.rm=T)/nrun,4)
type1_error_MAF2_lower_PC1<-round(type1_error_MAF2_average_PC1
   - qt(0.975, df=nrun-1) * sd(type1_error_MAF2_PC1, na.rm=T)/
   sqrt(nrun),4)
type1_error_MAF2_upper_PC1<-round(type1_error_MAF2_average_PC1
   + qt(0.975, df=nrun-1) * sd(type1_error_MAF2_PC1, na.rm=T)/
   sqrt(nrun),4)
```

```
type1_error_MAF2_CI_PC1<-c(paste("[",type1_error_MAF2_lower_PC1
   ,",",type1_error_MAF2_upper_PC1,"]",sep = ""))

## 9.) Type I error rates for classical PC1 adjustment
   dependent on MAF group 3
MAF3_PC1<-list(NA)
type1_error_MAF3_PC1<-NULL
for(i in 1:nrun){MAF3_PC1[[i]]<-cc2_pval_chisq_PC1[[i]][c(which
   (MAF_adj[[i]]>0.20 & MAF_adj[[i]]<=0.35)),]$pval
type1_error_MAF3_PC1[i]<-length(which(MAF3_PC1[[i]]<=0.05))/sum
   (is.na(MAF3_PC1[[i]])==F)}
type1_error_MAF3_average_PC1<- round(sum(type1_error_MAF3_PC1,
   na.rm=T)/nrun,4)
type1_error_MAF3_lower_PC1<-round(type1_error_MAF3_average_PC1
   - qt(0.975, df=nrun-1) * sd(type1_error_MAF3_PC1, na.rm=T)/
   sqrt(nrun),4)
type1_error_MAF3_upper_PC1<-round(type1_error_MAF3_average_PC1
   + qt(0.975, df=nrun-1) * sd(type1_error_MAF3_PC1, na.rm=T)/
   sqrt(nrun),4)
type1_error_MAF3_CI_PC1<-c(paste("[",type1_error_MAF3_lower_PC1
   ,",",type1_error_MAF3_upper_PC1,"]",sep = ""))

## 10.) Type I error rates for classical PC1 adjustment
   dependent on MAF group 4
MAF4_PC1<-list(NA)
type1_error_MAF4_PC1<-NULL
for(i in 1:nrun){MAF4_PC1[[i]]<-cc2_pval_chisq_PC1[[i]][c(which
   (MAF_adj[[i]]>0.35 & MAF_adj[[i]]<=0.50)),]$pval
type1_error_MAF4_PC1[i]<-length(which(MAF4_PC1[[i]]<=0.05))/sum
   (is.na(MAF4_PC1[[i]])==F)}
type1_error_MAF4_average_PC1<- round(sum(type1_error_MAF4_PC1,
   na.rm=T)/nrun,4)
type1_error_MAF4_lower_PC1<-round(type1_error_MAF4_average_PC1
   - qt(0.975, df=nrun-1) * sd(type1_error_MAF4_PC1, na.rm=T)/
   sqrt(nrun),4)
type1_error_MAF4_upper_PC1<-round(type1_error_MAF4_average_PC1
   + qt(0.975, df=nrun-1) * sd(type1_error_MAF4_PC1, na.rm=T)/
   sqrt(nrun),4)
type1_error_MAF4_CI_PC1<-c(paste("[",type1_error_MAF4_lower_PC1
   ,",",type1_error_MAF4_upper_PC1,"]",sep = ""))
type1_error_CI_MAF_results_PC1<-data.frame(type1_error_MAF1_
   average_PC1,type1_error_MAF1_CI_PC1,type1_error_MAF2_average_
   PC1,type1_error_MAF2_CI_PC1,type1_error_MAF3_average_PC1,
   type1_error_MAF3_CI_PC1,type1_error_MAF4_average_PC1,type1_
   error_MAF4_CI_PC1,stringsAsFactors = F)
```

```
write.table(type1_error_CI_MAF_results_PC1, file="type1_error_
   MAF_CV_test_test_reference_IT_UK_FR_DE_PRT_ES_PC1.txt", row.
   names=F, col.names=T)
```

C.9 R Code: Assessment of Power

The following R codes describe the assessment of the power of the simulated genetic association studies adjusted by the country information, classical PCs and (classical) AIMs based on the six European countries for Study Design 1. The R codes for further study designs and adjustment methods and the R codes for the assessment of the power of the genetic association study based on the Swiss sample were conducted analogously and are therefore not displayed.

Power of the six European countries sample for Study Design 1

```
## 1.) Read in simulated genetic association study data sets
nrun=100
nrunSNP=1000
set.seed(13022018)
n_TOTAL<-252
n_cases_TOTAL<-126
case_files = list.files(pattern = 'test_cases_controls_IT_UK_FR
   _DE_PRT_ES_simulated_reference_.*.txt')
library(gtools)
case_files <- mixedsort(case_files)
case_control_data = lapply(case_files, read.table, header=T)

## 2.) Read in classical PCs based genetic association study
   data sets
PC_files = list.files(pattern = 'test_test_st_ind_PC1_PC10_
   simulated_reference_IT_UK_FR_DE_PRT_ES_.*.txt')
library(gtools)
PC_files <- mixedsort(PC_files)
PC_data = lapply(PC_files, read.table, header=F)
PC_names<-c("PC1","PC2","PC3","PC4","PC5","PC6","PC7","PC8","
   PC9","PC10")
eig_ind_PC1_PC10<-list(NA)
for (i in 1:nrun){eig_ind_PC1_PC10[[i]]<-PC_data[[i]]
colnames(eig_ind_PC1_PC10[[i]])<-PC_names}

## 3.) Read in classical FST-AIMs based external data sets
FST_AIM_files = list.files(pattern = 'training_st_ind_FSTAIM1_
   FSTAIM10_simulated_reference_IT_UK_FR_DE_PRT_ES_.*.txt')
```

```
library(gtools)
FST_AIM_files <- mixedsort(FST_AIM_files)
FST_AIM_data = lapply(FST_AIM_files, read.table, header=F)
FST_AIM<-list(NA)
for(i in 1:nrun){  FST_AIM[[i]]<-data.frame(case_control_data[[
   i]][,c(as.character(FST_AIM_data[[i]][,1]))],
   stringsAsFactors = F)}
AIM_names<-c("AIM1","AIM2","AIM3","AIM4","AIM5","AIM6","AIM7","
   AIM8","AIM9","AIM10")
FST_AIM1_AIM10<-list(NA)
for (i in 1:nrun){FST_AIM1_AIM10[[i]]<-FST_AIM[[i]]
colnames(FST_AIM1_AIM10[[i]])<-AIM_names}

## 4.) Read in classical PC-AIMs based external data sets
PC_AIM_files = list.files(pattern = 'training_st_ind_PCAIM1_
   PCAIM10_simulated_reference_IT_UK_FR_DE_PRT_ES_.*.txt')
library(gtools)
PC_AIM_files <- mixedsort(PC_AIM_files)
PC_AIM_data = lapply(PC_AIM_files, read.table, header=F)
PC_AIM<-list(NA)
for(i in 1:nrun){  PC_AIM[[i]]<-data.frame(case_control_data[[i
   ]][,c(as.character(PC_AIM_data[[i]][,1]))], stringsAsFactors
   = F)}
AIM_names<-c("AIM1","AIM2","AIM3","AIM4","AIM5","AIM6","AIM7","
   AIM8","AIM9","AIM10")
PC_AIM1_AIM10<-list(NA)
for (i in 1:nrun){PC_AIM1_AIM10[[i]]<-PC_AIM[[i]]
colnames(PC_AIM1_AIM10[[i]])<-AIM_names}

## 5.) Read in classical WL-AIMs based external data sets
PCSNP_AIM_files = list.files(pattern = 'training_st_snp_
   PCAIMSNP1_PCAIMSNP10_simulated_reference_IT_UK_FR_DE_PRT_ES_.
   *.txt')
library(gtools)
PCSNP_AIM_files <- mixedsort(PCSNP_AIM_files)
PCSNP_AIM_data = lapply(PCSNP_AIM_files, read.table, header=F)
PCSNP_AIM<-list(NA)
for(i in 1:nrun){PCSNP_AIM[[i]]<-data.frame(case_control_data[[
   i]][,c(as.character(PCSNP_AIM_data[[i]][,1]))],
   stringsAsFactors = F)}
AIM_names<-c("AIM1","AIM2","AIM3","AIM4","AIM5","AIM6","AIM7","
   AIM8","AIM9","AIM10")
PCSNP_AIM1_AIM10<-list(NA)
for (i in 1:nrun){PCSNP_AIM1_AIM10[[i]]<-PCSNP_AIM[[i]]
colnames(PCSNP_AIM1_AIM10[[i]])<-AIM_names}

## 6.) Read in IN-AIMs based external data sets
```

```
IN_AIM_files = list.files(pattern = 'training_INAIM1_INAIM10_
   simulated_reference_IT_UK_FR_DE_PRT_ES_.*.txt')
IN_AIM_data = lapply(IN_AIM_files, read.table, header=F)
IN_AIM<-list(NA)
for(i in 1:nrun){IN_AIM[[i]]<-data.frame(case_control_data[[i
   ]][,c(as.character(IN_AIM_data[[i]][,1]))], stringsAsFactors
   = F)}
AIM_names<-c("AIM1","AIM2","AIM3","AIM4","AIM5","AIM6","AIM7","
   AIM8","AIM9","AIM10")
IN_AIM1_AIM10<-list(NA)
for (i in 1:nrun){IN_AIM1_AIM10[[i]]<-IN_AIM[[i]]
colnames(IN_AIM1_AIM10[[i]])<-AIM_names}

## 7.) Simulation causal/noncausal genetic variant
require(GeneticsDesign)
# 7.a) Additive Model Northern Europe
pA_add<-0.05
pD_add_North<-0.0335
GRR_hom_add<-seq(1.1,15,0.1)
additive_power_North<-NA
for (i in 1:length(GRR_hom_add)){additive_power_North[i]<-GPC.
   default(pA=pA_add, pD=pD_add_North, RRAa=round(((GRR_hom_add[
   i]+1)/2),1), RRAA=GRR_hom_add[i], Dprime=0.99, pB=pA_add,
   nCase=n_cases_TOTAL/2, ratio=1, alpha=0.05, quiet=TRUE)$power
   }
add_results_North<-data.frame(cbind(GRR_hom_add,round(((GRR_hom
   _add+1)/2),1),additive_power_North),stringsAsFactors = F)
colnames(add_results_North)<-c("RRAA","RRAa","power_additive")
RRAA_add_North<-add_results_North[min(which(add_results_North$
   power_additive >0.5)),]$RRAA
RRAa_add_North<-add_results_North[min(which(add_results_North$
   power_additive >0.5)),]$RRAa
geno_freq_add_North<-GPC.default(pA=pA_add, pD=pD_add_North,
   RRAa=RRAa_add_North, RRAA=RRAA_add_North, Dprime=0.99, pB=pA_
   add, nCase=n_cases_TOTAL/2, ratio=1, alpha=0.05, quiet=TRUE)$
   mat.gFreq
causal_snp_additive_model_North<-data.frame(replicate(nrunSNP,
   sample(c(0, 1, 2), n_cases_TOTAL/2, prob=c(geno_freq_add_
   North[3,1],geno_freq_add_North[2,1],geno_freq_add_North[1,1])
   , replace=T)), stringsAsFactors = F)
colnames(causal_snp_additive_model_North)<-c(1:nrunSNP)
non_causal_snp_additive_model_North<-data.frame(replicate(
   nrunSNP,sample(c(0, 1, 2), n_cases_TOTAL/2, prob=c(geno_freq_
   add_North[3,2],geno_freq_add_North[2,2],geno_freq_add_North
   [1,2]), replace=T)), stringsAsFactors = F)
colnames(non_causal_snp_additive_model_North)<-c(1:nrunSNP)
```

```
# 7.b) Additive Model Southern Europe
pA_add<-0.05
pD_add_South<-0.0407
GRR_hom_add<-seq(1.1,15,0.1)
additive_power_South<-NA
for (i in 1:length(GRR_hom_add)){additive_power_South[i]<-GPC.
   default(pA=pA_add, pD=pD_add_South, RRAa=round(((GRR_hom_add[
   i]+1)/2),1), RRAA=GRR_hom_add[i], Dprime=0.99, pB=pA_add,
   nCase=n_cases_TOTAL/2, ratio=1, alpha=0.05, quiet=TRUE)$power
   }
add_results_South<-data.frame(cbind(GRR_hom_add,round(((GRR_hom
   _add+1)/2),1),additive_power_South),stringsAsFactors = F)
colnames(add_results_South)<-c("RRAA","RRAa","power_additive")
RRAA_add_South<-add_results_South[min(which(add_results_South$
   power_additive >0.5)),]$RRAA
RRAa_add_South<-add_results_South[min(which(add_results_South$
   power_additive >0.5)),]$RRAa
geno_freq_add_South<-GPC.default(pA=pA_add, pD=pD_add_South,
   RRAa=RRAa_add_South, RRAA=RRAA_add_South, Dprime=0.99, pB=pA_
   add, nCase=n_cases_TOTAL/2, ratio=1, alpha=0.05, quiet=TRUE)$
   mat.gFreq
causal_snp_additive_model_South<-data.frame(replicate(nrunSNP,
   sample(c(0, 1, 2), n_cases_TOTAL/2, prob=c(geno_freq_add_
   South[3,1],geno_freq_add_South[2,1],geno_freq_add_South[1,1])
   , replace=T)), stringsAsFactors = F)
colnames(causal_snp_additive_model_South)<-c(1:nrunSNP)
non_causal_snp_additive_model_South<-data.frame(replicate(
   nrunSNP,sample(c(0, 1, 2), n_cases_TOTAL/2, prob=c(geno_freq_
   add_South[3,2],geno_freq_add_South[2,2],geno_freq_add_South
   [1,2]), replace=T)), stringsAsFactors = F)
colnames(non_causal_snp_additive_model_South)<-c(1:nrunSNP)

# 7.c) Merge non-/causal genetic variants
case_control_snp<-data.frame(matrix(NA, nrow=n_TOTAL, ncol=2*
   nrunSNP))
seq_SNP<-rep(1:100, each=10)
for (j in 1:nrunSNP){case_control_snp[,j]<-case_control_data[[
   seq_SNP[j]]]$case
case_control_snp[,(j+nrunSNP)][case_control_snp[,j]==1 & case_
   control_data[[seq_SNP[j]]]$country_reference %in% c("
   UnitedKingdom","France","Germany")]<-causal_snp_additive_
   model_North[,j]
case_control_snp[,(j+nrunSNP)][case_control_snp[,j]==0 & case_
   control_data[[seq_SNP[j]]]$country_reference %in% c("
   UnitedKingdom","France","Germany")]<-non_causal_snp_additive_
   model_North[,j]
```

```
case_control_snp[,(j+nrunSNP)][case_control_snp[,j]==1 & case_
   control_data[[seq_SNP[j]]]$country_reference %in% c("Italy","
   Spain","Portugal")]<-causal_snp_additive_model_South[,j]
case_control_snp[,(j+nrunSNP)][case_control_snp[,j]==0 & case_
   control_data[[seq_SNP[j]]]$country_reference %in% c("Italy","
   Spain","Portugal")]<-non_causal_snp_additive_model_South[,j]}
colnames(case_control_snp)<-c(paste("case", 1:1000, sep=""),
   paste("SNP", 1:1000, sep=""))
case_control_snp_country<-case_control_snp
case_control_snp_country$country<-as.character(case_control_
   data[[1]]$country_reference)

## 8.) Power country adjustment
cc1_causal_pval_add<-NA
for (j in 1:nrunSNP){cc1_causal_add_result= tryCatch({coef(
   summary(glm(case_control_snp_country[,j] ~ case_control_snp_
   country[,(1000+j)]+country, family=binomial, data=case_
   control_snp_country)))[2,4]}, warning = function(w) {NA},
   error = function(e) {NA})
if (is.na(cc1_causal_add_result)=="TRUE") {return_value_add <-
   NA} else {return_value_add <- cc1_causal_add_result}
cc1_causal_pval_add[j]<-as.numeric(return_value_add)}
cc1_power_causal_add<-length(which(cc1_causal_pval_add<=0.05))/
   length(cc1_causal_pval_add)

## 9.) Power classical PCs adjustment
case_control_snp_country_PC1_PC10<-data.frame(matrix(NA, nrow=n
   _TOTAL, ncol=2*nrunSNP+10*10*100))
case_control_snp_country_PC1_PC10[,c(1:2000)]<-case_control_snp
for (i in 1:nrun){case_control_snp_country_PC1_PC10[,(2000+i*
   100-99):(2000+i*100)]<-data.frame(rep(eig_ind_PC1_PC10[[i
   ]],10))}
colnames(case_control_snp_country_PC1_PC10)<-c(paste("case",
   1:1000, sep=""),paste("SNP", 1:1000, sep=""),c(rbind(paste("
   PC1_", 1:1000, sep=""),paste("PC2_", 1:1000, sep=""),paste("
   PC3_", 1:1000, sep=""),paste("PC4_", 1:1000, sep=""),paste("
   PC5_", 1:1000, sep=""),paste("PC6_", 1:1000, sep=""),paste("
   PC7_", 1:1000, sep=""),paste("PC8_", 1:1000, sep=""),paste("
   PC9_", 1:1000, sep=""),paste("PC10_", 1:1000, sep=""))))
case_control_snp_country_PC1_PC10$country<-as.character(case_
   control_data[[1]]$country_reference)
cc2_causal_pval_add<-matrix(NA, nrow=nrunSNP, ncol=length(PC_
   names))
for (j in 1:nrunSNP){for (c in 1:length(PC_names))
{cc2_causal_add_names <- c(names(case_control_snp_country_PC1_
   PC10)[(1000+j)],names(case_control_snp_country_PC1_PC10)
   [(2000+j*10-(10-1)):(2000+j*10-(10-c))],"country")
```

```
cc2_causal_add_f <-paste(cc2_causal_add_names, collapse="␣+␣")
cc2_causal_add_f2<-names(case_control_snp_country_PC1_PC10)[j]
cc2_causal_add_ff<-as.formula(paste(cc2_causal_add_f2,"␣~␣",
   paste(cc2_causal_add_f[!cc2_causal_add_f %in% "cci"],
   collapse = "␣+␣")))
cc2_causal_add_result = tryCatch({coef(summary(glm(cc2_causal_
   add_ff, family=binomial, data=case_control_snp_country_PC1_
   PC10)))[2,4]}, warning = function(w) {NA}, error = function(e
   ) {NA})
if (is.na(cc2_causal_add_result)=="TRUE") {return_value_add <-
   NA} else {return_value_add <- cc2_causal_add_result}
cc2_causal_pval_add[j,c]<-as.numeric(return_value_add)}}
cc2_power_causal_add<-apply(cc2_causal_pval_add,2,function(x){
   length(which(x<=0.05))/length(x)})
# Power of different adjustment methods was conducted
   analogously.
```