Supplemental source code

This section provides

1. the code adaptations to use the Hampel function in the robust logistic regression framework,
2. example source code for the simulation study in the reference scenario including the calculation of the statistical properties and
3. the source code for the real data application.

Note: The working directories have to be set accordingly in each R script. To use the Hampel function, the file glmrobMqle.r in the R package robustbase has to be adapted according to the instructions. Be aware of expensive calculations.

R extension to use the Hampel function

Code adaptations and instructions to extent the R package robustbase to the use with the Hampel function

General instructions

The instructions are exactly orientated on the glmrobMqle.R file of the package "robustbase" version 0.9-8 (Date: 14/06/2013). All code sections have to be placed exactly where they can be found in the original file for the Huber function. Afterwards, the package has again to be installed as a whole under a modified name, e.g. "robustbaseAdj".

Installation:
Only the 32-bit R version is allowed to be installed for the R version considered for the package. Otherwise the installation will not work. Then:

1. Open command line
2. Go to directory R/bin
3. Write into command line

   R CMD INSTALL path

   with path="path to folder robustbaseAdj containing the R package"

Code pieces

```
# Border values for the residual intervals relying on the Hampel
function
Ha <- floor(mu*ni-tcc[1]*sni*sV);
Ka <- floor(mu*ni+tcc[1]*sni*sV);
Hb <- floor(mu*ni-tcc[2]*sni*sV);
Kb <- floor(mu*ni+tcc[2]*sni*sV);
Hc <- floor(mu*ni-tcc[3]*sni*sV);
Kc <- floor(mu*ni+tcc[3]*sni*sV);

# psi(y_i, mu_i)-E[psi(y_i, mu_i)]
# with Epsi=E[psi(r_i)]
```

```
cpsi <-
  ifelse((0<=abs(residPS)) & (abs(residPS)<tcc[1]), residPS,
    ifelse((tcc[1]<=abs(residPS)) & (abs(residPS)<tcc[2]),
        sign(residPS)*tcc[1],
      ifelse((tcc[2]<=abs(residPS)) & (abs(residPS)<tcc[3]),
          sign(residPS)*tcc[1]*(tcc[3]-abs(residPS))/(tcc[3]-
           tcc[2]), 0))) - eval(Epsi)
# because:
# 0=sum(nu(y_i,mu_i)w(x_i)mu_i'-alpha(beta))
# <=> 0=sum(nu(y_i,mu_i)-E[nu(y_i,mu_i)])w(x_i)mu_i'
# <=> 0=sum(psi(y_i,mu_i)-
#          E[psi(y_i,mu_i)])w(x_i)mu_i'/V^(1/2)(mu_i)
# => cpsi:=psi(y_i,mu_i)-E[psi(y_i,mu_i)]

# psi(r_i)/r_i (weights)
w.r <- ifelse(abs(residPS)<tcc[1], 1,
  ifelse((tcc[1]<=abs(residPS)) & (abs(residPS)<tcc[2]),
      tcc[1]/abs(residPS),
    ifelse((tcc[2]<=abs(residPS)) & (abs(residPS)<tcc[3]),
        (tcc[3]-abs(residPS))/(tcc[3]-
          tcc[2])*tcc[1]/abs(residPS), 0)))

# Binomial distribution probabilities
EpsiBin.init <- expression({
  # P[Y_i <= j_z2]
  pKa <- pbinom(Ka, ni, mu); pKb <- pbinom(Kb, ni, mu);
  pKc <- pbinom(Kc, ni, mu)
  # P[Y_i <= j_z1]
  pHa <- pbinom(Ha, ni, mu); pHb <- pbinom(Hb, ni, mu);
  pHc <- pbinom(Hc, ni, mu)
  # P[Y_i~ <= j_z2-1]
  pKam1 <- pbinom(Ka-1, pmax.int(0, ni-1), mu)
  pKbm1 <- pbinom(Kb-1, pmax.int(0, ni-1), mu)
  pKcm1 <- pbinom(Kc-1, pmax.int(0, ni-1), mu)
  # P[Y_i~ <= j_z1-1]
  pHam1 <- pbinom(Ha-1, pmax.int(0, ni-1), mu)
  pHbm1 <- pbinom(Hb-1, pmax.int(0, ni-1), mu)
  pHcm1 <- pbinom(Hc-1, pmax.int(0, ni-1), mu)
  # P[Y_i~~ <= j_z2-2]
  pKam2 <- pbinom(Ka-2, pmax.int(0, ni-2), mu)
  pKbm2 <- pbinom(Kb-2, pmax.int(0, ni-2), mu)
  pKcm2 <- pbinom(Kc-2, pmax.int(0, ni-2), mu)
  # P[Y_i~~ <= j_z1-2]
  pHam2 <- pbinom(Ha-2, pmax.int(0, ni-2), mu)
  pHbm2 <- pbinom(Hb-2, pmax.int(0, ni-2), mu)
  pHcm2 <- pbinom(Hc-2, pmax.int(0, ni-2), mu)
})

# Fisher consistency correction
EpsiBin <- expression({
  tcc[1]*(pKb-pKa-pHa+pHb) +
  tcc[1]*tcc[3]/(tcc[3]-tcc[2])*(pKc-pKb-pHb+pHc) +
  (pKam1-pHam1-pKa+pHa)*mu*sni/sV -
  tcc[1]/(tcc[3]-tcc[2])*((pKcm1-pKbm1-pKc+pKb)*mu*sni/sV +
    (pHbm1-pHcm1-pHb+pHc)*mu*sni/sV)
})
```

```
# Asymptotic variance

# First expectation
Epsi2Bin <- expression({
  tcc[1]*tcc[1]*(pKb-pKa+pHa-pHb) +
  tcc[1]*tcc[1]*tcc[3]*tcc[3]/(tcc[3]-tcc[2])/(tcc[3]-tcc[2])*
    (pKc-pKb+pHb-pHc) +
  mu*mu*ni/Vmu*(pKa-pHa) +
  tcc[1]*tcc[1]/(tcc[3]-tcc[2])/(tcc[3]-tcc[2])*mu*mu*ni/Vmu*
    (pKc-pKb+pHb-pHc) +
  mu/Vmu*(ni-1)*mu*(pKam2-pHam2) +
  tcc[1]*tcc[1]/(tcc[3]-tcc[2])/(tcc[3]-tcc[2])*mu/Vmu*(ni-1)*
    mu*((pKcm2-pKbm2)+(pHbm2-pHcm2)) +
  mu/Vmu*(1-2*mu*ni)*(pKam1-pHam1) +
  tcc[1]*tcc[1]/(tcc[3]-tcc[2])/(tcc[3]-tcc[2])*mu/Vmu*
    (1-2*mu*ni)*((pKcm1-pKbm1)+(pHbm1-pHcm1)) -
  tcc[1]*tcc[1]/(tcc[3]-tcc[2])/(tcc[3]-tcc[2])*2*tcc[3]*
    ((pKcm1-pKbm1-pKc+pKb)*mu*sni/sV-(pHbm1-pHcm1-
      pHb+pHc)*mu*sni/sV)
})

# 2nd expectation
EpsiSBin <- expression({
  Q2V + ifelse(ni==0, 0, Q1V/sni/sV)
})
# with
Q1V <- mu*mu*ni/Vmu*(pKa-pHa) -
  tcc[1]/(tcc[3]-tcc[2])*mu*mu*ni/Vmu*(pKc-pKb+pHb-pHc) +
  mu*mu*(ni-1)/Vmu*(pKam2-pHam2) -
  tcc[1]/(tcc[3]-tcc[2])*mu*mu*(ni-1)/Vmu*
    ((pKcm2-pKbm2)+(pHbm2-pHcm2)) +
  mu/Vmu*(1-2*mu*ni)*(pKam1-pHam1) -
  tcc[1]/(tcc[3]-tcc[2])*mu/Vmu*(1-2*mu*ni)*
    ((pKcm1-pKbm1)+(pHbm1-pHcm1))
# and
Q2V <- tcc[1]*((pKbm1-pKam1-pKb+pKa)*mu/Vmu -
    (pHam1-pHbm1-pHa+pHb)*mu/Vmu) +
  tcc[1]*tcc[3]/(tcc[3]-tcc[2])*
    ((pKcm1-pKbm1-pKc+pKb)*mu/Vmu-(pHbm1-pHcm1-pHb+pHc)*mu/Vmu)
```

Simulation study

Code example for the reference scenario of the simulation study comprising the calculation of allele frequencies, simulation of the population and the logistic regression analysis with the calculation of the statistical properties (type I error rate, bias, variance, MSE, statistical power)

First, the random numbers are drawn to sample individuals from the population. Then, the allele frequencies in the population are calculated and the population is simulated. Finally, standard and robust logistic regression is applied and the statistical properties are calculated.

Random samples

Random numbers to draw 1000 cases and 1000 controls from the population comprising 3,500,000 cases and 3,500,000 controls for 400 studies

```
# Working directories
dir.save <- "Directory to save results"

# Settings
set.seed(12061950)
repetitions <- 400
pop.size <- 3500000
no.cases <- 1000
no.controls <- 1000

# Define 400 samples
stichprobe <- data.frame(reps=1:repetitions,
                 probe=matrix(NA, ncol=(no.cases+no.controls),
nrow=repetitions))
# Draw samples
for (i in 1:repetitions){
    sample.ca <- sample(1:pop.size, no.cases)
    sample.co <- sample((pop.size+1):(2*pop.size), no.controls)
    stichprobe[i, 2:(no.cases+1)] <- sort(sample.ca)
    stichprobe[i, (no.cases+2):(no.cases+no.controls+1)] <-
sort(sample.co)
}
# Save the results
setwd(dir.save)
write.csv2(stichprobe, "Stichprobe_Daten_JenaFinal.csv",
row.names=FALSE)
```

Allele frequencies

Structure:

0. Preparations
1. Marker
   a. Read data and prepare it
   b. Function definitions
   c. Calculate allele frequencies
2. Null marker
   a. Calculate allele frequencies

```
# ------------------------------------------------------------------ #
# ------------------------------------------------------------------ #
# 0. Preparations
# ------------------------------------------------------------------ #
# ------------------------------------------------------------------ #

# Working directory
dir.read <- "Directory to read data"
dir.save <- "Directory to save results"

# Constant values
# Study
no.cases      <- 1000
no.controls   <- 1000
repetition    <- 400
# Marker
p.c               <- 0.05
penetrance.model <- "dominant”
Dp.r2             <- data.frame(dp=1, r2=1)
# Null marker
no.null <- 10
GRR1 <- 1
GRR2 <- 1

# ------------------------------------------------------------------ #
# ------------------------------------------------------------------ #
# 1. Marker
# ------------------------------------------------------------------ #
# ------------------------------------------------------------------ #

# ------------------------------------------------------------------ #
# a. Read data and prepare it
# ------------------------------------------------------------------ #

# Read data: Incidence rates and age distribution
setwd(dir.read)
alter.inzidenz <- read.table("Globocan_Inzidenz.txt", header=TRUE,
  sep="\t")
alter.distr <- read.table("EU25Pop.txt", header=TRUE, sep=" ")

# Select colorectal cancer for incidence rates
alter.inzidenz <-
alter.inzidenz[alter.inzidenz$Cancer=="Colorectum", 4:12]
```

```
rownames(alter.inzidenz) <- "inzidenz"
alter.inzidenz <- t(alter.inzidenz)
inzidenz <- data.frame(alter=seq(35, 75, 5),
inzidenz=alter.inzidenz)
rownames(inzidenz) <- 1:nrow(inzidenz)

# Select age intervals
alter.distr <- alter.distr[, 9:17]
alter.distr <- t(alter.distr)
alter <- data.frame(alter=seq(35, 75, 5), distr=alter.distr)
rownames(alter) <- 1:nrow(alter)

# GRR
grr.c.hom.mat     <- data.frame(age.int=alter[, 1], grr.parts=1.43)

# Combine age and incidence rates, calculate cummulative incidence
inzidenz <- data.frame(inzidenz, kumm=cumsum(inzidenz$inzidenz))
# Calculate percentages incidence
inzidenz <- data.frame(inzidenz, proz=inzidenz$kumm*5/100000)
# Incidence according to case-control status
inzidenz <- data.frame(inzidenz, fall=alter$distr*inzidenz$proz,
  kontrolle=alter$distr*(1-inzidenz$proz))
# Scaling within cases and controls to 100%
inzidenz <- data.frame(inzidenz,
fall.proz=inzidenz$fall/sum(inzidenz$fall),
  kontrolle.proz=inzidenz$kontrolle/sum(inzidenz$kontrolle))
# Define age categories
inzidenz <- data.frame(inzidenz,
fall.int=cumsum(inzidenz$fall.proz),
kontrolle.int=cumsum(inzidenz$kontrolle.proz))

# Prevalence matrix
prevalence <- inzidenz[, c(1, 4)]
colnames(prevalence) <- c("age.int", "prev.parts")

# ------------------------------------------------------------------ #
# b. Function definitions
# ------------------------------------------------------------------ #

# Calculate heterozygotic GRR
grr.c.het.expr <- expression({
  grr.c.het <- ifelse(pen.mod=="dominant", grr.c.hom,
                  ifelse(pen.mod == "recessive", 1,
0.5*(grr.c.hom+1))
  )
})

# Calculate kappe_0
kappa.0.expr <- expression({
  kappa.0 <- prev/(pc*pc*grr.c.hom + 2*pc*(1-pc)*grr.c.het +
    (1-pc)*(1-pc))
})

# Calculate p_M
p.m.expr <- expression({
  pm <- 1/(korr*(1-pc)/pc/dp/dp + 1)
```

```
  d   <- dp*(1-pm)*pc
})

# Calculate GRR according to Lorenzo Bermejo et al. (2011)
grr.m.expr <- expression({
  PrCM <- pm*pc+d
  PrCm <- (1-pm)*pc-d
  PrcM <- pm*(1-pc)-d
  Prcm <- (1-pm)*(1-pc)+d

  PrGCMgCM <- grr.c.hom*PrCM*PrCM
  PrGCMgCm <- grr.c.hom*2*PrCM*PrCm
  PrGCmgCm <- grr.c.hom*PrCm*PrCm
  PrGCMgcM <- grr.c.het*2*PrCM*PrcM
  PrGCMgcm <- grr.c.het*2*PrCM*Prcm
  PrGCmgcM <- grr.c.het*2*PrCm*PrcM
  PrGCmgcm <- grr.c.het*2*PrCm*Prcm
  PrGcMgcM <- PrcM*PrcM
  PrGcMgcm <- 2*PrcM*Prcm
  PrGcmgcm <- Prcm*Prcm

  kMMnum  <- PrGCMgCM+PrGCMgcM+PrGcMgcM
  kMMden  <- PrCM*PrCM+2*PrCM*PrcM+PrcM*PrcM
  kMM     <- kMMnum/kMMden

  kMmnum  <- PrGCMgCm+PrGCMgcm+PrGCmgcM+PrGcMgcm
  kMmden  <- 2*PrCM*PrCm+2*PrCM*Prcm+2*PrCm*PrcM+2*PrcM*Prcm
  kMm     <- kMmnum/kMmden

  kmmnum  <- PrGCmgCm+PrGCmgcm+PrGcmgcm
  kmmden  <- PrCm*PrCm+2*PrCm*Prcm+Prcm*Prcm
  kmm     <- kmmnum/kmmden

  grr.m.hom <- kMM/kmm
  grr.m.het <- kMm/kmm

})

# GRR age-dependent
# pA = P(M)
# Result as dataframe

# Allele frequency within cases and controls
# based on script by Justo Lorenzo Bermejo
allelfrequenz <- function(pA, GRR1, GRR2, prev){
  ausgabe <- data.frame(geno=c("aa", "Aa", "AA"), fall=NA,
kontrolle=NA)

  k <- prev
  enda=0
  f=0.00001
  while (enda==0){
    f=f+0.00001;
    #Genotypes among cases
    uno1=(1-pA)*(1-pA)*f;
    uno2=2*pA*(1-pA)*f*GRR2;
```

```
    uno3=pA*pA*f*GRR1;
    den=uno1+uno2+uno3;
    kp=den;
    pAA_case=uno1/den;
    pAB_case=uno2/den;
    pBB_case=uno3/den;

    #Genotypes among controls
    dos1=(1-pA)*(1-pA)*(1-f);
    dos2=2*pA*(1-pA)*(1-f*GRR2);
    dos3=pA*pA*(1-f*GRR1);

    den=dos1+dos2+dos3;
    pAA_cont=dos1/den;
    pAB_cont=dos2/den;
    pBB_cont=dos3/den;

    if (k <= kp) {enda=1};
  }

  ausgabe$fall <- matrix(c(pAA_case, pAB_case, pBB_case), ncol=1)
  ausgabe$kontrolle <- matrix(c(pAA_cont, pAB_cont, pBB_cont),
    ncol=1)

  return(ausgabe)
}

vergleich <- function(geno){
  geno.new <- ifelse(rn.gte<0.5 & geno!="MM", "MM",
                ifelse(rn.gte>=0.5 & geno!="mm", "mm", "Mm"))
  return(geno.new)
}

# ------------------------------------------------------------------ #
# c. Calculate allele frequencies
# ------------------------------------------------------------------ #

rownumber <- nrow(grr.c.hom.mat)*length(p.c)*
  length(penetrance.model)* nrow(Dp.r2)*3
allelfrequenzen <- data.frame(alter=rep(NA, rownumber),
  prev=rep(NA, rownumber), grr=rep(NA, rownumber),
  pen.mod=rep(NA, rownumber), pc=rep(NA, rownumber),
  dp=rep(NA, rownumber), r2=rep(NA, rownumber),
  geno=rep(NA, rownumber), fall=rep(NA, rownumber),
  kontrolle=rep(NA, rownumber))
idx <- 1
for (dpr2 in 1:nrow(Dp.r2)){
  dp <- Dp.r2$dp[dpr2]
  korr <- Dp.r2$r2[dpr2]
  for(pen.mod in penetrance.model){
    for (pc in p.c){
      eval(p.m.expr)
      for (i in 1:length(grr.c.hom.mat$grr.parts)){
        grr.c.hom <- grr.c.hom.mat$grr.parts[i]
        eval(grr.c.het.expr)
        eval(grr.m.expr)
```

```
        prev <- prevalence$prev.parts[i]
        eval(kappa.0.expr)
        z1 <- allelfrequenz(pA=pm, GRR1=grr.m.hom, GRR2=grr.m.het,
          prev=prev)
        allelfrequenzen[idx:(idx+2),
          (length(allelfrequenzen)-1):(length(allelfrequenzen))] <-
           z1[, 2:3]
        allelfrequenzen$geno[idx:(idx+2)] <- c(0,1,2)
        allelfrequenzen$prev[idx:(idx+2)] <- prev
        allelfrequenzen$alter[idx:(idx+2)] <- prevalence$age.int[i]
        allelfrequenzen$grr[idx:(idx+2)] <- grr.c.hom
        allelfrequenzen$pen.mod[idx:(idx+2)] <- pen.mod
        allelfrequenzen$pc[idx:(idx+2)] <- pc
        allelfrequenzen$dp[idx:(idx+2)] <- dp
        allelfrequenzen$r2[idx:(idx+2)] <- korr
        idx <- idx+3
      }
    }
  }
}
setwd(dir.save)
titel <- "Allelfrequenzen_Grundlage_Fall_Kontrolle_EU.csv"
write.csv2(allelfrequenzen, titel)

# ------------------------------------------------------------------ #
# ------------------------------------------------------------------ #
# 2. NULL MARKER
# ------------------------------------------------------------------ #
# ------------------------------------------------------------------ #

set.seed(687541)

# MAF at locus j, j=1, ..., no.null
p0 <- runif(no.null, 0.05, 0.5)

# ------------------------------------------------------------------ #
# a. Calculate allele frequencies
# ------------------------------------------------------------------ #

rownumber <- length(p0)*nrow(prevalence)*3
allelfrequenzen <- data.frame(prev=rep(NA, rownumber),
  p0=rep(NA, rownumber), id=rep(NA, rownumber),
  geno=rep(NA, rownumber), fall=rep(NA, rownumber),
  kontrolle=rep(NA, rownumber))
idx <- 1
idx.p0 <- 0
for (pc in p0){
  idx.p0 <- idx.p0+1
  for(prev in prevalence$prev.parts){
    z1 <- allelfrequenz(pA=pc, GRR1=GRR1, GRR2=GRR2, prev=prev)
    allelfrequenzen[idx:(idx+2),
      (length(allelfrequenzen)- 1):(length(allelfrequenzen))] <-
      z1[, 2:3]
    allelfrequenzen$geno[idx:(idx+2)] <- z1$geno
    allelfrequenzen$prev[idx:(idx+2)] <- prev
    allelfrequenzen$p0[idx:(idx+2)] <- pc
```

```
    allelfrequenzen$id[idx:(idx+2)] <- idx.p0
    idx <- idx+3
  }
}
setwd(dir.save)
titel <-
  "Allelfrequenzen_Grundlage_Fall_Kontrolle_Nullmarker_EU.csv"
write.csv2(allelfrequenzen, titel)
```

Populations

Population at the marker locus

Structure:

0. Preparations
1. Marker
    a. Read data and prepare it
    b. Function definitions
    c. Age, GRR and genotypes

```
# ------------------------------------------------------------------ #
# ------------------------------------------------------------------ #
# 0. Preparations
# ------------------------------------------------------------------ #
# ------------------------------------------------------------------ #

# Working directories
dir.read <- "Directory to read data"
dir.save <- "Directory to save results"

# Constant values
# Number cases and controls
no.cases <- 3500000
no.controls <- 3500000
# Marker
pc <- 0.05
Dp.r2 <- data.frame(dp=1, r2=1)
pen.mod <- "dominant"

# Seed
set.seed(341950)

# ------------------------------------------------------------------ #
# ------------------------------------------------------------------ #
# 1. Marker
# ------------------------------------------------------------------ #
# ------------------------------------------------------------------ #

# ------------------------------------------------------------------ #
# a. Read data and prepare it
# ------------------------------------------------------------------ #
```

```
# Read data: Incidence rates and age distribution
setwd(dir.read)
alter.inzidenz <- read.table("Globocan_Inzidenz.txt", header=TRUE,
sep="\t")
alter.distr <- read.table("EU25Pop.txt", header=TRUE, sep=" ")

# Select colorectal cancer for incidence rates
alter.inzidenz <-
  alter.inzidenz[alter.inzidenz$Cancer=="Colorectum", 4:12]
rownames(alter.inzidenz) <- "inzidenz"
alter.inzidenz <- t(alter.inzidenz)
inzidenz <- data.frame(alter=seq(35, 75, 5),
  inzidenz=alter.inzidenz)
rownames(inzidenz) <- 1:nrow(inzidenz)

# Select age intervals
alter.distr <- alter.distr[, 9:17]
alter.distr <- t(alter.distr)
alter <- data.frame(alter=seq(35, 75, 5), distr=alter.distr)
rownames(alter) <- 1:nrow(alter)

# Combine age and incidence rates, calculate cummulative incidence
inzidenz <- data.frame(inzidenz, kumm=cumsum(inzidenz$inzidenz))
# Calculate percentages incidence
inzidenz <- data.frame(inzidenz, proz=inzidenz$kumm*5/100000)
# Incidence according to case-control status
inzidenz <- data.frame(inzidenz, fall=alter$distr*inzidenz$proz,
  kontrolle=alter$distr*(1-inzidenz$proz))
# Scaling within cases and controls to 100%
inzidenz <- data.frame(inzidenz,
  fall.proz=inzidenz$fall/sum(inzidenz$fall),
  kontrolle.proz=inzidenz$kontrolle/sum(inzidenz$kontrolle))
# Define age categories
inzidenz <- data.frame(inzidenz,
  fall.int=cumsum(inzidenz$fall.proz),
  kontrolle.int=cumsum(inzidenz$kontrolle.proz))

# Prevalence matrix
prevalence <- inzidenz[, c(1, 4)]
colnames(prevalence) <- c("age.int", "prev.parts")

# GRR
grr.c.hom.mat <- data.frame(age.int=alter[, 1], grr.parts=1.43)

# Allele frequencies
allelfrequenzen <-
  read.csv2("Allelfrequenzen_Grundlage_Fall_Kontrolle_EU.csv")
allelfrequenzen <- allelfrequenzen[, 2:length(allelfrequenzen)]

# ------------------------------------------------------------------ #
# b. Function definitions
# ------------------------------------------------------------------ #

# Age calculation
alter.func <- function(zufall){
  if (zufall[2]=="control"){
```

```
     if (as.numeric(zufall[1])<min(inzidenz$kontrolle.int)){
        alter.aus <- min(inzidenz$alter)
     }
     if (as.numeric(zufall[1])>min(inzidenz$kontrolle.int)){
       alter.aus <- inzidenz$alter[max(which(inzidenz$kontrolle.int
        < as.numeric(zufall[1])))+1]
     }
     if (as.numeric(zufall[1])==1){
        alter.aus <- max(inzidenz$alter)
     }
  }
  if (zufall[2]=="case"){
    if (as.numeric(zufall[1])<min(inzidenz$fall.int)){
        alter.aus <- min(inzidenz$alter)
     }
     if (as.numeric(zufall[1])>min(inzidenz$fall.int) &
      as.numeric(zufall[1])<1){
        alter.aus <- inzidenz$alter[max(which(inzidenz$fall.int
          <as.numeric(zufall[1])))+1]
     }
     if (as.numeric(zufall[1])==1){
        alter.aus <- max(inzidenz$alter)
     }
  }
  return(alter.aus)
}

# Calculate heterozygotic GRR
grr.c.het.expr <- expression({
  grr.c.het <- ifelse(rep(pen.mod, no.cases+no.controls) ==
    "dominant", grr.c.hom,
    ifelse(rep(pen.mod, no.cases+no.controls) == "recessive",
    rep(1, no.cases+no.controls), 0.5*(grr.c.hom+1))
  )
})

# Calculate kappe_0
kappa.0.expr <- expression({
  kappa.0 <- prev.1000/(pc*pc*grr.c.hom + 2*pc*(1-pc)*grr.c.het +
    (1-pc)*(1-pc))
})

# Calculate p_M
p.m.expr <- expression({
  pm <- 1/(korr*(1-pc)/pc/dp/dp + 1)
  d   <- dp*(1-pm)*pc
})

# Calculate GRR according to Lorenzo Bermejo et al. (2011)
grr.m.expr <- expression({
  PrCM <- pm*pc+d
  PrCm <- (1-pm)*pc-d
  PrcM <- pm*(1-pc)-d
  Prcm <- (1-pm)*(1-pc)+d

  PrGCMgCM <- grr.c.hom*PrCM*PrCM
```

```
  PrGCMgCm <- grr.c.hom*2*PrCM*PrCm
  PrGCmgCm <- grr.c.hom*PrCm*PrCm
  PrGCMgcM <- grr.c.het*2*PrCM*PrcM
  PrGCMgcm <- grr.c.het*2*PrCM*Prcm
  PrGCmgcM <- grr.c.het*2*PrCm*PrcM
  PrGCmgcm <- grr.c.het*2*PrCm*Prcm
  PrGcMgcM <- PrcM*PrcM
  PrGcMgcm <- 2*PrcM*Prcm
  PrGcmgcm <- Prcm*Prcm

  kMMnum  <- PrGCMgCM+PrGCMgcM+PrGcMgcM
  kMMden  <- PrCM*PrCM+2*PrCM*PrcM+PrcM*PrcM
  kMM     <- kMMnum/kMMden

  kMmnum  <- PrGCMgCm+PrGCMgcm+PrGCmgcM+PrGcMgcm
  kMmden  <- 2*PrCM*PrCm+2*PrCM*Prcm+2*PrCm*PrcM+2*PrcM*Prcm
  kMm     <- kMmnum/kMmden

  kmmnum  <- PrGCmgCm+PrGCmgcm+PrGcmgcm
  kmmden  <- PrCm*PrCm+2*PrCm*Prcm+Prcm*Prcm
  kmm     <- kmmnum/kmmden

  grr.m.hom <- kMM/kmm
  grr.m.het <- kMm/kmm

})

# ------------------------------------------------------------------ #
# c. Age, GRR and genotypes
# ------------------------------------------------------------------ #

# Age
zufallszahlen <- data.frame(zahl=runif((no.cases+no.controls), 0,
  1), status=c(rep("case", no.cases), rep("control", no.controls)))
alter <- apply(as.matrix(zufallszahlen), 1, alter.func)
daten <- data.frame(ca.co=c(rep(1, no.cases), rep(0, no.controls)),
  alter=alter, geno=NA)

# GRR
prev.1000 <- numeric(no.cases+no.controls)
for (prev.age in 1:length(alter)){
    prev.1000[prev.age] <- prevalence$prev.parts[prevalence$age.int
      == alter[prev.age]]
}
grr.c.hom <- numeric(no.cases+no.controls)
for (grr.age in 1:length(alter)){
    grr.c.hom[grr.age] <-
grr.c.hom.mat$grr.parts[grr.c.hom.mat$age.int
  == alter[grr.age]]
}
eval(grr.c.het.expr)
eval(kappa.0.expr)

# Remove objects that are not needed anymore
rm("inzidenz")
rm("vergleich")
```

```
rm("zufallszahlen")

# Genotypes
for (dp.r2 in 1:nrow(Dp.r2)){
    daten <- data.frame(ca.co=c(rep(1, no.cases),
      rep(0, no.controls)), alter=alter, geno=NA)
    dp <- Dp.r2$dp[dp.r2]
    korr <- Dp.r2$r2[dp.r2]
    eval(p.m.expr)    # p_M, d
    eval(grr.m.expr)  # GRR_M,hom , GRR_M,het
    rn <- runif(no.cases+no.controls, 0, 1)
    referenz <- data.frame(alter=alter, aa=NA, Aa=NA, AA=NA)
    bed01 <- allelfrequenzen$pen.mod==pen.mod &
      as.character(allelfrequenzen$pc)==as.character(pc) &
      as.character(allelfrequenzen$dp)==as.character(dp) &
      as.character(allelfrequenzen$r2)==as.character(korr)
    referenz.ca <- referenz[1:no.cases, ]
    for (a.schleife in prevalence$age.int){
       if (dim(referenz.ca[referenz.ca$alter==a.schleife,
        2:4])[1]>0){
          referenz.ca[referenz.ca$alter==a.schleife, 2:4] <-
            data.frame(matrix(rep(allelfrequenzen$fall[bed01 &
            allelfrequenzen$grr ==
            grr.c.hom.mat$grr.parts[grr.c.hom.mat$age.int ==
            a.schleife] &
            as.character(allelfrequenzen$prev) ==
            as.character(prevalence$prev.parts[prevalence$age.int ==
            a.schleife])],
            nrow(referenz.ca[referenz.ca$alter==a.schleife, 2:4])),
            nrow=nrow(referenz.ca[referenz.ca$alter==a.schleife,
            2:4]), byrow=TRUE))
       }
    }
    referenz.co <- referenz[(no.cases+1):(no.cases+no.controls), ]
    for (a.schleife in prevalence$age.int){
       if(dim(referenz.co[referenz.co$alter==a.schleife,
        2:4])[1]>0){
          referenz.co[referenz.co$alter==a.schleife, 2:4] <-
            data.frame(matrix(rep(allelfrequenzen$kontrolle[bed01 &
            allelfrequenzen$grr ==
            grr.c.hom.mat$grr.parts[grr.c.hom.mat$age.int ==
            a.schleife] &
            as.character(allelfrequenzen$prev) ==
            as.character(prevalence$prev.parts[prevalence$age.int ==
            a.schleife])],
            nrow(referenz.co[referenz.co$alter==a.schleife, 2:4])),
            nrow=nrow(referenz.co[referenz.co$alter==a.schleife,
            2:4]), byrow=TRUE))
       }
    }
    referenz <- rbind(referenz.ca, referenz.co)
    rm("referenz.co")
    rm("referenz.ca")
    z01 <- which(rn < referenz$AA)
    daten$geno[z01] <- "MM"
    z01 <- which((rn >= referenz$AA) & (rn < referenz$Aa +
```

```
      referenz$AA))
    daten$geno[z01] <- "Mm"
    z01 <- which(rn >= referenz$Aa + referenz$AA)
    daten$geno[z01] <- "mm"
    rm("referenz")
    rm("rn")
    rm("z01")

    # Numerical genotypes according to a dominant penetrance model
    daten$geno[daten$geno=="MM"] <- 1
    daten$geno[daten$geno=="Mm"] <- 1
    daten$geno[daten$geno=="mm"] <- 0

    setwd(dir.save)
    titel <- "Daten_Bevoelkerung_EU_Geno_7Mio.csv"
    write.csv2(daten, titel, row.names=FALSE)
    rm("daten")
    rm("titel")
    gc()
    print(dp.r2)
}
```

Population at the null marker loci

Structure:

0. Preparations
1. Null marker
    a. Read data and prepare it
    b. Function definitions
    c. Age, GRR and genotypes

```
# ------------------------------------------------------------------ #
# ------------------------------------------------------------------ #
# 0. Preparations
# ------------------------------------------------------------------ #
# ------------------------------------------------------------------ #

# Working directories
dir.read <- "Directory to read data"
dir.save <- "Directory to save results"

# Constant values
# Number cases and controls
no.cases <- 3500000
no.controls <- 3500000

# Null marker
no.null <- 10
pen.mod <- "dominant"
# Seed
set.seed(687541)
# MAF at locus j, j=1, ..., no.null
p0 <- runif(no.null, 0.05, 0.5)
```

```
# Seed
set.seed(18041985)

# ------------------------------------------------------------------ #
# ------------------------------------------------------------------ #
# 1. Null marker
# ------------------------------------------------------------------ #
# ------------------------------------------------------------------ #

# ------------------------------------------------------------------ #
# a. Read data and prepare it
# ------------------------------------------------------------------ #

# Read data: Incidence rates and age distribution
setwd(dir.read)
alter.inzidenz <- read.table("Globocan_Inzidenz.txt", header=TRUE,
  sep="\t")
alter.distr <- read.table("EU25Pop.txt", header=TRUE, sep=" ")

# Select colorectal cancer for incidence rates
alter.inzidenz <-
alter.inzidenz[alter.inzidenz$Cancer=="Colorectum", 4:12]
rownames(alter.inzidenz) <- "inzidenz"
alter.inzidenz <- t(alter.inzidenz)
inzidenz <- data.frame(alter=seq(35, 75, 5),
  inzidenz=alter.inzidenz)
rownames(inzidenz) <- 1:nrow(inzidenz)

# Select age intervals
alter.distr <- alter.distr[, 9:17]
alter.distr <- t(alter.distr)
alter <- data.frame(alter=seq(35, 75, 5), distr=alter.distr)
rownames(alter) <- 1:nrow(alter)

# GRR
grr.c.hom.mat <- data.frame(age.int=alter[, 1], grr.parts=a.)

# Combine age and incidence rates, calculate cummulative incidence
inzidenz <- data.frame(inzidenz, kumm=cumsum(inzidenz$inzidenz))
# Calculate percentages incidence
inzidenz <- data.frame(inzidenz, proz=inzidenz$kumm*5/100000)
# Incidence according to case-control status
inzidenz <- data.frame(inzidenz, fall=alter$distr*inzidenz$proz,
  kontrolle=alter$distr*(1-inzidenz$proz))
# Scaling within cases and controls to 100%
inzidenz <- data.frame(inzidenz,
  fall.proz=inzidenz$fall/sum(inzidenz$fall),
  kontrolle.proz=inzidenz$kontrolle/sum(inzidenz$kontrolle))
# Define age categories
inzidenz <- data.frame(inzidenz,
  fall.int=cumsum(inzidenz$fall.proz),
  kontrolle.int=cumsum(inzidenz$kontrolle.proz))

# Prevalence matrix
prevalence    <- inzidenz[, c(1, 4)]
```

```
colnames(prevalence) <- c("age.int", "prev.parts")

# Allele frequencies
titel <-
  "Allelfrequenzen_Grundlage_Fall_Kontrolle_Nullmarker_EU.csv"
allelfrequenzen <- read.csv2(titel)
allelfrequenzen <- allelfrequenzen[, 2:length(allelfrequenzen)]

# ------------------------------------------------------------------ #
# b. Function definitions
# ------------------------------------------------------------------ #

# Age calculation
alter.func <- function(zufall){
  if (zufall[2]=="control"){
     if (as.numeric(zufall[1])<min(inzidenz$kontrolle.int)){
        alter.aus <- min(inzidenz$alter)
     }
     if (as.numeric(zufall[1])>min(inzidenz$kontrolle.int)){
        alter.aus <- inzidenz$alter[max(which(inzidenz$kontrolle.int
          < as.numeric(zufall[1])))+1]
     }
     if (as.numeric(zufall[1])==1){
        alter.aus <- max(inzidenz$alter)
     }
  }
  if (zufall[2]=="case"){
    if (as.numeric(zufall[1])<min(inzidenz$fall.int)){
        alter.aus <- min(inzidenz$alter)
     }
     if (as.numeric(zufall[1])>min(inzidenz$fall.int) &
      as.numeric(zufall[1])<1){
        alter.aus <- inzidenz$alter[max(which(inzidenz$fall.int <
          as.numeric(zufall[1])))+1]
     }
     if (as.numeric(zufall[1])==1){
        alter.aus <- max(inzidenz$alter)
     }
  }
  return(alter.aus)
}

# ------------------------------------------------------------------ #
# c. Age, GRR and genotypes
# ------------------------------------------------------------------ #

# Age
zufallszahlen <- data.frame(zahl=runif((no.cases+no.controls), 0,
  1), status=c(rep("case", no.cases), rep("control", no.controls)))
alter <- apply(as.matrix(zufallszahlen), 1, alter.func)
rm("alter.distr")
rm("alter.func")
rm("alter.inzidenz")

# Prevalence
prev.1000 <- numeric(no.cases+no.controls)
```

```
for (prev.age in 1:length(alter)){
    prev.1000[prev.age] <- prevalence$prev.parts[prevalence$age.int
      == alter[prev.age]]
}
rm("inzidenz")
rm("zufallszahlen")

# Genotypes
daten <- data.frame(ca.co=c(rep(1, no.cases), rep(0, no.controls)),
  alter=alter, geno=matrix(NA, ncol=no.null,
  nrow=(no.cases+no.controls)))
idx <- 0
for (pc in p0){
    idx <- idx+1
    rn <- runif(no.cases+no.controls, 0, 1)
    referenz <- data.frame(alter=alter, aa=NA, Aa=NA, AA=NA)
    bed01 <- as.character(allelfrequenzen$p0)==as.character(pc)
    referenz.ca <- referenz[1:no.cases, ]
    for (a.schleife in prevalence$age.int){
        if (dim(referenz.ca[referenz.ca$alter==a.schleife,
         2:4])[1]>0){
          referenz.ca[referenz.ca$alter==a.schleife, 2:4] <-
            data.frame(matrix(rep(allelfrequenzen$fall[bed01 &
            as.character(allelfrequenzen$prev) ==
            as.character(prevalence$prev.parts[prevalence$age.int ==
            a.schleife])],
            nrow(referenz.ca[referenz.ca$alter==a.schleife, 2:4])),
            nrow=nrow(referenz.ca[referenz.ca$alter==a.schleife,
            2:4]), byrow=TRUE))
        }
    }
    referenz.co <- referenz[(no.cases+1):(no.cases+no.controls), ]
    for (a.schleife in prevalence$age.int){
      if(dim(referenz.co[referenz.co$alter==a.schleife, 2:4])[1]>0){
          referenz.co[referenz.co$alter==a.schleife, 2:4] <-
            data.frame(matrix(rep(allelfrequenzen$kontrolle[bed01 &
            as.character(allelfrequenzen$prev) ==
            as.character(prevalence$prev.parts[prevalence$age.int ==
            a.schleife])],
            nrow(referenz.co[referenz.co$alter==a.schleife, 2:4])),
            nrow=nrow(referenz.co[referenz.co$alter==a.schleife,
            2:4]), byrow=TRUE))
      }
   }
   referenz <- rbind(referenz.ca, referenz.co)
   z01 <- which(rn < referenz$AA)
   daten[z01, idx+2] <- "MM"
   z01 <- which((rn >= referenz$AA) & (rn < referenz$Aa +
     referenz$AA))
   daten[z01, idx+2] <- "Mm"
   z01 <- which(rn >= referenz$Aa + referenz$AA)
   daten[z01, idx+2] <- "mm"

   # Numerical genotypes according to a dominant penetrance model
   daten[daten[, idx+2]=="MM", idx+2] <- 1
   daten[daten[, idx+2]=="Mm", idx+2] <- 1
```

```
   daten[daten[, idx+2]=="mm", idx+2] <- 0

   print(idx)

}
# Save results
setwd(dir.save)
titel <- paste("Daten_Bevoelkerung_EU_Geno_7Mio_Nullmarker.csv",
sep="")
write.csv2(daten, titel, row.names=FALSE)
```

Logistic regression and statistical properties

Logistic regression at the marker locus and the statistical properties bias, variance, MSE and statistical power

Structure:

0. Preparations
1. Logistic regression
   a. Standard
   b. Huber
   c. Hampel
2. Statistical properties
   a. Power
   b. MSE

```
# ------------------------------------------------------------------ #
# ------------------------------------------------------------------ #
# 0. Preparations
# ------------------------------------------------------------------ #
# ------------------------------------------------------------------ #

# Working directory
dir.read <- "Directory to read data"
dir.save <- "Directory to save results"

# ------------------------------------------------------------------ #
# ------------------------------------------------------------------ #
# 1. Logistic regression
# ------------------------------------------------------------------ #
# ------------------------------------------------------------------ #

# ------------------------------------------------------------------ #
# a. Standard
# ------------------------------------------------------------------ #

# Read data
setwd(dir.read)
dominant <- read.csv2("Daten_Bevoelkerung_EU_Geno_7Mio.csv ")

# Read matrix with randomly drawn individuals from the population
```

```
stichprobe <- read.csv2("Stichprobe_Daten_JenaFinal.csv")

# Settings
repetitions <- 400
pop.size <- 3500000
no.cases <- 1000
no.controls <- 1000

speicher.stand <- data.frame(reps=1:repetitions, pVal.dom=NA,
koeff.dom=NA, se.dom=NA, ci.low.dom=NA, ci.up.dom=NA)
for (i in 1:repetitions){
    # Get case control status for sample i
    ca.co <- dominant$ca.co[as.vector(unlist(stichprobe[i,
      2:length(stichprobe)]))]
    # Get age for sample i
    alter <- dominant$alter[as.vector(unlist(stichprobe[i,
      2:length(stichprobe)]))]
    # Genotype for sample i
    geno.dom <- dominant$geno[as.vector(unlist(stichprobe[i,
      2:length(stichprobe)]))]

    # Regression
    zwischen1 <- glm(ca.co ~ geno.dom + alter, family=binomial(link
      = "logit"))

    # Saving
    speicher.stand$pVal.dom[i] <-
      summary(zwischen1)$coefficients[2,4]
    speicher.stand$koeff.dom[i] <-
      summary(zwischen1)$coefficients[2,1]
    speicher.stand$se.dom[i] <- summary(zwischen1)$coefficients[2,2]
    speicher.stand$ci.low.dom[i] <- speicher.stand$koeff.dom[i] –
      1.96*speicher.stand$se.dom[i]
    speicher.stand$ci.up.dom[i] <-
      speicher.stand$koeff.dom[i]+1.96*speicher.stand$se.dom[i]

    cat(paste("Rep ", i, "\n", sep=""))
}
# Save results
setwd(dir.save)
write.csv2(speicher.stand, "Standard_DomSimu.csv", row.names=FALSE)

# ------------------------------------------------------------------ #
# b. Huber
# ------------------------------------------------------------------ #

# Read data
setwd(dir.read)
dominant <-
  read.csv2("Daten_Bevoelkerung_EU_7Mio_341950_domRef_domNum.csv")

# Read matrix with randomly drawn individuals from the population
stichprobe <- read.csv2("Stichprobe_Daten_JenaFinal.csv")

# Settings
repetitions <- 400
```

```
pop.size <- 3500000
no.cases <- 1000
no.controls <- 1000

library(robustbase)
tun.huber <- 1.345
speicher.huber <- data.frame(reps=1:repetitions, pVal.dom=NA,
  koeff.dom=NA, se.dom=NA, ci.low.dom=NA, ci.up.dom=NA)
for (i in 1:repetitions){
    # Get case control status for sample i
    ca.co <- dominant$ca.co[as.vector(unlist(stichprobe[i,
      2:length(stichprobe)]))]
    # Get age for sample i
    alter <- dominant$alter[as.vector(unlist(stichprobe[i,
      2:length(stichprobe)]))]
    # Genotype for sample i
    geno.dom <- dominant$geno[as.vector(unlist(stichprobe[i,
      2:length(stichprobe)]))]

    # Regression
    zwischen1 <- try(glmrob(ca.co ~ geno.dom + alter,
      family=binomial("logit"), weights.on.x = "hat",tcc=tun.huber),
      silent=TRUE)

    # Saving
    if (sum(class(zwischen1)=="try-error")==0){
      speicher.huber$pVal.dom[i] <-
        summary(zwischen1)$coefficients[2,4]
      speicher.huber$koeff.dom[i] <-
        summary(zwischen1)$coefficients[2,1]
      speicher.huber$se.dom[i] <-
        summary(zwischen1)$coefficients[2,2]
      speicher.huber$ci.low.dom[i] <- speicher.huber$koeff.dom[i] –
        1.96*speicher.huber$se.dom[i]
      speicher.huber$ci.up.dom[i] <-
        speicher.huber$koeff.dom[i]+1.96*speicher.huber$se.dom[i]
    }

    cat(paste("Rep ", i, "\n", sep=""))
}
# Save results
setwd(dir.save)
write.csv2(speicher.huber, "Huber_DomSimu.csv", row.names=FALSE)
detach("package:robustbase")

# ------------------------------------------------------------------ #
# c. Hampel
# ------------------------------------------------------------------ #

# Read data
setwd(dir.read)
dominant <-
read.csv2("Daten_Bevoelkerung_EU_7Mio_341950_domRef_domNum.csv")

# Read matrix with randomly drawn individuals from the population
stichprobe <- read.csv2("Stichprobe_Daten_JenaFinal.csv")
```

```
# Settings
repetitions <- 400
pop.size <- 3500000
no.cases <- 1000
no.controls <- 1000

library(robustbaseAdj)
tun.hampel1 <- c(1.5, 3.5, 8)*0.9016085
tun.hampel2 <- c(2, 4, 8)*0.690794

speicher.hampel <- data.frame(reps=1:repetitions, pVal.dom.09=NA,
  koeff.dom.09=NA, se.dom.09=NA, ci.low.dom.09=NA, ci.up.dom.09=NA,
  pVal.dom.69=NA, koeff.dom.69=NA, se.dom.69=NA, ci.low.dom.69=NA,
  ci.up.dom.69=NA)
for (i in 1:repetitions){
    # Get case control status for sample i
    ca.co <- dominant$ca.co[as.vector(unlist(stichprobe[i,
      2:length(stichprobe)]))]
    # Get age for sample i
    alter <- dominant$alter[as.vector(unlist(stichprobe[i,
      2:length(stichprobe)]))]
    # Genotype for sample i
    geno.dom <- dominant$geno[as.vector(unlist(stichprobe[i,
      2:length(stichprobe)]))]

    # Regression with 1st tuning constant
    zwischen1 <- try(glmrob(ca.co ~ geno.dom + alter,
      family=binomial("logit"), weights.on.x = "hat",
      tcc=tun.hampel1), silent=TRUE)

    # Saving
    if (sum(class(zwischen1)=="try-error")==0){
      speicher.hampel$pVal.dom.09[i] <-
        summary(zwischen1)$coefficients[2,4]
      speicher.hampel$koeff.dom.09[i] <-
        summary(zwischen1)$coefficients[2,1]
      speicher.hampel$se.dom.09[i] <-
        summary(zwischen1)$coefficients[2,2]
      speicher.hampel$ci.low.dom.09[i] <-
        speicher.hampel$koeff.dom.09[i] –
        1.96*speicher.hampel$se.dom.09[i]
      speicher.hampel$ci.up.dom.09[i] <-
        speicher.hampel$koeff.dom.09[i] +
        1.96*speicher.hampel$se.dom.09[i]
    }

    # Regression with 2nd tuning constant
    zwischen1 <- try(glmrob(ca.co ~ geno.dom + alter,
      family=binomial("logit"), weights.on.x = "hat",
      tcc=tun.hampel2), silent=TRUE)

    # Saving
    if (sum(class(zwischen1)=="try-error")==0){
      speicher.hampel$pVal.dom.69[i] <-
        summary(zwischen1)$coefficients[2,4]
```

```
      speicher.hampel$koeff.dom.69[i] <-
        summary(zwischen1)$coefficients[2,1]
      speicher.hampel$se.dom.69[i] <-
        summary(zwischen1)$coefficients[2,2]
      speicher.hampel$ci.low.dom.69[i] <-
        speicher.hampel$koeff.dom.69[i] –
        1.96*speicher.hampel$se.dom.69[i]
      speicher.hampel$ci.up.dom.69[i] <-
        speicher.hampel$koeff.dom.69[i] +
        1.96*speicher.hampel$se.dom.69[i]
    }

    cat(paste("Rep ", i, "\n", sep=""))
}
# Save results
setwd(dir.save)
write.csv2(speicher.hampel, "Hampel_DomSimu.csv", row.names=FALSE)
detach("package:robustbaseAdj")

# ------------------------------------------------------------------ #
# ------------------------------------------------------------------ #
# 2. Statistical properties
# ------------------------------------------------------------------ #
# ------------------------------------------------------------------ #

rm(list=ls())

# ------------------------------------------------------------------ #
# a. Power
# ------------------------------------------------------------------ #

setwd(dir.save)
speicher.stand <- read.csv2("Standard_DomSimu.csv")
speicher.huber <- read.csv2("Huber_DomSimu.csv")
speicher.hampel <- read.csv2("Hampel_DomSimu.csv")

# Standard
power.speicher <- data.frame(Method=c("Standard", "Huber", "Hampel",
  ""), Tuning=c("none", "1.345", "(1.5,3.5,8)*0.9016085",
  "(2,4,8)*0.690794"), Dominant=NA)
power.speicher$Dominant[1] <-
  round(length(speicher.stand$pVal.dom[
  is.na(speicher.stand$pVal.dom)==F &
  speicher.stand$pVal.dom<0.05]) /
  length(speicher.stand$pVal.dom[
  is.na(speicher.stand$pVal.dom)==F])*100, digits=1)

# Huber
power.speicher$Dominant[2] <-
  round(length(speicher.huber$pVal.dom[
  is.na(speicher.huber$pVal.dom)==F &
  speicher.huber$pVal.dom<0.05]) /
  length(speicher.huber$pVal.dom[
  is.na(speicher.huber$pVal.dom)==F])*100, digits=1)

# Hampel with 1st tuning constant
```

```
power.speicher$Dominant[3] <-
  round(length(speicher.hampel$pVal.dom.09[
  is.na(speicher.hampel$pVal.dom.09)==F &
  speicher.hampel$pVal.dom.09<0.05]) /
  length(speicher.hampel$pVal.dom.09[
  is.na(speicher.hampel$pVal.dom.09)==F])*100, digits=1)

# Hampel with 2nd tuning constant
power.speicher$Dominant[4] <-
  round(length(speicher.hampel$pVal.dom.69[
  is.na(speicher.hampel$pVal.dom.69)==F &
  speicher.hampel$pVal.dom.69<0.05]) /
  length(speicher.hampel$pVal.dom.69[
  is.na(speicher.hampel$pVal.dom.69)==F])*100, digits=1)

# Save results
write.table(power.speicher, "Power_SimuDom.txt", row.names=F,
  quote=F, sep=";", dec=".")

# ------------------------------------------------------------- #
# b. MSE
# ------------------------------------------------------------- #

ref <- log(1.43)
mse.speicher <- data.frame(Method=c("Standard", "Huber", "Hampel",
  ""), Tuning=c("none", "1.345", "(1.5,3.5,8)*0.9016085",
  "(2,4,8)*0.690794"), Dominant.MeanBias=NA,
  Dominant.Variance=NA, Dominant.MeanSE=NA)

# Mean bias
mse.speicher$Dominant.MeanBias[1] <-
  round(mean(speicher.stand$koeff.dom)-ref, digits=4)
mse.speicher$Dominant.MeanBias[2] <-
  round(mean(speicher.huber$koeff.dom)-ref, digits=4)
mse.speicher$Dominant.MeanBias[3] <-
  round(mean(speicher.hampel$koeff.dom.09)-ref, digits=4)
mse.speicher$Dominant.MeanBias[4] <-
  round(mean(speicher.hampel$koeff.dom.69)-ref, digits=4)

# Variance
mse.speicher$Dominant.Variance[1] <-
  round(var(speicher.stand$koeff.dom), digits=4)
mse.speicher$Dominant.Variance[2] <-
  round(var(speicher.huber$koeff.dom), digits=4)
mse.speicher$Dominant.Variance[3] <-
  round(var(speicher.hampel$koeff.dom.09), digits=4)
mse.speicher$Dominant.Variance[4] <-
  round(var(speicher.hampel$koeff.dom.69), digits=4)

# Mean squared error
mse.speicher$Dominant.MeanSE[1] <-
  round((mse.speicher$Dominant.MeanBias[1])^2 +
  mse.speicher$Dominant.Variance[1], digits=4)
 mse.speicher$Dominant.MeanSE[2] <-
  round((mse.speicher$Dominant.MeanBias[2])^2 +
  mse.speicher$Dominant.Variance[2], digits=4)
```

```
mse.speicher$Dominant.MeanSE[3] <-
   round((mse.speicher$Dominant.MeanBias[3])^2 +
  mse.speicher$Dominant.Variance[3], digits=4)
mse.speicher$Dominant.MeanSE[4] <-
  round((mse.speicher$Dominant.MeanBias[4])^2 +
  mse.speicher$Dominant.Variance[4], digits=4)

# Save results
write.table(mse.speicher, "MSE_SimuDom.txt", row.names=F, quote=F,
sep=";", dec=".")
```

Logistic regression at the null marker loci and the type I error rate

Structure:

0. Preparations
1. Logistic regression
   a. Standard
   b. Huber
   c. Hampel
2. Statistical properties
   a. Type I error rate

```
# ------------------------------------------------------------------ #
# ------------------------------------------------------------------ #
# 0. Preparations
# ------------------------------------------------------------------ #
# ------------------------------------------------------------------ #

# Working directory
dir.read <- "Directory to read data"
dir.save <- "Directory to save results"

# ------------------------------------------------------------------ #
# ------------------------------------------------------------------ #
# 1. Logistic regression
# ------------------------------------------------------------------ #
# ------------------------------------------------------------------ #

# ------------------------------------------------------------------ #
# a. Standard
# ------------------------------------------------------------------ #

# Read data
setwd(dir.read)
daten <- read.csv2("Daten_Bevoelkerung_EU_Geno_7Mio_Nullmarker.csv")

# Read matrix with randomly drawn individuals from the population
stichprobe <- read.csv2("Stichprobe_Daten_JenaFinal.csv")

# Settings
repetitions <- 400
pop.size <- 3500000
```

```
no.cases <- 1000
no.controls <- 1000

speicher.stand <- data.frame(reps=rep(1:repetitions, 10),
  p0=sort(rep(1:10, 400)), pVal.dom=NA, koeff.dom=NA, se.dom=NA,
  ci.low.dom=NA, ci.up.dom=NA)
idx <- 0
for (pc in 1:10){
   for (i in 1:repetitions){
      idx <- idx+1
      # Get case control status for sample i
      ca.co <- daten$ca.co[as.vector(unlist(stichprobe[i,
        2:length(stichprobe)]))]
      # Get age for sample i
      alter <- daten$alter[as.vector(unlist(stichprobe[i,
        2:length(stichprobe)]))]
      # Genotype for sample i
      geno <- daten[as.vector(unlist(stichprobe[i,
        2:length(stichprobe)])), pc+2]
      geno <- as.character(geno)
      geno.dom <- geno
      geno.dom[geno.dom=="MM"] <- "1"
      geno.dom[geno.dom=="Mm"] <- "1"
      geno.dom[geno.dom=="mm"] <- "0"
      geno.dom <- as.numeric(geno.dom)

      # Regression
      zwischen1 <- glm(ca.co ~ geno.dom + alter,
        family=binomial(link = "logit"))

      # Saving
      speicher.stand$pVal.dom[idx] <-
        summary(zwischen1)$coefficients[2,4]
      speicher.stand$koeff.dom[idx] <-
        summary(zwischen1)$coefficients[2,1]
      speicher.stand$se.dom[idx] <-
        summary(zwischen1)$coefficients[2,2]
      speicher.stand$ci.low.dom[idx] <-
        speicher.stand$koeff.dom[idx] –
        1.96*speicher.stand$se.dom[idx]
      speicher.stand$ci.up.dom[idx] <-
        speicher.stand$koeff.dom[idx] +
        1.96*speicher.stand$se.dom[idx]
   }
   print(pc)
}
# Save results
setwd(dir.save)
write.csv2(speicher.stand, "Standard_DomSimu_Nullmarker.csv",
  row.names=FALSE)

# ------------------------------------------------------------- #
# b. Huber
# ------------------------------------------------------------- #

# Read data
```

```
setwd(dir.read)
daten <- read.csv2("Daten_Bevoelkerung_EU_Geno_7Mio_Nullmarker.csv")

# Read matrix with randomly drawn individuals from the population
stichprobe <- read.csv2("Stichprobe_Daten_JenaFinal.csv")

# Settings
repetitions <- 400
pop.size <- 3500000
no.cases <- 1000
no.controls <- 1000

library(robustbase)
tun.huber <- 1.345
speicher.huber <- data.frame(reps=rep(1:repetitions, 10),
  p0=sort(rep(1:10, 400)), pVal.dom=NA, koeff.dom=NA, se.dom=NA,
  ci.low.dom=NA, ci.up.dom=NA)
idx <- 0
for (pc in 1:10){
   for (i in 1:repetitions){
      idx <- idx+1
      # Get case control status for sample i
      ca.co <- daten$ca.co[as.vector(unlist(stichprobe[i,
        2:length(stichprobe)]))]
      # Get age for sample i
      alter <- daten$alter[as.vector(unlist(stichprobe[i,
        2:length(stichprobe)]))]
      # Genotype for sample i
      geno <- daten[as.vector(unlist(stichprobe[i,
        2:length(stichprobe)])), pc+2]
      geno <- as.character(geno)
      geno.dom <- geno
      geno.dom[geno.dom=="MM"] <- "1"
      geno.dom[geno.dom=="Mm"] <- "1"
      geno.dom[geno.dom=="mm"] <- "0"
      geno.dom <- as.numeric(geno.dom)

      # Regression
      zwischen1 <- try(glmrob(ca.co ~ geno.dom + alter,
        family=binomial("logit"), weights.on.x =
        "hat",tcc=tun.huber), silent=TRUE)

      # Saving
      if (sum(class(zwischen1)=="try-error")==0){
         speicher.huber$pVal.dom[idx] <-
           summary(zwischen1)$coefficients[2,4]
         speicher.huber$koeff.dom[idx] <-
           summary(zwischen1)$coefficients[2,1]
         speicher.huber$se.dom[idx] <-
           summary(zwischen1)$coefficients[2,2]
         speicher.huber$ci.low.dom[idx] <-
           speicher.huber$koeff.dom[idx] –
           1.96*speicher.huber$se.dom[idx]
         speicher.huber$ci.up.dom[idx] <-
           speicher.huber$koeff.dom[idx] +
           1.96*speicher.huber$se.dom[idx]
```

```
      }
   }
print(pc)
}
# Save results
setwd(dir.save)
write.csv2(speicher.huber, "Huber_DomSimu_Nullmarker.csv",
  row.names=FALSE)
detach("package:robustbase")

# ------------------------------------------------------------------ #
# c. Hampel
# ------------------------------------------------------------------ #

# Read data
setwd(dir.read)
daten <- read.csv2("Daten_Bevoelkerung_EU_Geno_7Mio_Nullmarker.csv")

# Read matrix with randomly drawn individuals from the population
stichprobe <- read.csv2("Stichprobe_Daten_JenaFinal.csv")

# Settings
repetitions <- 400
pop.size <- 3500000
no.cases <- 1000
no.controls <- 1000

library(robustbaseAdj)
tun.hampel1 <- c(1.5, 3.5, 8)*0.9016085
tun.hampel2 <- c(2, 4, 8)*0.690794

speicher.hampel <- data.frame(reps=rep(1:repetitions, 10),
  p0=sort(rep(1:10, 400)), pVal.dom.09=NA, koeff.dom.09=NA,
  se.dom.09=NA, ci.low.dom.09=NA, ci.up.dom.09=NA, pVal.dom.69=NA,
  koeff.dom.69=NA, se.dom.69=NA, ci.low.dom.69=NA, ci.up.dom.69=NA)
idx <- 0
for (pc in 1:10){
   for (i in 1:repetitions){
      idx <- idx+1
      # Get case control status for sample i
      ca.co <- daten$ca.co[as.vector(unlist(stichprobe[i,
        2:length(stichprobe)]))]
      # Get age for sample i
      alter <- daten$alter[as.vector(unlist(stichprobe[i,
        2:length(stichprobe)]))]
      # Genotype for sample i
      geno <- daten[as.vector(unlist(stichprobe[i,
        2:length(stichprobe)])), pc+2]
      geno <- as.character(geno)
      geno.dom <- geno
      geno.dom[geno.dom=="MM"] <- "1"
      geno.dom[geno.dom=="Mm"] <- "1"
      geno.dom[geno.dom=="mm"] <- "0"
      geno.dom <- as.numeric(geno.dom)

      # Regression with 1st tuning constant
```

```
      zwischen1 <- try(glmrob(ca.co ~ geno.dom + alter,
        family=binomial("logit"), weights.on.x = "hat",
        tcc=tun.hampel1), silent=TRUE)

      # Saving
      if (sum(class(zwischen1)=="try-error")==0){
         speicher.hampel$pVal.dom.09[idx] <-
           summary(zwischen1)$coefficients[2,4]
         speicher.hampel$koeff.dom.09[idx] <-
           summary(zwischen1)$coefficients[2,1]
         speicher.hampel$se.dom.09[idx] <-
           summary(zwischen1)$coefficients[2,2]
         speicher.hampel$ci.low.dom.09[idx] <-
           speicher.hampel$koeff.dom.09[idx] –
           1.96*speicher.hampel$se.dom.09[idx]
         speicher.hampel$ci.up.dom.09[idx] <-
           speicher.hampel$koeff.dom.09[idx] +
           1.96*speicher.hampel$se.dom.09[idx]
      }

      # Regression with 2nd tuning constant
      zwischen1 <- try(glmrob(ca.co ~ geno.dom + alter,
         family=binomial("logit"), weights.on.x = "hat",
         tcc=tun.hampel2), silent=TRUE)

      # Saving
      if (sum(class(zwischen1)=="try-error")==0){
         speicher.hampel$pVal.dom.69[idx] <-
           summary(zwischen1)$coefficients[2,4]
         speicher.hampel$koeff.dom.69[idx] <-
           summary(zwischen1)$coefficients[2,1]
         speicher.hampel$se.dom.69[idx] <-
           summary(zwischen1)$coefficients[2,2]
         speicher.hampel$ci.low.dom.69[idx] <-
           speicher.hampel$koeff.dom.69[idx] –
           1.96*speicher.hampel$se.dom.69[idx]
         speicher.hampel$ci.up.dom.69[idx] <-
           speicher.hampel$koeff.dom.69[idx] +
           1.96*speicher.hampel$se.dom.69[idx]
      }
   }
   print(pc)
}
# Save results
setwd(dir.save)
write.csv2(speicher.hampel, "Hampel_DomSimu_Nullmarker.csv",
  row.names=FALSE)
detach("package:robustbaseAdj")

# ------------------------------------------------------------------ #
# ------------------------------------------------------------------ #
# 2. Statistical properties
# ------------------------------------------------------------------ #
# ------------------------------------------------------------------ #

rm(list=ls())
```

```
# ------------------------------------------------------------------ #
# a. Type I error rate
# ------------------------------------------------------------------ #

setwd(dir.save)
speicher.stand <- read.csv2("Standard_DomSimu_Nullmarker.csv")
speicher.huber <- read.csv2("Huber_DomSimu_Nullmarker.csv")
speicher.hampel <- read.csv2("Hampel_DomSimu_Nullmarker.csv")

fpr.speicher <- data.frame(Method=c("Standard", "Huber", "Hampel",
  ""), Tuning=c("none", "1.345", "(1.5,3.5,8)*0.9016085",
  "(2,4,8)*0.690794"), FPR.dom=NA)

# Standard
fpr <- round(length(speicher.stand$pVal.dom[
  is.na(speicher.stand$pVal.dom)==F &
  speicher.stand$pVal.dom<0.05]) /
  length(speicher.stand$pVal.dom[
  is.na(speicher.stand$pVal.dom)==F])*100, digits=1)
low <- round((fpr/100-1.96*sqrt(fpr/100*(1-fpr/100)/4000))*100,
  digits=1)
up <- round((fpr/100+1.96*sqrt(fpr/100*(1-fpr/100)/4000) )*100,
  digits=1)
fpr.speicher$FPR.dom[1] <- paste(fpr, " (", low, ", ", up, ")",
  sep="")
# Huber
fpr <- round(length(speicher.huber$pVal.dom[
  is.na(speicher.huber$pVal.dom)==F &
  speicher.huber$pVal.dom<0.05]) /
  length(speicher.huber$pVal.dom[
  is.na(speicher.huber$pVal.dom)==F])*100, digits=1)
low <- round((fpr/100-1.96*sqrt(fpr/100*(1-fpr/100)/4000))*100,
  digits=1)
up <- round((fpr/100+1.96*sqrt(fpr/100*(1-fpr/100)/4000) )*100,
  digits=1)
fpr.speicher$FPR.dom[2] <- paste(fpr, " (", low, ", ", up, ")",
  sep="")
# Hampel with 1st tuning constant
fpr <- round(length(speicher.hampel$pVal.dom.09[
  is.na(speicher.hampel$pVal.dom.09)==F &
  speicher.hampel$pVal.dom.09<0.05]) /
  length(speicher.hampel$pVal.dom.09[
  is.na(speicher.hampel$pVal.dom.09)==F])*100, digits=1)
low <- round((fpr/100-1.96*sqrt(fpr/100*(1-fpr/100)/4000))*100,
  digits=1)
up <- round((fpr/100+1.96*sqrt(fpr/100*(1-fpr/100)/4000) )*100,
  digits=1)
fpr.speicher$FPR.dom[3] <- paste(fpr, " (", low, ", ", up, ")",
  sep="")
# Hampel with 2nd tuning constant
fpr <- round(length(speicher.hampel$pVal.dom.69[
  is.na(speicher.hampel$pVal.dom.69)==F &
 speicher.hampel$pVal.dom.69<0.05]) /
  length(speicher.hampel$pVal.dom.69[
  is.na(speicher.hampel$pVal.dom.69)==F])*100, digits=1)
```

```
low <- round((fpr/100-1.96*sqrt(fpr/100*(1-fpr/100)/4000))*100,
  digits=1)
up <- round((fpr/100+1.96*sqrt(fpr/100*(1-fpr/100)/4000) )*100,
  digits=1)
fpr.speicher$FPR.dom[4] <- paste(fpr, " (", low, ", ", up, ")",
  sep="")

# Save results
write.table(fpr.speicher, "FPR_SimuDom_Nullmarker.txt", row.names=F,
  quote=F, sep=";", dec=".")
```

Real data application

Code for the analysis of the real data as well as the visualisation of their results

Analysis of the real data

Structure:

1. Input formats
2. Settings
3. Preparations
4. Logistic regression

```
# ------------------------------------------------------------------ #
# ------------------------------------------------------------------ #
# 1. Input formats
# ------------------------------------------------------------------ #
# ------------------------------------------------------------------ #

# Genotype data:
#   .csv-file
#   as given by PGP

# Demographic data:
#   .csv-file
#     column names:
#     id            : given PGP
#     age.yrs       : age in years
#     height.cm     : height in cm
#     DataAvailable : is genotype data available?

# ------------------------------------------------------------------ #
# ------------------------------------------------------------------ #
# 2. Settings
# ------------------------------------------------------------------ #
# ------------------------------------------------------------------ #

genotype.dir    <- "Directory of genotype data"
demographic.dir <- "Directory of demographic information"
save.dir        <- "Directory to save analysis results"

# Number of SNPs to read
no.snps <- 1000

# ------------------------------------------------------------------ #
# ------------------------------------------------------------------ #
# 3. Preparations
# ------------------------------------------------------------------ #
# ------------------------------------------------------------------ #

# Get a list of genotype files
setwd(genotype.dir)
dateien <- dir()
```

```
# If there is more than one file for an individual, take the latest
one
# Which individuals do have several?
personen <- substr(dateien, start=1, stop=8)
mehrere <- which(duplicated(personen)==T)
mehrere <- unique(personen[mehrere])
# Manual exclusion
dateien.auswahl <- dateien[dateien !=
  "hu11603C_genome_Angela_Harris_Full_20120618075158.txt"]
dateien.auswahl <- dateien.auswahl[dateien.auswahl !=
  "hu2A4D22_genome_Stephan_George_Full_20130210221109.txt"]
dateien.auswahl <- dateien.auswahl[dateien.auswahl !=
  "hu394092_genome_Paul_Conroy_Full_20110111011125.txt"]
dateien.auswahl <- dateien.auswahl[dateien.auswahl !=
  "hu3D355A_20110727023010.txt"]
dateien.auswahl <- dateien.auswahl[dateien.auswahl !=
  "hu459AD0_genome_Bernard_Moscia_Full_20110116053218.txt"]
dateien.auswahl <- dateien.auswahl[dateien.auswahl !=
  "hu5A1D5F_genome_Matthew_Kelty_Mito_20110331040943.txt"]
  # no mitochondrial or Y chromosome
dateien.auswahl <- dateien.auswahl[dateien.auswahl !=
  "hu5A1D5F_genome_Matthew_Kelty_Y_20110331040918.txt"]
  # no mitochondrial or Y chromosome
dateien.auswahl <- dateien.auswahl[dateien.auswahl !=
  "hu6ED94A_genome_Norman_Megill_Full_20100527043516.txt"]
dateien.auswahl <- dateien.auswahl[dateien.auswahl !=
  "hu6FECE9_genome_jim_berry_Full_20110112103611.txt"]
dateien.auswahl <- dateien.auswahl[dateien.auswahl !=
  "hu840B0B_genome_Brandon_Galbraith_Full_20110124131334.txt"]
dateien.auswahl <- dateien.auswahl[dateien.auswahl !=
  "huAC827A_genome_Jim_Turner_Full_20110324084155.txt"]
dateien.auswahl <- dateien.auswahl[dateien.auswahl !=
  "huBC03A7_genome_Debra_Patek_Full_20110107081441.txt"]
dateien.auswahl <- dateien.auswahl[dateien.auswahl !=
  "huC4A276_genome_Anastasia_Webber_Full_20110910195237.txt"]
dateien.auswahl <- dateien.auswahl[dateien.auswahl !=
  "huC92BC9_genome_William_Ramey_Mito_20120508065539.txt"]
  # no mitochondrial or Y chromosome
dateien.auswahl <- dateien.auswahl[dateien.auswahl !=
  "huC92BC9_genome_William_Ramey_Y_20120508065802.txt"]
  # no mitochondrial or Y chromosome
dateien.auswahl <- dateien.auswahl[dateien.auswahl !=
  "huD57BBF_genome_James_Vick_Full_20101216062019.txt"]
dateien.auswahl <- dateien.auswahl[dateien.auswahl !=
  "huDB1635_20110727031252.txt"]
dateien.auswahl <- dateien.auswahl[dateien.auswahl !=
  "huDD1522_genome_Ken_Mortimer_Full_20140223140859.txt"]
dateien.auswahl <- dateien.auswahl[dateien.auswahl !=
  "huF06AD0_genome_Beau_Gunderson_Full_20110307220402.txt"]

# Read clinical data and exclude individuals without information
# about age, genotype or height
# Read
setwd(demographic.dir)
klin <-
  read.csv2("KlinischeDaten_perHandAusgelesen_23anMe_Verwendet.csv")
```

```
# Exclusion: no age or no genotype data
klin <- klin[is.na(klin$age.yrs)==F, ]
ausschluss <- grep("no", klin$DataAvailable)
klin <- klin[-ausschluss, ]
# Exclusion: no height
probanden <- klin$id[is.na(klin$height.cm)==F]

# Create genotype matrix
setwd(paste(basedir, "23andMe/Text", sep=""))
daten <- data.frame(snp="rs", chr=0, pos=0, geno="XY", pers="hu")
for (i in probanden){
   datei <- grep(i, dateien.auswahl)
   # Exclusion of individuals with inconsistent files
   if (!(i %in% c("hu52E130", "huD00199", "huBB5257"))){
      daten.pers <- read.table(dateien.auswahl[datei], sep="\t",
        nrows=no.snps, header=F)
      colnames(daten.pers) <- c("snp", "chr", "pos", "geno")
      daten.pers <- data.frame(daten.pers, pers=i)
      daten <- rbind(daten, daten.pers)
   }
}
# Delete first row that was created for initialisation
daten <- daten[2:nrow(daten), ]
# Drop unused levels in the genotype matrix
library(gdata)
daten <- drop.levels(daten)

# SNPs that were represented by 208 individuals
snps.max <- c("rs10492938", "rs10492940", "rs10737190",
  "rs10737192", "rs10752733", "rs10797342", "rs10797348",
  "rs10797368", "rs10797380", "rs10797386", "rs10797417",
  "rs10799202", "rs10907192", "rs10909845", "rs10909852",
  "rs10909890", "rs10909901", "rs10909918", "rs10910025",
  "rs10910047", "rs10910050", "rs10910061", "rs10910078",
  "rs10915433", "rs1108600", "rs1123571", "rs11260549",
  "rs11578011", "rs11583257", "rs11583804", "rs11587331",
  "rs11588930", "rs11589102", "rs11590198", "rs11590912",
  "rs1175549", "rs1181875", "rs1181877", "rs1181883", "rs1181888",
  "rs12022929", "rs12024847", "rs12031557", "rs12046158",
  "rs12049628", "rs12073172", "rs12082157", "rs12085231",
  "rs12117836", "rs12119470", "rs12119556", "rs12119711",
  "rs12124147", "rs12135298", "rs12138909", "rs12562167",
  "rs12562637", "rs12562988", "rs12724233", "rs12731705",
  "rs12748963", "rs12749761", "rs12757342", "rs13376356",
  "rs1553291", "rs1569419", "rs1572657", "rs16823542", "rs16823802",
  "rs16824089", "rs16825336", "rs17373634", "rs17399569",
  "rs17399998", "rs1890336", "rs1984069", "rs2017143", "rs2031709",
  "rs2045331", "rs2045332", "rs2055204", "rs2142569", "rs2173049",
  "rs2257182", "rs2275819", "rs2275831", "rs2279702", "rs2279703",
  "rs2296716", "rs2297829", "rs2298217", "rs2377041", "rs2455118",
  "rs2455144", "rs2459994", "rs2460000", "rs2474460", "rs2477703",
  "rs2483260", "rs2483274", "rs2485945", "rs2487670", "rs2487680",
  "rs2493272", "rs2493275", "rs2493285", "rs2493292", "rs2493310",
  "rs2493314", "rs2494428", "rs2494626", "rs2500262", "rs2500286",
  "rs2606411", "rs263526", "rs2643891", "rs2643901", "rs2645065",
  "rs2649588", "rs2651899", "rs2651906", "rs2799182", "rs2817178",
```

```
  "rs2817185", "rs2821007", "rs2821023", "rs2821025", "rs2821040",
  "rs2821063", "rs2840528", "rs2840532", "rs2840538", "rs2843127",
  "rs2843142", "rs2843143", "rs2843160", "rs2887286", "rs2993493",
  "rs3001336", "rs3002685", "rs3002686", "rs3107151", "rs3128291",
  "rs3736330", "rs3737589", "rs3748816", "rs3753242", "rs3762444",
  "rs3765703", "rs3765705", "rs3765731", "rs3765736", "rs3765761",
  "rs3765766", "rs3795263", "rs3813199", "rs3890745", "rs3934834",
  "rs4131373", "rs424079", "rs4276857", "rs4415513", "rs4531246",
  "rs4648377", "rs4648380", "s4648381", "rs4648392", "rs4648398",
  "rs4648426", "rs4648441", "rs4648453", "rs4648482", "rs4648487",
  "rs4648489", "rs4648505", "rs4648524", "rs4648527", "rs4648545",
  "rs4648592", "rs4648808", "rs4648831", "rs4648843", "rs4654479",
  "rs4654480", "rs4654482", "rs6424069", "rs6424074", "rs6603811",
  "rs6659405", "rs6661168", "rs6663840", "rs6667605", "rs6675798",
  "rs6681347", "rs6681938", "rs6683273", "rs6685064", "rs6687776",
  "rs6691155", "rs6695346", "rs6697749", "rs731031", "rs734999",
  "rs7367066", "rs7412983", "rs747827", "rs7515488", "rs7519349",
  "rs7519458", "rs7519807", "rs7522140", "rs7523732", "rs7525092",
  "rs7527871", "rs7528494", "rs7531583", "rs7534897", "rs7535528",
  "rs7538096", "rs7544357", "rs819980", "rs868688", "rs870124",
  "rs870171", "rs871822", "rs878063", "rs878201", "rs880051",
  "rs884080", "rs884940", "rs897634", "rs903901", "rs903903",
  "rs903904", "rs903914", "rs903916", "rs903919", "rs905135",
  "rs908742", "rs926244", "rs9442373", "rs9442380", "rs946758",
  "rs947344", "rs947354", "rs9628616")

# Reduce genotype matrix to the needed SNPs
daten.max.snps <- daten[daten$snp %in% snps.max, ]

# Drop unused levels
library(gdata)
daten.max.snps <- drop.levels(daten.max.snps)
# 208 individuals with 245 SNPs

# Genotype coding according to MAF: identification of the minor
#  allele

# Matrix
# 1 Individual per row, 1 SNP per column, 1 column per clinical
# information

# ID of individuals and SNPs
personen.unique <- as.character(unique(daten.max.snps$pers))
snps.unique <- as.character(unique(daten.max.snps$snp))

# Initialisation
daten.matrix <- matrix(NA, ncol=(length(snps.unique)+2),
  nrow=length(personen.unique))
daten.matrix <- data.frame(daten.matrix)
colnames(daten.matrix) <- c("age", "height", sort(snps.unique))
rownames(daten.matrix) <- personen.unique

# Fill matrix
for (i in 1:nrow(daten.matrix)){
   id <- rownames(daten.matrix)[i]
   daten.matrix$age[i] <- klin$age.yrs[klin$id==id]
```

```
   daten.matrix$height[i] <- klin$height.cm[klin$id==id]
   inter <- data.frame(rs=colnames(daten.matrix)[
     3:ncol(daten.matrix)])
   inter.2 <- merge(inter, daten.max.snps[daten.max.snps$pers==id,
     c("snp", "geno")], by.x="rs", by.y="snp", all.x=T, all.y=F)
   inter.2 <- inter.2[order(inter.2$rs),]
   daten.matrix[i, 3:ncol(daten.matrix)] <- inter.2$geno
}

# Code "--" as NA
for (i in 3:ncol(daten.matrix)){
   daten.matrix[daten.matrix[, i]=="--", i] <- NA
}

# Delete individuals with missing values
daten.matrix.2 <- na.omit(daten.matrix)
daten.matrix <- daten.matrix.2

# Calculate MAF per SNP
# Create tabel with rs, chromosome, position, minor allele, MAF and
# genotype requency
tab.snps <- data.frame(snp=rep(NA, length(snps.unique)), chr=NA,
  pos=NA, min.allel=NA, maf=NA, maf.min.allel=NA, geno.min.allel=NA,
  geno.hetero=NA, geno.max.allel=NA)
# Preparation step: matrix with rs, chromosome and position
tab.inter <- data.frame(snp=snps.unique, chr=NA, pos=NA)
for (i in 1:nrow(tab.inter)){
   tab.inter$chr[i] <- as.numeric(daten$chr[daten$snp ==
     as.character(tab.inter$snp[i])][1])
   tab.inter$pos[i] <- as.numeric(daten$pos[daten$snp ==
     as.character(tab.inter$snp[i])][1])
}
tab.inter <- tab.inter[order(tab.inter$pos), ]
tab.snps[, 1:3] <- tab.inter
# Main step: MAF calculation
for (i in 1:nrow(tab.snps)){
   vektor <- daten.matrix[, colnames(daten.matrix) ==
     as.character(tab.snps$snp[i])]
   summ <- summary(as.factor(vektor))
   # 3 different genotypes
   if (length(names(summ))==3){
     maf.1 <- (2*summ[[1]]+summ[[2]])/(2*sum(summ))
     maf.2 <- (2*summ[[3]]+summ[[2]])/(2*sum(summ))
     selten <- ifelse(maf.1 <= maf.2, 1, 2)
     minor.allel <- ifelse(selten==1, substr(names(summ)[1],
       start=1, stop=1), substr(names(summ)[3], start=1, stop=1))
     genoFrequ.min.allel <- ifelse(selten==1,
       paste(round(100*summ[[1]]/sum(summ), digits=0), " (",
       names(summ)[1], ")", sep=""),
       paste(round(100*summ[[3]]/sum(summ), digits=0), " (",
       names(summ)[3], ")", sep=""))
     genoFrequ.hetero <- paste(round(100*summ[[2]]/sum(summ),
       digits=0), " (", names(summ)[2], ")", sep="")
     genoFrequ.max.allel <- ifelse(selten==1,
       paste(round(100*summ[[3]]/sum(summ), digits=0), " (",
       names(summ)[3], ")", sep=""),
```

```
    paste(round(100*summ[[1]]/sum(summ), digits=0), " (",
    names(summ)[1], ")", sep=""))
# 2 different genotypes
} else if (length(names(summ))==2){
   print(paste("2 genotypes: ", i, ". ", sep=""))
   if (substr(names(summ)[1], start=1,
    stop=1)==substr(names(summ)[1], start=2, stop=2)){
      if (substr(names(summ)[2], start=1,
       stop=1)!=substr(names(summ)[2], start=2, stop=2)){
         maf.1 <- (2*summ[[1]]+summ[[2]])/(2*sum(summ))
         maf.2 <- summ[[2]]/(2*sum(summ))
         selten <- ifelse(maf.1 <= maf.2, 1, 2)
         if (selten==1){
            minor.allel <- substr(names(summ)[1], start=1,
              stop=1)
            a <- substr(names(summ)[2], start=1, stop=1)
            b <- substr(names(summ)[2], start=2, stop=2)
            major.allel <- c(a, b)[which(!(c(a,b)) %in%
              minor.allel)]
         } else if (selten==2){
            major.allel <- substr(names(summ)[1], start=1,
              stop=1)
            a <- substr(names(summ)[2], start=1, stop=1)
            b <- substr(names(summ)[2], start=2, stop=2)
            minor.allel <- c(a, b)[which(!(c(a,b)) %in%
              major.allel)]
         }
         genoFrequ.min.allel <- ifelse(selten==1,
           paste(round(100*summ[[1]]/sum(summ), digits=0), " (",
           names(summ)[1], ")", sep=""), paste("0 (",
           minor.allel, minor.allel, ")", sep=""))
         genoFrequ.hetero <- paste(round(100*summ[[2]]/sum(summ),
           digits=0), " (", names(summ)[2], ")", sep="")
         genoFrequ.max.allel <- ifelse(selten==1, paste("0 (",
           major.allel, major.allel, ")", sep=""),
           paste(round(100*summ[[1]]/sum(summ), digits=0), " (",
           names(summ)[1], ")", sep=""))
      } else if (substr(names(summ)[2], start=1,
       stop=1)==substr(names(summ)[2], start=2, stop=2)){
         maf.1 <- (2*summ[[1]])/(2*sum(summ))
         maf.2 <- (2*summ[[2]])/(2*sum(summ))
         selten <- ifelse(maf.1 <= maf.2, 1, 2)
         if (selten==1){
            minor.allel <- substr(names(summ)[1], start=1,
              stop=1)
         } else if (selten==2){
            minor.allel <- substr(names(summ)[2], start=1,
              stop=1)
         }
         genoFrequ.min.allel <- ifelse(selten==1,
           paste(round(100*summ[[1]]/sum(summ), digits=0), " (",
           names(summ)[1], ")", sep=""),
           paste(round(100*summ[[2]]/sum(summ), digits=0), " (",
           names(summ)[2], ")", sep=""))
         genoFrequ.hetero <- paste("0 (", substr(names(summ)[1],
           start=1, stop=1), substr(names(summ)[2], start=1,
```

```
            stop=1), ")", sep="")
          genoFrequ.max.allel <- ifelse(selten==1,
            paste(round(100*summ[[2]]/sum(summ), digits=0), " (",
            names(summ)[2], ")", sep=""),
            paste(round(100*summ[[1]]/sum(summ), digits=0), " (",
            names(summ)[1], ")", sep=""))
       }
    } else if (substr(names(summ)[1], start=1, stop=1) !=
     substr(names(summ)[1], start=2, stop=2)){
       maf.1 <- summ[[1]]/(2*sum(summ))
       maf.2 <- (summ[[1]]+2*summ[[2]])/(2*sum(summ))
       selten <- ifelse(maf.1 <= maf.2, 1, 2)
       if (selten==2){
          minor.allel <- substr(names(summ)[2], start=1, stop=1)
          a <- substr(names(summ)[2], start=1, stop=1)
          b <- substr(names(summ)[2], start=2, stop=2)
          major.allel <- c(a, b)[which(!(c(a,b)) %in%
            minor.allel)]
       } else if (selten==1){
          a <- substr(names(summ)[1], start=1, stop=1)
          b <- substr(names(summ)[1], start=2, stop=2)
          minor.allel <- c(a, b)[which(!(c(a,b)) %in%
            substr(names(summ)[2], start=1, stop=1))]
          major.allel <- c(a, b)[which(!(c(a,b)) %in%
            minor.allel)]
       }
       genoFrequ.min.allel <- ifelse(selten==1, paste("0 (",
         minor.allel, minor.allel, ")", sep=""),
         paste(round(100*summ[[2]]/sum(summ), digits=0), " (",
         names(summ)[2], ")", sep=""))
       genoFrequ.hetero <- paste(round(100*summ[[1]]/sum(summ),
         digits=0), " (", names(summ)[1], ")", sep="")
       genoFrequ.max.allel <- ifelse(selten==1,
         paste(round(100*summ[[2]]/sum(summ), digits=0), " (",
         names(summ)[2], ")", sep=""), paste("0 (", major.allel,
         major.allel, ")", sep=""))
    }
 # 1 genotype
 } else if (length(names(summ))==1){
    print(paste("1 genoytpe: ", i, ". ", sep=""))
    if (substr(names(summ)[1], start=1, stop=1) ==
     substr(names(summ)[1], start=2, stop=2)){
       maf.1 <- summ[[1]]/(sum(summ))
       maf.2 <- 0
       selten <- ifelse(maf.1 <= maf.2, 1, 2)
       minor.allel <- "-"
       genoFrequ.min.allel <- "0 (--)"
       genoFrequ.hetero <- "0 (--)"
       genoFrequ.max.allel <- paste(round(100*summ[[1]]/sum(summ),
         digits=0), " (", names(summ)[1], ")", sep="")
    } else if (substr(names(summ)[1], start=1, stop=1) !=
     substr(names(summ)[1], start=2, stop=2)){
       maf.1 <- summ[[1]]/(2*sum(summ))
       maf.2 <- summ[[1]]/(2*sum(summ))
       selten <- 1
       minor.allel <- substr(names(summ)[1], start=1, stop=1)
```

```
         genoFrequ.min.allel <- paste("0 (", minor.allel,
           minor.allel, ")", sep="")
         genoFrequ.hetero <- paste(round(100*summ[[1]]/sum(summ),
           digits=0), " (", names(summ)[1], ")", sep="")
         genoFrequ.max.allel <- paste("0 (", substr(names(summ)[1],
           start =2, stop=2), substr(names(summ)[1], start=2,
           stop=2), sep="")
      }
   }
   frequ <- ifelse(selten==1, round(maf.1*100, digits=0),
      round(maf.2*100, digits=0))
   minor.allel.frequ <- paste(frequ, " (", minor.allel, ")", sep="")

   tab.snps$min.allel[i] <- as.character(minor.allel)
   tab.snps$maf[i] <- frequ
   tab.snps$maf.min.allel[i] <- minor.allel.frequ
   tab.snps$geno.min.allel[i] <- genoFrequ.min.allel
   tab.snps$geno.hetero[i] <- genoFrequ.hetero
   tab.snps$geno.max.allel[i] <- genoFrequ.max.allel
}
# Save results
titel <- paste(save.dir, "/SNPs_Uebersicht.csv", sep="")
write.csv2(tab.snps, titel, row.names=F)

# Function for numerical coding of SNPs
kodierung <- function(vektor, rs.no){
   minor <- tab.snps$min.allel[tab.snps$snp==rs.no]
   geno <- paste(minor, minor, sep="")
   vektor <- as.character(vektor)
   vektor[vektor==geno] <- "2"
   vektor[grep(minor, vektor)] <- "1"
   vektor[!(vektor %in% c("1", "2"))] <- "0"
   return(vektor)
}

# Coding
for (i in 3:ncol(daten.matrix)){
   daten.matrix[, i] <- as.numeric(kodierung(daten.matrix[, i],
colnames(daten.matrix)[i]))
}

# Response - median-dichotomised
med <- median(daten.matrix$height)
daten.matrix <- data.frame(response=rep(NA, nrow(daten.matrix)),
  daten.matrix)
daten.matrix$response[daten.matrix$height <= med] <- 0
daten.matrix$response[daten.matrix$height > med] <- 1

# Overview table clinical data
tab.klin <- data.frame(vari=c("No. Persons", "Height [cm]",
  "Median (Q1, Q3)", "Age [years]", "Median (Q1, Q3)", "No. SNPs",
  "MAF [%]", "Median (Q1, Q3)"), value=NA)
idx <- 1
tab.klin$value[idx] <- nrow(daten.matrix)
idx <- idx+1
tab.klin$value[idx] <- ""
```

```
idx <- idx+1
tab.klin$value[idx] <- paste(median(daten.matrix$height), " (",
  quantile(daten.matrix$height, probs=0.25), ", ",
  quantile(daten.matrix$height, probs=0.75), ")", sep="")
idx <- idx+1
tab.klin$value[idx] <- ""
idx <- idx+1
tab.klin$value[idx] <- paste(median(daten.matrix$age), " (",
  quantile(daten.matrix$age, probs=0.25), ", ",
  quantile(daten.matrix$age, probs=0.75), ")", sep="")
idx <- idx+1
tab.klin$value[idx] <- ncol(daten.matrix)-3
idx <- idx+1
tab.klin$value[idx] <- ""
idx <- idx+1
tab.klin$value[idx] <- paste(median(tab.snps$maf), " (",
  quantile(tab.snps$maf, probs=0.25), ", ", quantile(tab.snps$maf,
  probs=0.75), ")", sep="")
titel <- paste(save.dir, "/Datenuebersicht.csv", sep="")
write.csv2(tab.klin, titel, row.names=F)

# ------------------------------------------------------------------ #
# ------------------------------------------------------------------ #
# 4. Logistic regression
# ------------------------------------------------------------------ #
# ------------------------------------------------------------------ #

# Regression - for a fitted additive, dominant and recessive
penetrance model
# In the article: only recessive
tab.rezessiv <- data.frame(snp=tab.snps$snp, chr=tab.snps$chr,
  pos=tab.snps$pos, p.stand=NA, koeff.stand=NA, p.huber=NA,
  koeff.huber=NA, p.hampel.09=NA, koeff.hampel.09=NA,
  p.hampel.06=NA, koeff.hampel.06=NA)

# Standard
for (i in 1:nrow(tab.rezessiv)){
   response <- daten.matrix$response
   alter <- daten.matrix$age
   geno <- daten.matrix[, colnames(daten.matrix) ==
     as.character(tab.rezessiv$snp[i])]
   geno[geno==1] <- 0
   geno[geno==2] <- 1
   zwischen1 <- glm(response ~ geno + alter,
     family=binomial(link = "logit"))
   if (sum(dim(summary(zwischen1)$coeff)==c(3,4))==2){
      est <- summary(zwischen1)$coeff["geno", ]
      tab.rezessiv$p.stand[i] <- est[[4]]
      tab.rezessiv$koeff.stand[i] <- est[[1]]
   }
}

# Huber
library(robustbase)
tun.huber <- 1.345
for (i in 1:nrow(tab.rezessiv)){
```

```
   response <- daten.matrix$response
   alter <- daten.matrix$age
   geno <- daten.matrix[, colnames(daten.matrix) ==
     as.character(tab.rezessiv$snp[i])]
   geno[geno==1] <- 0
   geno[geno==2] <- 1
   zwischen1 <- try(glmrob(response ~ geno + alter,
     family=binomial("logit"), weights.on.x = "hat", tcc=tun.huber),
     silent=TRUE)
   if (sum(class(zwischen1)=="try-error")==0){
         if (sum(dim(summary(zwischen1)$coeff)==c(3,4))==2){
         est <- summary(zwischen1)$coeff["geno", ]
         tab.rezessiv$p.huber[i] <- est[[4]]
         tab.rezessiv$koeff.huber[i] <- est[[1]]
      }
   }
}
detach("package:robustbase")

# Hampel
library(robustbaseAdj)
tun.hampel1 <- c(1.5, 3.5, 8)*0.9016085
tun.hampel2 <- c(2, 4, 8)*0.690794
for (i in 1:nrow(tab.rezessiv)){
   response <- daten.matrix$response
   alter <- daten.matrix$age

   # 1st tuning constant
   geno <- daten.matrix[, colnames(daten.matrix) ==
     as.character(tab.rezessiv$snp[i])]
   geno[geno==1] <- 0
   geno[geno==2] <- 1
   zwischen1 <- try(glmrob(response ~ geno + alter,
     family=binomial("logit"), weights.on.x = "hat",
     tcc=tun.hampel1), silent=TRUE)
   if (sum(class(zwischen1)=="try-error")==0){
         if (sum(dim(summary(zwischen1)$coeff)==c(3,4))==2){
         est <- summary(zwischen1)$coeff["geno", ]
         tab.rezessiv$p.hampel.09[i] <- est[[4]]
         tab.rezessiv$koeff.hampel.09[i] <- est[[1]]
      }
   }

   # 2nd tuning constant
   geno <- daten.matrix[, colnames(daten.matrix) ==
     as.character(tab.rezessiv$snp[i])]
   geno[geno==1] <- 0
   geno[geno==2] <- 1
   zwischen1 <- try(glmrob(response ~ geno + alter,
     family=binomial("logit"), weights.on.x = "hat",
     tcc=tun.hampel2), silent=TRUE)
   if (sum(class(zwischen1)=="try-error")==0){
         if (sum(dim(summary(zwischen1)$coeff)==c(3,4))==2){
         est <- summary(zwischen1)$coeff["geno", ]
         tab.rezessiv$p.hampel.06[i] <- est[[4]]
         tab.rezessiv$koeff.hampel.06[i] <- est[[1]]
```

```
      }
   }
}
detach("package:robustbaseAdj")

# Save results
titel <- paste(save.dir, "/Ergebnisse/Regression_komplErg_Rez.csv",
  sep="")
write.csv2(tab.rezessiv, titel, row.names=F)
```

Results of the real data analysis

Structure:

1. Settings
2. Input
3. Manhattan and OR plots
4. Diagnostic plots

```
# ------------------------------------------------------------------ #
# ------------------------------------------------------------------ #
# 1. Settings
# ------------------------------------------------------------------ #
# ------------------------------------------------------------------ #

data.dir <- "Directory to analysis results"
save.dir <- "Directory to save plot"

# ------------------------------------------------------------------ #
# ------------------------------------------------------------------ #
# 2. Input
# ------------------------------------------------------------------ #
# ------------------------------------------------------------------ #

# Read results
titel <- paste(data.dir, "/Regression_komplErg_Rez.csv", sep="")
tab.rezessiv <- read.csv2(titel)

# Titles to save plots
titel.manhattan.or <- paste(save.dir,
  "/Manhattan_OR_Plots_Rezessiv.png", sep="")
titel.diagnostics <- paste(save.dir,
  "/Cook_rs7519458_rs2500262_Rezessiv.png", sep="")

# ------------------------------------------------------------------ #
# ------------------------------------------------------------------ #
# 3. Manhattan and OR plots
# ------------------------------------------------------------------ #
# ------------------------------------------------------------------ #

# Create plot as .png with resolution equal to 300dpi

# Open graphic device
bitmap(titel.manhattan.or, res=300, width=25, height=40)
```

```
# Layout of plot
  par(mar=c(5,5.5,5,1))

layout(mat=matrix(c(1,1,2,2,3,3,4,4,5,5,6,6,7,7,8,8,9,9,10,10,11,11,
  12,12), ncol=6, byrow=T), widths=rep(4, 6), heights=rep(10, 4))

#1: Row name
  plot(1,1, bty="n", col="white", xaxt="n", yaxt="n", xlab="",
    ylab="", main="", ylim=c(0,3))
  text(1,1.5,"Standard", cex=2)

#2: Manhattan
  plot(1:nrow(tab.rezessiv), -log10(tab.rezessiv$p.stand), pch=19,
    col="black", cex=2, ylim=c(0,3), xlab="SNP",
    ylab=expression(-log[10]~"(p-value)"), main="", cex.lab=2,
    cex.axis=1.8, xaxt="n", yaxt="n")
  lines(x=c(0, nrow(tab.rezessiv)+1), y=rep(-log10(0.05), 2),
    col="black", lwd=4, type="l", lty="dotted")
  axis(2, at=c(0,1,2,3), labels=c(0,1,2,3), cex.axis=1.8)

#3: estimated ORs
  plot(1:nrow(tab.rezessiv), exp(tab.rezessiv$koeff.stand), pch=19,
    col="black", cex=2, xlab="SNP",
    ylab=expression("Estimated OR ["~10^7~"]"), main="", cex.lab=2,
    cex.axis=1.8, xaxt="n", yaxt="n")
  lines(x=c(0, nrow(tab.rezessiv)+1), y=rep(1, 2), col="black",
    lwd=4, type="l", lty="dotted")
  axis(2, at=c(0,0.5,1,1.5)*10000000, labels=c(0,0.5,1.0,1.5),
    cex.axis=1.8)

#4: Row name
  plot(1,1, bty="n", col="white", xaxt="n", yaxt="n", xlab="",
    ylab="", main="", ylim=c(0,3))
  text(1,2,"Huber", cex=2)
  text(1,1,"[1.345]", cex=2)

#5: Manhattan
  plot(1:nrow(tab.rezessiv), -log10(tab.rezessiv$p.huber), pch=19,
    col="darkgrey", cex=2, ylim=c(0,3), xlab="SNP",
    ylab=expression(-log[10]~"(p-value)"), main="", cex.lab=2,
    cex.axis=1.8, cex.main=2.5, xaxt="n", yaxt="n")
  lines(x=c(0, nrow(tab.rezessiv)+1), y=rep(-log10(0.05), 2),
    col="black", lwd=4, type="l", lty="dotted")
  axis(2, at=c(0,1,2,3), labels=c(0,1,2,3), cex.axis=1.8)

#6: estimated ORs
  plot(1:nrow(tab.rezessiv), exp(tab.rezessiv$koeff.huber), pch=19,
    col="darkgrey", cex=2, ylim=c(0,12), xlab="SNP",
    ylab="Estimated OR", main="", cex.lab=2, cex.axis=1.8,
    cex.main=2.5, xaxt="n", yaxt="n")
  lines(x=c(0, nrow(tab.rezessiv)+1), y=rep(1, 2), col="black",
    lwd=4, type="l", lty="dotted")
  axis(2, at=c(0,4,8,12), labels=c(0,4,8,12), cex.axis=1.8)

#7: Row name
  plot(1,1, bty="n", col="white", xaxt="n", yaxt="n", xlab="",
```

```
    ylab="", main="", ylim=c(0,3))
  text(1,2,"Hampel", cex=2)
  text(1,1,"[(1.5, 3.5, 8)x0.9]", cex=2)

#8: Manhattan
  plot(1:nrow(tab.rezessiv), -log10(tab.rezessiv$p.hampel.09),
    pch=19, col="grey", cex=2, ylim=c(0,3), xlab="SNP",
    ylab=expression(-log[10]~"(p-value)"), main="", cex.lab=2,
    cex.axis=1.8, cex.main=2.5, xaxt="n", yaxt="n")
  lines(x=c(0, nrow(tab.rezessiv)+1), y=rep(-log10(0.05), 2),
    col="black", lwd=4, type="l", lty="dotted")
  axis(2, at=c(0,1,2,3), labels=c(0,1,2,3), cex.axis=1.8)

#9: estimated ORs
  plot(1:nrow(tab.rezessiv), exp(tab.rezessiv$koeff.hampel.09),
    pch=19, col="grey", cex=2, ylim=c(0,12), xlab="SNP",
    ylab="Estimated OR", main="", cex.lab=2, cex.axis=1.8,
    cex.main=2.5, xaxt="n", yaxt="n")
  lines(x=c(0, nrow(tab.rezessiv)+1), y=rep(1, 2), col="black",
    lwd=4, type="l", lty="dotted")
  axis(2, at=c(0,4,8,12), labels=c(0,4,8,12), cex.axis=1.8)

#10: Row name
  plot(1,1, bty="n", col="white", xaxt="n", yaxt="n", xlab="",
    ylab="", main="", ylim=c(0,3))
  text(1,2,"Hampel", cex=2)
  text(1,1,"[(2, 4, 8)x0.7]", cex=2)

#11: Manhattan
  plot(1:nrow(tab.rezessiv), -log10(tab.rezessiv$p.hampel.06),
    pch=19, col="lightgrey", cex=2, ylim=c(0,3), xlab="SNP",
    ylab=expression(-log[10]~"(p-value)"), main="", cex.lab=2,
    cex.axis=1.8, cex.main=2.5, xaxt="n", yaxt="n")
  lines(x=c(0, nrow(tab.rezessiv)+1), y=rep(-log10(0.05), 2),
    col="black", lwd=4, type="l", lty="dotted")
  axis(2, at=c(0,1,2,3), labels=c(0,1,2,3), cex.axis=1.8)

#11: estimated ORs
  plot(1:nrow(tab.rezessiv), exp(tab.rezessiv$koeff.hampel.06),
    pch=19, col="lightgrey", cex=2, ylim=c(0,12), xlab="SNP",
    ylab="Estimated OR", main="", cex.lab=2, cex.axis=1.8,
    cex.main=2.5, xaxt="n", yaxt="n")
  lines(x=c(0, nrow(tab.rezessiv)+1), y=rep(1, 2), col="black",
    lwd=4, type="l", lty="dotted")
  axis(2, at=c(0,4,8,12), labels=c(0,4,8,12), cex.axis=1.8)

# close graphic device
dev.off()

# ------------------------------------------------------------------ #
# ------------------------------------------------------------------ #
# 4. Diagnostic plots
# ------------------------------------------------------------------ #
# ------------------------------------------------------------------ #

# for rs2500262 and rs2500262
```

```
# Indices of these SNPs in the analysis loop
# rs2500262
i <- 155

# rs7519458
i <- 230

# Save the standard logistic regression models for these SNPs
zwischen.rs7519458 <-
  "standard logistic regression model for SNP rs7519458"
zwischen.rs2500262 <-
  "standard logistic regression model for SNP rs2500262"

# Create plot as .png with resolution equal to 300dpi

# Open graphic device
bitmap(titel.diagnostics, res=300, width=10, height=5)

# Layout
  par(mfrow=c(1,2), mar=c(5,7,5,1))

# Plots
  plot(zwischen.rs7519458, which=4, main="rs7519458",
    caption='Observation number', sub.caption = "", lwd=2, cex=2,
    cex.axis=1.8, cex.lab=2, cex.main=2.5)
  plot(zwischen.rs2500262, which=4, main="rs2500262",
    caption='Observation number', sub.caption = "", lwd=2, cex=2,
    cex.axis=1.8, cex.lab=2, cex.main=2.5)

# Close device
dev.off()
```