```r
#
# Christoph Jüschke
# IMBA - Institute of Molecular Biotechnology. All rights reserved.
#
#
# executing the script:
# > R --slave --args "file" "genotype" < fit_parameter.R
#
# e. g.:
# R --slave --args "angles_template.csv" "MP_E135_ki" < fit_parameter.R

set.seed(1)

cmd_args = commandArgs(T)

file = cmd_args[1]
geno = cmd_args[2]

res = read.table(file, sep="\t", as.is=T, header=T)
sigma = c(1,1)

n = 50000

C_vb = array(data=c(rnorm(n), rnorm(n), rnorm(n)),
            dim=c(n, 3),
            dimnames=list(cell=1:n, coord=c("x", "y", "z")))
C_v = C_vb
P_n = array(rep(c(0,0,1), n), dim=c(3,n), dimnames=list(coord=c("X","Y","Z"), cell=1:n))

data = res[res$genotype==geno, "alpha"]/180*pi
edat = ecdf(data)
dat = edat(data)

fitfun <- function(sigma) {
  C_v[,"x"] = C_vb[,"x"] * sigma[1]
  C_v[,"y"] = C_vb[,"y"] * sigma[1]
  C_v[,"z"] = C_vb[,"z"] * sigma[2]
  C_n = C_v / sqrt(apply(C_v, 1, crossprod))

  alp5 = ecdf(abs(asin(colSums(P_n * t(C_n)))))
  sum((alp5(data)-dat)^2)
}

fit = optim(sigma, fitfun, method="L-BFGS-B", lower=c(0,0), upper=c(1,1),control = list(trace=2))
sigma = fit$par

C_v = array(data=c(rnorm(n, 0, sigma[1]), rnorm(n, 0, sigma[1]), rnorm(n, 0, sigma[2])),
            dim=c(n, 3),
            dimnames=list(cell=1:n, coord=c("x", "y", "z")))
C_n = C_v / sqrt(apply(C_v, 1, crossprod))

alp5 = abs(asin(colSums(P_n * t(C_n))))


pdf(file=paste("distr_",geno,".pdf", sep=""), width=3.5, height=3.5)
par(mar=c(3,3,0,0)+0.1, las=1, mgp=c(2,0.7,0))
plot(0:180/2, sin(0:180/2/180*pi), xlim=c(0,90), type="l", col="blue", lwd=1, axes=F,
xlab=expression("angle"~alpha~"(degree)"), ylab="fraction (%)")

plot.ecdf(res[res$genotype==geno, "alpha"], pch=20, verticals=T, add=T, col.01line = NULL)

pl = ecdf(alp5*180/pi)
lines(0:180/2, pl(0:180/2), col="darkred", lwd=2)

ks = ks.test(alp5,data)
legend(x="bottomright", col=c("darkred",NA,NA), lwd=2, bty="n",
legend=sapply(c(bquote(lambda[v] == .(round(-log(sigma[1]),2))),
                bquote(lambda[h] == .(round(-log(sigma[2]),2))),
                bquote(D[KS] == .(round(ks$statistic,3)))), as.expression))
axis(1, at=0:6*15)
axis(2, at=0:5/5, labels=0:5*20)
dev.off()

ks
print(c(bquote(lambda[v] == .(round(-log(sigma[1]),2))),
                bquote(lambda[h] == .(round(-log(sigma[2]),2))),
                bquote(D[KS] == .(round(ks$statistic,3)))))
```