Penalised regression with multiple sources of prior effects

Armin Rauschenberger [1,*], Zied Landoulsi [1],
Mark A. van de Wiel [2,3,†], Enrico Glaab [1,†]

11 November 2022

Initialisation

Choose the directory containing the sub-directories "data", "results" and "manuscript".

- The directory "data" should contain the file "data/pone.0181468.s001.csv" for the second external application and the files "vcf_with_pvalue.tab" and "LuxPark_pheno.txt" for the internal application. It will also contain the file "app_int_data.RData" for the internal application.
- The directory "results" will contain the files "sim_int.RData" and "sim_ext.RData" for the external and internal simulation, the file "app_grridge.RData" for the first external application, the file "app_fwelnet.RData" for the second external application, and the file "app_int.RData" for the internal application.
- The directory "manuscript" will contain the files "table_int.tex" and "table_ext.tex" for the internal and external simulations, the file "figure_example.pdf" for the methods section, the file "figure_ext.pdf" for the external applications, and the file "figure_int.pdf" for the internal application.

```
rm(list=ls())
dir <- "~/Desktop/transreg/"
setwd(dir)
if(!all(c("data","results","manuscript") %in% dir())){
 stop("Missing folders!")
}
knitr::opts_chunk$set(eval=TRUE,echo=TRUE)
```

Install missing R packages from CRAN and GitHub. Note that `ecpc` is and `transreg` will also be available on CRAN. For package versions, see session information at the end of this document and in the text files associated with each R data file.

```
pkgs <- c("devtools","palasso","glmtrans","xtable")
utils::install.packages(setdiff(pkgs,rownames(installed.packages())))
pkgs <- c("kjytay/fwelnet","Mirrelijn/ecpc/Rpackage","rauschenberger/transreg")
remotes::install_github(pkgs)
rm(pkgs)
```

Methods

Generate figure for methods section.

```
#<<init>>
grDevices::pdf(file=paste0(dir,"manuscript/figure_example.pdf"),width=8,height=5,pointsize=15)

set.seed(1)
n <- 200; p <- 500
X <- matrix(stats::rnorm(n*p),nrow=n,ncol=p)
temp <- stats::rnorm(p)
range <- stats::qnorm(p=c(0.01,0.99))
temp[temp<range[1]] <- range[1]
temp[temp>range[2]] <- range[2]

beta <- list()
beta$ident <- temp
beta$sqrt <- sign(temp)*sqrt(abs(temp))
beta$quad <- sign(temp)*abs(temp)^2
beta$trunc <- ifelse(temp<=0,0,temp)
beta$step <- ifelse(temp<=1,0,1)
beta$combn <- ifelse(temp<0,sign(temp)*sqrt(abs(temp)),sign(temp)*abs(temp)^2)

graphics::par(mfrow=c(2,3),mar=c(3,3,0.5,0.5))
for(i in seq_along(beta)){

  prior <- matrix(temp,ncol=1)
  eta <- X %*% beta[[i]]
  y <- stats::rnorm(n=n,mean=eta,sd=sd(eta))
  a <- transreg:::.exp.multiple(y=y,X=X,prior=prior,family="gaussian",select=FALSE)
  b <- transreg:::.iso.fast.single(y=y,X=X,prior=prior,family="gaussian")

  graphics::plot.new()
  graphics::plot.window(xlim=range(prior,-prior),ylim=range(a$beta,b$beta))
  graphics::axis(side=1)
  graphics::axis(side=2)
  graphics::abline(h=0,lty=2,col="grey")
  graphics::abline(v=0,lty=2,col="grey")
  graphics::box()
  graphics::title(xlab=expression(z),ylab=expression(gamma),line=2)
  graphics::points(x=prior,y=a$beta,col="red",cex=0.7)
  graphics::points(x=prior,y=b$beta,col="blue",cex=0.7)
  graphics::lines(x=prior[order(prior)],y=beta[[i]][order(prior)],lwd=1.2)
  graphics::legend(x="topleft",legend=paste0("(",i,")"),bty="n",x.intersp=0)
}

grDevices::dev.off()
```

```
## pdf
##   2
```

Simulations

Perform internal and external simulation study. **Execution time: several minutes.**

```r
#<<init>>

# - - - modify glmtrans::models function - - -
glmtrans.models <- glmtrans::models
string <- base::deparse(glmtrans.models)
# return target beta
string <- gsub(pattern="list\\(x \\= NULL, y \\= NULL\\)",
               replacement="list(x = NULL, y = NULL, beta = wk)",
               x=string)
# return source beta
string <- gsub(pattern="list\\(x \\= x, y \\= y\\)",
               replacement="list(x = x, y = y, beta = wk)",
               x=string)
glmtrans.models <- eval(parse(text=string))
rm(string)
# - - - - - - - - - - - - - - - - - - - - - - -

for(mode in c("ext","int")){

    # simulation setting
    if(mode=="ext"){
      frame <- expand.grid(Ka=as.integer(c(1,3,5)),
                           K=as.integer(5),
                           h=as.integer(c(5,250)),
                           alpha=as.integer(c(0,1)),
                           family=c("gaussian","binomial"))
      frame$seed <- rep(1:4,each=6)
    } else if(mode=="int"){
      frame <- expand.grid(rho.x=c(0.95,0.99),
                           rho.beta=c(0.70,0.85,0.99),
                           alpha=as.integer(c(0,1)),
                           family=c("gaussian","binomial"))
      frame$seed <- 1:24
    }
    frame$family <- as.character(frame$family)
    frame[,c("cor.x","cor.beta","mean","glmnet","glmtrans","transreg")] <- NA
    p <- 1000; n0 <- 100; n1 <- 10000
    n.target <- n0+n1
    foldid.ext <- rep(c(0,1),times=c(n0,n1))

    for(iter in seq_len(nrow(frame))){

      if(!is.na(frame$seed[iter])){set.seed(frame$seed[iter])}
      #alpha <- frame$alpha[iter]
      #family <- as.character(frame$family[iter])
      #rho.beta <- frame$rho.beta[iter]

      # data simulation
      if(mode=="ext"){
        message("Using external simulation study!")
        s <- ifelse(frame$alpha[iter]==0,50,15)
        data <- glmtrans.models(family=frame$family[iter],type="all",
                                p=p,n.target=n.target,s=s,
```

```
                                Ka=frame$Ka[iter],K=frame$K[iter],h=frame$h[iter])
        target <- data$target
        source <- data$source
        beta <- cbind(sapply(data$source,function(x) x$beta),data$target$beta)
      } else if(mode=="int"){
        message("Using internal simulation study!")
        prop <- ifelse(frame$alpha[iter]==0,0.2,0.05)
        data <- transreg:::simulate(p=p,n.target=n.target,family=frame$family[iter],
                                prop=prop,rho.beta=frame$rho.beta[iter],w=0.5,
                                rho.x=frame$rho.x[iter],k=3,exp=c(1,2,0.5),
                                trans=c(FALSE,TRUE,TRUE))
        target <- data$target
        source <- data$source
        beta <- data$beta
      }

      # correlation
      temp <- abs(stats::cor(data$target$x,method="pearson"))
      temp[lower.tri(temp,diag=TRUE)] <- NA
      frame$cor.x[iter] <- mean(temp,na.rm=TRUE)
      temp <- abs(stats::cor(beta,method="pearson"))
      temp[lower.tri(temp,diag=TRUE)] <- NA
      frame$cor.beta[iter] <- max(temp[,ncol(temp)],na.rm=TRUE)

      # predictive performance
      loss <- transreg:::compare(target=target,source=source,
                          family=frame$family[iter],alpha=frame$alpha[iter],
                          foldid.ext=foldid.ext,nfolds.ext=1,
                          scale=c("exp","iso"),
                          sign=FALSE,switch=FALSE,select=TRUE,alpha.prior=NULL,
                          seed=frame$seed[iter])
      frame[iter,names(loss$deviance)] <- loss$deviance
    }
    save(frame,file=paste0(dir,"results/sim_",mode,".RData")))
}
writeLines(text=capture.output(utils::sessionInfo(),cat("\n"),
      sessioninfo::session_info()),con="results/info_sim.txt")

#colMeans(frame[,paste0("transreg.",c("exp","iso"),".sta")],na.rm=TRUE)
#colMeans(frame[,paste0("transreg.",c("exp","iso"),".sim")],na.rm=TRUE)

#stats::wilcox.test(x=frame$glmtrans,y=frame$transreg,alternative="greater")

#load("results/sim_int.RData")
#load("results/sim_ext.RData")

#data <- frame[c("mean","glmnet","glmtrans","stage","transreg")]/frame$mean
#cbind(frame$family,frame$alpha,frame$K,round(100*data,digits=0))
#colMeans(data)
#apply(data,2,median)
```

Generate LaTeX tables for external and internal simulation. **Requires execution of previous chunk.**

```r
.table <- function(loss,info=NULL,hline=NULL,pvalue=NULL,caption="",file=""){
  loss[is.na(loss)] <- Inf

  if(nrow(loss)>5){
  quote <- NA*loss
  quote[1,] <- FALSE
  quote[2,] <- loss[1,]==loss[2,] & loss[1,]==loss[3,] & loss[1,]==loss[4,]
  quote[3,] <- loss[2,]==loss[3,] & loss[3,]==loss[4,] & loss[4,]==loss[5,]
  for(i in 4:(nrow(loss)-1)){
    quote[i,] <- (loss[i,]==loss[i-1,] & loss[i,]==loss[i-2,] & loss[i,]==loss[i-3,]) |
      (loss[i,]==loss[i-1,] & loss[i,]==loss[i+1,] & loss[i,]==loss[i+2,])
  }
  quote[nrow(quote),] <- loss[nrow(quote),]==loss[nrow(quote)-1,] &
          loss[nrow(quote),]==loss[nrow(quote)-2,] &
          loss[nrow(quote),]==loss[nrow(quote)-3,]
  }

  loss <- round(loss,digits=3)

  min <- cbind(seq_len(nrow(loss)),apply(loss,1,which.min))
  pos <- loss >= 0
  leq <- loss >= loss[,"glmnet"]
  loss <- format(round(loss,digits=3),digits=3)
  if(nrow(loss)>5){loss[quote==1] <- "\""}
  loss[leq] <- paste0("\\textcolor{gray}{",loss[leq],"}")
  loss[min] <- paste0("\\underline{",loss[min],"}")
  loss[pos] <- paste0("~",loss[pos])
  colnames(loss) <- paste0("\\texttt{",colnames(loss),"}")
  rownames(loss) <- paste0("\\textsc{",tolower(rownames(loss)),"}")

  if(is.null(info)){
    table <- loss
    align <- paste0("|r|",paste0(rep("r",times=ncol(loss)),collapse=""),"|")
    include.rownames <- TRUE
  } else {
    info[is.na(info)] <- "-"
    table <- cbind(info,loss)
    align <- paste0("|r|",paste0(rep("r",times=ncol(info)),collapse=""),"|",
                    paste0(rep("r",times=ncol(loss)),collapse=""),"|")
    include.rownames <- FALSE
  }
  xtable <- xtable::xtable(x=table,caption=caption,align=align)
  xtable::print.xtable(x=xtable,
                       sanitize.text.function=identity,
                       include.rownames=include.rownames,
                       floating=TRUE,
                       comment=FALSE,
                       hline.after=c(-1,0,hline,nrow(table)),
                       caption.placement="top",
                       file=file)
}

for(mode in c("ext","int")){
```

```
    file <- paste0(dir,"results/sim_",mode,".RData")
    if(!file.exists(file)){warning("Missing file ",file,".");next}
    load(file)
    cond <- colnames(frame) %in% c("Ka","h","rho.x","rho.beta","cor.x","cor.beta","cor.t","alpha","famil
    info <- frame[,cond]
    info <- info[,colMeans(is.na(info))<1]
    colnames(frame) <- gsub(pattern="transreg.",replacement="",x=colnames(frame))
    #colnames(frame) <- gsub(pattern="_",replacement=".",x=colnames(frame))
    names <- c("mean","glmnet","glmtrans","exp.sta","exp.sim","iso.sta","iso.sim")
    loss <- frame[,names]
    loss <- round(100*loss/loss$mean,digits=1)
    loss <- loss[,!colnames(loss) %in% "mean"]
    colnames(info) <- sapply(colnames(info),function(x) switch(x,"Ka"="$K_a$","K"="$K$","h"="$h$","alpha
    external <- "number of transferable source data sets ($K_a$), differences between source and target
    internal <- "correlation \\textcolor{blue}{parameter for} features ($\\rho_x$), correlation \\textco
    caption <- paste0("Testing loss in ",mode,"ernal simulation\\textcolor{blue}{, as a percentage of th
    .table(info=info,loss=loss,caption=caption,file=paste0(dir,"manuscript/table_",mode,".tex"))
}
```

External applications

Perform first external application. **Execution time: several minutes.**

```
#<<init>>
data(dataVerlaat,package="GRridge")

target <- list()
target$y <- as.numeric(as.factor(respVerlaat))-1
target$x <- t(datcenVerlaat)

z <- -log10(pvalFarkas) # ecpc and fwelnet
prior <- sign(diffmeanFarkas)*(-log10(pvalFarkas)) # transreg

loss <- list()
for(i in 1:10){
  cat("---",i,"---\n")
  loss[[i]] <- transreg:::compare(target=target,prior=prior,z=as.matrix(z,ncol=1),
                                  family="binomial",alpha=0,scale=c("exp","iso"),sign=FALSE,switch=FA
}
save(loss,file=paste0(dir,"results/app_grridge.RData"))
writeLines(text=capture.output(utils::sessionInfo(),cat("\n"),
      sessioninfo::session_info()),con="results/info_app_grridge.txt")

load(paste0(dir,"results/app_grridge.RData"),verbose=TRUE)
table <- as.data.frame(t(sapply(loss,function(x) x$deviance)))
table <- (table-table$glmnet)/table$glmnet
table <- table[,c("glmnet","transreg.exp.sta","transreg.exp.sim","transreg.iso.sta","transreg.iso.sim",
sapply(table[,-1],function(x) sum(x<table$glmnet))
round(100*colMeans(table[,-1]),digits=2)
```

Perform second external application. **Execution time: several minutes.**

```r
table <- utils::read.csv("data/pone.0181468.s001.csv",header=TRUE,skip=3)

extract <- function(cond,y,X,id){
  if(length(unique(c(length(cond),length(y),nrow(X),length(id))))!=1){stop("Invalid input.")}
  n <- table(id,cond)[,"TRUE"]
  y <- y[cond]
  X <- X[cond,]
  id <- id[cond]
  weights <- rep(1/n,times=n)
  ids <- unique(id)
  ys <- sapply(ids,function(x) unique(y[id==x]))
  foldid <- rep(NA,length=length(ids))
  foldid[ys==0] <- sample(rep(1:10,length.out=sum(ys==0)))
  foldid[ys==1] <- sample(rep(1:10,length.out=sum(ys==1)))
  foldid <- rep(foldid,times=n[n!=0])
  if(length(unique(c(length(y),nrow(X),length(weights),length(foldid))))!=1){
    stop("Invalid output.")
  }
  return(list(y=y,x=X,weights=weights,foldid=foldid))
}

loss <- ridge <- lasso <- list()
for(i in 1:10){
  cat("---",i,"---\n")
  set.seed(i)

  y <- table$LatePE
  X <- as.matrix(table[,grepl(pattern="SL",x=colnames(table))])
  X <- scale(X)

  min <- sapply(unique(table$ID),function(i) min(table$GA[table$ID==i]))
  max <- sapply(unique(table$ID),function(i) max(table$GA[table$ID==i]))

  limit <- 20
  group <- stats::rbinom(n=max(table$ID),size=1,prob=0.5)
  source.id <- which(group==0 | min > limit)
  target.id <- which(group==1 & min <= limit)
  if(any(!table$ID %in% c(source.id,target.id))){stop()}
  if(any(!c(source.id,target.id) %in% table$ID)){stop()}
  if(any(duplicated(c(source.id,target.id)))){stop()}

  source <- list()
  source[[1]] <- extract(cond=(table$ID %in% source.id) & (table$GA<=limit),y=y,X=X,id=table$ID)
  source[[2]] <- extract(cond=(table$ID %in% source.id),y=y,X=X,id=table$ID)

  prior <- z <- matrix(NA,nrow=ncol(X),ncol=length(source))
  for(j in seq_along(source)){
    base <- glmnet::cv.glmnet(y=source[[j]]$y,x=source[[j]]$x,
                              family="binomial",alpha=0,
                              weights=source[[j]]$weights,
                              foldid=source[[j]]$foldid)
    prior[,j] <- coef(base,s="lambda.min")[-1]
    z[,j] <- abs(coef(base,s="lambda.min")[-1])
```

```
  }

  target <- list()
  target$y <- sapply(target.id,function(i) unique(y[table$ID==i]))
  target$x <- t(sapply(target.id,function(i) X[table$ID==i & table$GA==min(table$GA[table$ID==i]),]))

  loss[[i]] <- transreg:::compare(target=target,prior=prior,family="binomial",alpha=0,scale=c("exp","iso
}
save(loss,file=paste0(dir,"results/app_fwelnet.RData"))
writeLines(text=capture.output(utils::sessionInfo(),cat("\n"),
      sessioninfo::session_info()),con="results/info_app_fwelnet.txt")

load(paste0(dir,"results/app_fwelnet0.RData"))
table <- as.data.frame(t(sapply(loss,function(x) x$deviance)))
table <- (table-table$glmnet)/table$glmnet
table <- table[,c("glmnet","transreg.exp.sta","transreg.exp.sim","transreg.iso.sta","transreg.iso.sim",
sapply(table[,-1],function(x) sum(x<table$glmnet))
round(100*colMeans(table[,-1]),digits=2)
```

Generate figure for both external applications. **Requires execution of previous two chunks.**

```
grDevices::pdf(file=paste0(dir,"manuscript/figure_int.pdf"),width=8,height=6,pointsize=15)

graphics::par(mfrow=c(2,1),mar=c(2.5,3.5,0.5,0.5))

for(k in c("grridge0","fwelnet0")){
  file <- paste0(dir,"results/app_",k,".RData")
  if(!file.exists(file)){plot.new();next}
  load(file)
  loss <- as.data.frame(t(sapply(loss,function(x) x$deviance)))
  colnames(loss) <- gsub(pattern="transreg.",replacement="",x=colnames(loss))
  #colnames(loss) <- gsub(pattern="_",replacement=".",x=colnames(loss))
  loss <- 100*(loss-loss$glmnet)/loss$glmnet

  temp <- c("exp.sta","exp.sim","iso.sta","iso.sim")
  name <- c("fwelnet","ecpc",temp)
  graphics::plot.new()
  graphics::plot.window(xlim=c(0.5,length(name)+0.5),ylim=range(loss,na.rm=TRUE))
  #graphics::abline(h=median(loss$mean),lty=2,col="grey")
  graphics::abline(h=0,lty=2,col="grey")
  graphics::axis(side=1,at=seq_along(name),labels=name,cex.axis=0.7)
  if(grepl(pattern="grridge",x=k)){at <- seq(from=-10,to=10,by=5)}
  if(grepl(pattern="fwelnet",x=k)){at <- seq(from=-20,to=20,by=10)}
  labels <- ifelse(at==0,"0%",ifelse(at<0,paste0(at,"%"),paste0("+",at,"%")))
  graphics::axis(side=2,cex.axis=0.7,at=at,labels=labels)
  graphics::title(ylab="change in loss",line=2.5,cex.lab=0.7)
  graphics::box()
  for(i in seq_along(name)){
    palasso:::.boxplot(loss[,name[i]],at=i,invert=FALSE)
    graphics::points(x=i,y=mean(loss[,name[i]]),pch=22,col="white",bg="black",cex=0.7)
  }
}
```

```
grDevices::dev.off()
```

```
## pdf
##   2
```

Internal application

Perform internal application. **Execution time: several hours.**

```
geno <- read.table("data/vcf_with_pvalue.tab",header=TRUE)

switch <- ifelse(geno$REF==geno$A1_gwas & geno$ALT==geno$A2_gwas,1,
                 ifelse(geno$REF==geno$A2_gwas & geno$ALT==geno$A1_gwas,-1,0))
#prior <- log10(geno$p_value)*sign(geno$beta)*switch # pseudo effect sizes
prior <- -geno$beta*switch # original effect sizes
pvalue <- geno$p_value

X <- geno[,grepl(pattern="ND",colnames(geno))]
X[X=="0/0"] <- 0
X[X=="0/1"] <- 1
X[X=="1/1"] <- 1
X <- sapply(X,as.numeric)
X <- t(X)

pheno <- read.delim("data/LuxPark_pheno.txt",sep=" ",header=FALSE)
y <- ifelse(pheno$V2==1,0,ifelse(pheno$V2==2,1,NA)); names(y) <- pheno$V1

names <- intersect(names(y[!is.na(y)]),rownames(X))
X <- X[names,]; y <- y[names]

cor <- as.numeric(stats::cor(y,X))
graphics::plot(x=prior,y=cor)

save(y,X,prior,pvalue,switch,file=paste0(dir,"data/app_int_data.RData"))

# descriptive statistics
sum(p.adjust(pvalue,method="BH")<=0.05)
sum(p.adjust(pvalue,method="holm")<=0.05)
mean(pvalue<=0.05)
dim(X)
table(y)

load(paste0(dir,"data/app_int_data.RData"))
power <- seq(from=-2,to=-10,by=-1)
cutoff <- 5*10^power
frame <- expand.grid(cutoff=cutoff,alpha=0:1,seed=1:10,count=NA)

loss <- list()
for(i in seq_len(nrow(frame))){
  cat("--- i =",i,"---","\n")
  set.seed(frame$seed[i])
```

```
  foldid <- transreg:::.folds(y=y,nfolds.ext=10,nfolds.int=10)
  cond <- switch!=0 & pvalue < frame$cutoff[i]
  loss[[i]] <- transreg:::compare(target=list(y=y,x=X[,cond]),prior=prior[cond],family="binomial",alpha=
  frame$count[i] <- sum(cond)
}
save(frame,loss,file=paste0(dir,"results/app_int.RData"))
writeLines(text=capture.output(utils::sessionInfo(),cat("\n"),
      sessioninfo::session_info()),con="results/info_app_int.txt")

# use repeated 10-fold CV (TO DO)
# modify function cv.transfer so that it accepts named list with "y and X" (not only "y and x") (TO DO)
```

Generate figure for internal application. **Requires execution of previous chunk.**

```
#<<init>>
grDevices::pdf(file=paste0(dir,"manuscript/figure_int.pdf"),width=8,height=6,pointsize=15)

load(paste0(dir,"results/app_int.RData")) # TO DO: change this line
frame <- frame[seq_along(loss),colnames(frame)!="seed"]
loss <- as.data.frame(t(sapply(loss,function(x) x$auc))) # temporary AUC
names(loss)[names(loss)=="prs"] <- "naive" # TO DO: delete this line

table <- lapply(loss,function(x) tapply(X=x,INDEX=list(frame$cutoff,frame$alpha),FUN=function(x) mean(x)

cutoff <- unique(frame$cutoff)
number <- unique(frame$count)

graphics::par(mfrow=c(2,2),mar=c(3,1.8,1.0,0.9))
for(scale in c("exp","iso")){
  for(alpha in c("0","1")){
    graphics::plot.new()
    graphics::plot.window(xlim=range(log(cutoff)),ylim=range(table))
    graphics::box()
    graphics::title(main=paste(ifelse(alpha==0,"ridge",ifelse(alpha==1,"lasso",NA)),"-",scale),cex.main=
    on <- rep(c(TRUE,FALSE),length.out=length(cutoff))
    graphics::axis(side=1,at=log(cutoff),labels=rep("",times=length(on)),cex.axis=0.7)
    graphics::axis(side=1,at=log(cutoff)[on],labels=paste0(cutoff[on],"\n","(",number[on],")"),cex.axis=

    graphics::axis(side=2,cex.axis=0.7)
    #graphics::abline(h=unique(table[["mean"]][,alpha]),col="grey",lty=2)
    graphics::abline(h=0.5,col="grey",lty=2)
    for(i in 1:3){
    for(method in c("glmnet",paste0("transreg_",scale,c("_lp","_mf")),"naive")){
      lty <- switch(method,"mean"=1,"glmnet"=1,"transreg_exp_lp"=2,"transreg_exp_mf"=2,"transreg_iso_lp
      col <- switch(method,"mean"="grey","glmnet"="black","transreg_exp_lp"=rgb(0.2,0.2,1),"transreg_is
      y <- table[[method]][,alpha]
      x <- log(as.numeric(names(y)))
      #cond <- x >= log(5e-08)
      if(i==1){graphics::lines(x=x,y=y,col=col,lty=lty)}
      if(i==2){graphics::points(x=x,y=y,col="white",pch=16)}
      if(i==3){graphics::points(x=x,y=y,col=col)}
    }
    }
```

```
  }
}

grDevices::dev.off()
```

```
## pdf
##   2
```

Session information

Reformat list of consortium members for acknowledgements: [see source].

Print session information.

```
utils::toLatex(utils::sessionInfo())
```

- R version 4.2.1 (2022-06-23), `aarch64-apple-darwin20`
- Locale: `en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8`
- Running under: `macOS Monterey 12.6`
- Matrix products: default
- BLAS: `/Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/lib/libRblas.0.dylib`
- LAPACK: `/Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/lib/libRlapack.dylib`
- Base packages: base, datasets, graphics, grDevices, methods, stats, utils
- Loaded via a namespace (and not attached): cli 3.4.1, codetools 0.2-18, compiler 4.2.1, digest 0.6.30, evaluate 0.16, fastmap 1.1.0, foreach 1.5.2, glmnet 4.1-4, grid 4.2.1, htmltools 0.5.3, iterators 1.0.14, knitr 1.40, lattice 0.20-45, magrittr 2.0.3, Matrix 1.4-1, palasso 0.0.8, Rcpp 1.0.9, rlang 1.0.6, rmarkdown 2.16, rstudioapi 0.14, shape 1.4.6, splines 4.2.1, starnet 0.0.6, stringi 1.7.8, stringr 1.4.1, survival 3.3-1, tools 4.2.1, transreg 0.0.1, xfun 0.33, xtable 1.8-4, yaml 2.3.5