```
########################################################################
########################################################################
########################################################################
########################################################################
####################
## can find this in customapsr.R
## Start of modified apsrtable function
########## Begin real function
########################################################################
####################
apsr <- function (models,
                  se=c("robust", "vcov", "both"),
                  ## se=c("vcov"),
                                        # model.names can be shorter,
others numbered;
                                        # numbers start at value of
model.counter
                  model.names=NULL, model.counter=1, digits=2,
                                        # stars="default" prints R
default
                                        # this function default is one
star at .05
                  stars=1, lev=0.05,
                  align=c("left","center","right"),
                  order=c("lr","rl","longest"),
                  notes=list(se.note(), stars.note() ),
                  omitcoef=NULL,coef.names=NULL,
                  coef.rows=2,
                  multicolumn.align=c("center","left","right"),
                  col.hspace=NULL,
                  Sweave=FALSE, float="table",
                  tsize=1, label=NULL,caption=NULL
                  )
{
    x <- list()
    signif.stars <- TRUE
    order <- match.arg(order,c("lr","rl","longest"))
    opts <- match.call(expand.dots=FALSE)
    ## se <- match.arg(se,c("robust","vcov","both"))
    se <- "vcov"
    align <- substr(align,1,1)
    align <- match.arg(align,c("l","c","r"))
    multicolumn.align <- match.arg(substr(multicolumn.align,1,1),
                                   c("c","l","r"))
    adigits <- ifelse(align=="c",
                      -1,
                      digits)
    models <- models ## modified this line to have main argument be a
list
    nmodels <- length(models)
```

```
    if(!Sweave){
        x <- paste(x,"\\begin{",float,"}[!ht]\n\\caption{", caption,"}
\n\\label{", label,"}\n\\resizebox{", tsize, "\\textwidth}{!}
{",sep="")
    }
    ## used to multiply later for column counts
    coef.cols <- ifelse(coef.rows==2, 1, 2)
    ## Two default behaviors for col.hspace:
    ##  if in two-column-per-model mode, default to 1em
    ##  otherwise, empty. If not "", add some latex around it.
    if (is.null(col.hspace)) {
        col.hspace <- ifelse(coef.cols==1,"",
                             "1em")
    }
    if(col.hspace != "") {
        col.hspace <- paste("@{\\hspace{",col.hspace,"}}",sep="")
    }
    ## get the summaries for the objects
    model.summaries <- lapply(models,
                              function(x) {
                                  s <- try(apsrtableSummary(x),
silent=TRUE)
                                  if (inherits(s, "try-error")) {
                                      s <- list()
                                      s$coefficients <- summary(x)
$coef
                                      s$residuals <- resid(x)
                                      s$rank <- as.vector(summary(x)
$ngrps)
                                      s$terms <- terms(x)
                                      s$call <- x@call
                                      se.mod <- (vcov(x))@factors
$correlation@sd
                                      s$se.mod <- se.mod
                                      s$AICtab <- summary(x)$AICtab
                                      s$n <- dim(x@frame)[1]
                                      between.group.var <-
as.numeric(VarCorr(x)[1])
                                      within.group.var <-
sqrt(attributes(VarCorr(x))$sc)
                                      ## between.group.var <-
as.numeric((summary(x)$REmat[, "Variance"])[1])
                                      ## within.group.var <-
as.numeric((summary(x)$REmat[, "Variance"])[2])
                                      s$icc <- between.group.var/
(between.group.var + within.group.var)
                                      ## s$dims <- x$dims
                                      s$loglik <-
as.numeric(logLik(x))
                                      class(s) <- "glm"
```

```
                                    }
                                    ## if(!is.null(se.mod) && se !=
"vcov") {
                                    est <- as.numeric(s$coefficients[,
1])
                                    if(class(se.mod) == "matrix") {
                                        se.mod <- sqrt(diag(se.mod))
                                    }
                                    s$coefficients[,3] <- tval <- est /
s$se.mod
                                    s$coefficients <- cbind(s
$coefficients, (2 * pt(abs(tval),

length(s$residuals) - s$rank,

lower.tail=FALSE)))
                                    colnames(s$coefficients)[4] <-
"Pr(>|t|)"
                                          #}
                                    return(s)
                                })
    ## Set up the model names
    ## If there's a vector of names, use that, or as many as there are
    ## and either all or the remainder.
    ## Optionally, model.number.start allows you to resetcounter
    ## TO DO: allow model "name" attribute to be used
    ##        but overridden by vector here.
    if (is.null(model.names)) {
        m.first = model.counter; m.last=m.first+(nmodels-1)
        model.names=paste("Model", m.first:m.last)
    } else if (!is.null(model.names) && (length(model.names) <
nmodels) ) {
        m.first = length(model.names)+1
        model.names=c(model.names, paste( "Model", m.first:nmodels))
    }
    ## get and order the coefficient names from all models
    coefnames <- orderCoef(model.summaries, order=order)
    ## mark those to omit from the output
    incl <- rep(TRUE,length(coefnames))
    names(incl) <- coefnames
    if(!is.null(omitcoef)) {
        incl[omitcoef] <- FALSE
    }
    ## now figure out position of each coef in each model
    model.summaries <- coefPosition(model.summaries, coefnames)
    ## Now that the coef name matching is done, switch to pretty names
    ## if they are supplied.
    if(!is.null(coef.names)) {
        if(length(coef.names) != sum(incl)) {
            warning("Supplied coef.names not the same length as
```

```
output. Check automatic names before supplying 'pretty' names.\n") }
        coefnames[incl] <- coef.names}
    out.table <- lapply(model.summaries, function(x){
        var.pos <- attr(x,"var.pos")
        model.out <- model.se.out <- star.out <-
rep(NA,length(coefnames))
        model.out[var.pos] <- x$coefficients[,1]
        star.out[var.pos] <- apsrStars(x
$coefficients,stars=stars,lev=lev,signif.stars=TRUE)
        model.out <- ifelse(!is.na(model.out),

paste(formatC(model.out,digits=digits,format="f"),
                                star.out),
                            "")
        model.se.out[var.pos] <- x$coefficients[,2]
        se <- "vcov"
        if( !is.null(x$se.mod) & se %in% c("robust","both") ) {
            model.se.out[var.pos] <- x$se.mod
        }
        model.se.out <- ifelse(!is.na(model.se.out),
                               paste("(",
                                     formatC(model.se.out,
                                             digits=digits,
                                             format="f"),
                                     ")",sep=""),
                               "")
        if(se =="both" && !is.null(x$se.mod)){
            model.se.out[var.pos] <- ifelse(model.se.out != "",
                                            paste(model.se.out," [",
                                                  formatC(x
$coefficients[,2],

digits=digits,
                                                                format="f"),
                                                  "]",sep=""),
                                            "")
        }
        if(coef.rows==2) {
            ## Create two side by side columns and mesh them together
            model.out <- rep(model.out[incl], each=2)
            model.se.out <- rep(model.se.out[incl], each=2)
            pos.se <- (1:length(model.out))[(1:length(model.out) %%
2==0)]
            model.out[pos.se] <- model.se.out[pos.se]
            ## Add a new model info attribute to the model's output
entry
            ## To change modelInfo for a given model, change the
method for it
            ## see ?modelInfo, it is reasonably well documented.
        } else {
```

```
            ## two columns per model
            model.out <- model.out[incl]
            model.out <- cbind(model.out, model.se.out[incl])
        }
        attr(model.out,"model.info") <- modelInfo(x)
        return(model.out)
    })
    out.matrix <- matrix(unlist(out.table),
                         length(coefnames[incl])*coef.rows,
                         nmodels*coef.cols)
    out.matrix <- cbind(rep(coefnames[incl],each=coef.rows),
out.matrix)
    if(coef.rows==2) {
        out.matrix[ (row(out.matrix)[,1] %% 2 ==0) , 1] <- ""
    }
    out.info <- lapply(out.table, attr, "model.info")
    info.names <- orderCoef(out.info)
    out.info <- coefPosition( out.info, orderCoef(out.info) )
    out.info <- lapply(out.info, function(x) {
        var.pos <- attr(x,"var.pos")
        model.out <- rep("",length(info.names))
        model.out[var.pos] <- coef(x)
        return(model.out)
    } )
    out.info <- matrix(unlist(out.info), length(info.names), nmodels)
    out.info <- cbind(as.character(info.names), out.info)
    if(coef.rows==2) {
        out.matrix <- rbind(c("%",model.names ),out.matrix)
    }
    outrows <- nrow(out.matrix)
    ## This does the pretty latex formatting, where commented model
names
    ## line up with appropriately sized columns of numbers.
    ## Paul Johnson suggested a 'wide' or two column format for tables
    ## which then means model info needs to be underneath the two
    ## in a multicolumn span. But, for normal (two row, one column per
coef)
    ## format, this is extraneous markup and hard to read.
    if(coef.cols==1) {
        out.matrix <- rbind(out.matrix,out.info)
        out.matrix[,-1] <- format(out.matrix[,-1])
        out.matrix[,1] <- format(out.matrix)[,1]
        out.matrix <- apply(out.matrix, 1, paste, collapse=" & ")
        out.info <- out.matrix[ (1+outrows) : length(out.matrix) ]
        out.matrix <- out.matrix[ 1:outrows ]
    } else {
        out.matrix <- format(out.matrix)
        out.matrix <- apply(out.matrix, 1, paste, collapse=" & ")
        ## now do the out.info as multicolumn blocks
        out.info[,-1] <- format(out.info[,-1])
```

```
        out.info[,-1] <- sapply(as.matrix(out.info[,-1]), function(x)
{
            paste("\\multicolumn{",coef.cols,"}{",multicolumn.align,
                  "}{",x,"}",sep="")
        })
        out.info[,1] <- format(out.info[,1])
        out.info <- apply(out.info, 1, paste, collapse=" & ")
    }
    x <- c(x,paste("\\begin{tabular}{",
                   align,
                   paste(rep(paste("D{.}{.}{",
                                   rep(adigits,coef.cols),
                                   "}",
                                   sep="",collapse=""),nmodels),
                         collapse=col.hspace)
                   ,"}",sep=""), "\\hline \n &")
    x <- c(x, paste("", paste("\\multicolumn{",coef.cols,"}{",
                              multicolumn.align,"}{",
                              model.names,"}", collapse=" & ")  ))
    x <- c(x,"\\\\ \\hline\n")
    x <- c(x,paste(out.matrix, collapse="\\\\ \n"))
    x <- c(x,"\\\\\n")
    x <- c(x,paste(out.info, collapse="\\\\ \n"))
    ## Do notes
    ## Evaluate the notes list
    ## Switch the se to either robust or regular --
    ## Robust is the default, but if only vcov are given,
    ## quietly switch the argument.
    ## se <- ifelse((se != "vcov" &&
    ##               sum(unlist(lapply(model.summaries,
    ##                                 function(x) !is.null(x
$se.mod))) >0 ) ) ,
    ##              "robust","vcov")
    se <- "vcov"
    notes <- lapply(notes,evalq)
    x <- c(x,"\\\\ \\hline\n")
    notes <- lapply(notes, function(x) {  # eek! note coef cols was
wrong
                                          # fixed 2009-05-07 mjm
        paste("\\multicolumn{",(nmodels*coef.cols)+1,"}{l}{\
\footnotesize{", x , "}}",sep="")
    } )
    x <- c(x, paste(notes, collapse="\\\\\n"))
    x <- c(x,"\n\\end{tabular}}\n")
    if(!Sweave) { x <- c(x,paste("\\end{",float,"}\n",sep="")) }
    class(x) <- "apsrtable"
    return(x)
}
apsrStars <- function (x, digits = max(3, getOption("digits") - 2),
                       signif.stars = getOption("show.signif.stars"),
```

```
                      signif.legend = signif.stars,
                      dig.tst = max(1, min(5, digits - 1)), cs.ind =
1:k,
                      tst.ind = k + 1, zap.ind = integer(0),
                      P.values = NULL,
                      has.Pvalue = nc >= 3 && # used to be 4
                      substr(colnames(x)[nc],
                             1, 3) == "Pr(" ||
                      grep("t",colnames(x)[nc]) == TRUE,
                      eps.Pvalue = .Machine$double.eps, na.print =
"NA",
                      stars="default", lev=0.05,
                      ...)
{
    if (is.null(d <- dim(x)) || length(d) != 2)
        stop("'x' must be coefficient matrix/data frame")
    nc <- d[2]
    if (is.null(P.values)) {
        scp <- getOption("show.coef.Pvalues")
        if (!is.logical(scp) || is.na(scp)) {
            warning("option \"show.coef.Pvalues\" is invalid: assuming
TRUE")
            scp <- TRUE
        }
        P.values <- has.Pvalue && scp
    }
    else if (P.values && !has.Pvalue)
        stop("'P.values' is TRUE, but 'has.Pvalue' is not")
    if (has.Pvalue && !P.values) {
        d <- dim(xm <- data.matrix(x[, -nc, drop = FALSE]))
        nc <- nc - 1
        has.Pvalue <- FALSE
    }
    else xm <- data.matrix(x)
    k <- nc - has.Pvalue - (if (missing(tst.ind))
                            1
    else length(tst.ind))
    if (!missing(cs.ind) && length(cs.ind) > k)
        stop("wrong k / cs.ind")
    Cf <- array("", dim = d, dimnames = dimnames(xm))
    ok <- !(ina <- is.na(xm))
    if (length(cs.ind) > 0) {
        acs <- abs(coef.se <- xm[, cs.ind, drop = FALSE])
        if (any(is.finite(acs))) {
            digmin <- 1 + floor(log10(range(acs[acs != 0], na.rm =
TRUE)))
            Cf[, cs.ind] <- format(round(coef.se, max(1, digits -
                                                      digmin)), digits
= digits)
        }
```

```
    }
    if (length(tst.ind) > 0)
        Cf[, tst.ind] <- format(round(xm[, tst.ind], digits =
dig.tst),
                                digits = digits)
    if (length(zap.ind) > 0)
        Cf[, zap.ind] <- format(zapsmall(xm[, zap.ind], digits =
digits),
                                digits = digits)
    if (any(r.ind <- !((1:nc) %in% c(cs.ind, tst.ind, zap.ind,
                                     if (has.Pvalue) nc))))
        Cf[, r.ind] <- format(xm[, r.ind], digits = digits)
    okP <- if (has.Pvalue)
        ok[, -nc]
    else ok
    x1 <- Cf[okP]
    dec <- getOption("OutDec")
    if (dec != ".")
        x1 <- chartr(dec, ".", x1)
    x0 <- (xm[okP] == 0) != (as.numeric(x1) == 0)
    if (length(not.both.0 <- which(x0 & !is.na(x0)))) {
        Cf[okP][not.both.0] <- format(xm[okP][not.both.0], digits =
max(1,
                                                           digits -
1))
    }
    if (any(ina))
        Cf[ina] <- na.print
    if (P.values) {
        if (!is.logical(signif.stars) || is.na(signif.stars)) {
            warning("option \"show.signif.stars\" is invalid: assuming
TRUE")
            signif.stars <- TRUE
        }
        if (any(okP <- ok[, nc])) {
            pv <- as.vector(xm[, nc])
            Cf[okP, nc] <- format.pval(pv[okP], digits = dig.tst,
                                       eps = eps.Pvalue)
            signif.stars <- signif.stars && any(pv[okP] < 0.1)
            Signif <- ""
            if (signif.stars && stars=="default") {
                Signif <- symnum(pv, corr = FALSE, na = FALSE,
                                 cutpoints = c(0, 0.001, 0.01, 0.05,
0.1, 1),
                                 symbols = c("^{***}", "^{**}", "^*",
"^\\dagger\ ", " "))
                Cf <- cbind(Cf, format(Signif))
            }
            else if (signif.stars && stars==1) {
                Signif <- symnum(pv, corr = FALSE, na = FALSE,
```

```
                                cutpoints = c(0,lev,1),
                                symbols = c("^*"," "))
            }
            return(Signif)
        }
    }
    return()
}
## End
## define a custom glm one
setMethod("modelInfo", "glm", function(x) {
    env <- sys.parent()
    digits <- evalq(digits, env)
    model.info <- list(
                        "$N$"=formatC(x$n,format="d"),
                        Countries=formatC(x$rank,format="d"),
                        ICC=formatC(as.numeric(x
$icc),format="f",digits=2),
                        AIC=formatC(as.numeric(x
$AICtab[1]),format="f",digits=0),
                        BIC= formatC(as.numeric(x
$AICtab[2]),format="f",digits=0),
                        "$\\log L$"=formatC(as.numeric(x
$loglik),format="f",digits=0)
                        )
    class(model.info) <- "model.info"
    invisible(model.info)
})
## RULES: All according to longest model,
##        then left to right
## RESULT: union of all models' coefficient names in requested order.
orderCoef <- function(model.summaries,order="lr") {
    nmodels <- length(model.summaries)
    mlength <- sapply(model.summaries, function(x) length(coef(x)) )
    longest <- which.max(mlength) # longest model
    if(order=="rl") {
        modelorder <- nmodels:1 } else {
            modelorder <- 1:nmodels }
    if(order=="longest") {
        coefnames <-  rownames(coef(model.summaries[[longest]]))
    } else {
        coefnames <-
rownames(coef(model.summaries[[modelorder[1]]])) }
    for(i in seq_along(model.summaries)) {
        matched <- match(rownames(coef(model.summaries[[i]])),
coefnames, nomatch=0)
        unmatched <- which(is.na(matched) | matched==0)
        coefnames <- c(coefnames,
                        rownames(coef(model.summaries[[i]]))[unmatched]
                        )
```

```
    }
    return(coefnames)
}
## Append an attribute to each element containing its coefs' position
in the
## master coefficient list
coefPosition <- function(model.summaries, coefnames) {
    model.summaries <- lapply(model.summaries, function(x) {
        pos <- match(rownames(coef(x)), coefnames)
        attr(x,"var.pos") <- pos
        return(x)
    })
    return(model.summaries)
}
se.note <- function() {
    env <- sys.frame(-3)
    se <- "vcov"
    note <- paste(ifelse(se != "vcov","S","S"), # changed this line,
cut "Robust"
                  "tandard errors in parentheses",
                  ifelse(se =="both",
                         paste("\\\\\n\\multicolumn{",
                               evalq(nmodels,env)+1,"}{l}{",
                               'Na\\"ive standard errors in brackets',
                               collapse="",sep=""),
                         "" ) ,sep="")
    return(note)
}
stars.note <- function() {
    ## env <- sys.frame(-3)
    ## paste(ifelse(evalq(stars, envir = env)=="default",
    ##              paste("$^\\dagger$ significant at $p<.10$; $^* p<.
05$; $^{**} p<.01$; $^{***} p<.001$"),
    ##              paste("$^*$ indicates significance at
$p<",#evalq(lev, envir = env)
    ##                    0.05,"$")))
    paste("$^\\dagger$ significant at $p<.10$; $^* p<.05$; $^{**} p<.
01$; $^{***} p<.001$")
}
```

```
##########################################################################
##########################################################################
##########################################################################
##########################################################################
##########################################################################
##########################################################################
```

```
#####################################################################
#####################################################################
#####################################################################
#####################################################################
#####################################################################
#####################################################################
#############################################################



## ## Redefine the default summary method for R to display the P-
values, even though its wrong
## setMethod("summary","mer",
##           function (object, ...) {
##             require(nlme)
##             REML <- object@dims["REML"]
##             fcoef <- lme4:::fixef(object)
##             vcov <- vcov(object)
##             corF <- vcov@factors$correlation
##             dims <- object@dims
##             coefs <- cbind(Estimate = fcoef, `Std. Error` =
corF@sd)
##             llik <- logLik(object, REML)
##             dev <- object@deviance
##             mType <- if ((non <- as.logical(length(object@V))))
##               "NMM"
##             else "LMM"
##             if (gen <- as.logical(length(object@muEta)))
##               mType <- paste("G", mType, sep = "")
##             mName <- switch(mType, LMM = "Linear", NMM =
"Nonlinear",
##                             GLMM = "Generalized linear", GNMM =
"Generalized nonlinear")
##             method <- {
##               if (mType == "LMM")
##                 if (REML)
##                   "REML"
##                 else "maximum likelihood"
##               else paste("the", if (dims["nAGQ"] == 1)
##                          "Laplace"
##               else "adaptive Gaussian Hermite", "approximation")
##             }
##             AICframe <- data.frame(AIC = AIC(llik), BIC =
BIC(llik),
##                                    logLik = as.vector(llik),
deviance = dev["ML"], REMLdev = dev["REML"],
##                                    row.names = "")
##             if (is.na(AICframe$REMLdev))
##               AICframe$REMLdev <- NULL
```

```
##             varcor <- lme4:::VarCorr(object)
##             REmat <- lme4:::formatVC(varcor)
##             if (is.na(attr(varcor, "sc")))
##               REmat <- REmat[-nrow(REmat), , drop = FALSE]
##             if (nrow(coefs) > 0) {
##               if (!dims["useSc"]) {
##                 coefs <- coefs[, 1:2, drop = FALSE]
##                 stat <- coefs[, 1]/coefs[, 2]
##                 pval <- 2 * pnorm(abs(stat), lower = FALSE)
##                 coefs <- cbind(coefs, `z value` = stat, `Pr(>|z|)`
= pval)
##               }
##               else {
##                 stat <- coefs[, 1]/coefs[, 2]
##                 coefs <- cbind(coefs, `t value` = stat)
##                 #Hack these 2 lines:
##                 pval <- 2 * pt(abs(stat),df=(dims["n"] -
dims["p"]) , lower = FALSE)
##                 coefs <- cbind(coefs, `Pr(>|t|)` = pval)
##               }
##             }
##             new("summary.mer", object,
##                 methTitle = paste(mName, "mixed model fit by",
##                   method), logLik = llik, ngrps =
sapply(object@flist,
##                                               function(x)
length(levels(x))), sigma = lme4:::sigma(object),
##                 coefs = coefs, vcov = vcov, REmat = REmat,
##                 AICtab = AICframe)
##           })
```