contrasts.fit exploration

Léopold Guyot

2024-08-21

```r
library("scp")
library("msqrob2")
library("limma")
library("lme4")
library("UpSetR")
library("ggplot2")
library("tidyverse")
```

```r
msqrobLm_fix <- function (y, formula, data, robust = TRUE, maxitRob = 5)
{
    myDesign <- model.matrix(formula, data)
    models <- apply(y, 1, function(y, design) {
        obs <- is.finite(y)
        type <- "fitError"
        model <- list(coefficients = NA, vcovUnscaled = NA,
            sigma = NA, df.residual = NA, w = NA)
        if (sum(obs) > 0) {
            X <- design[obs, , drop = FALSE]
            qrX <- qr(X)
            X <- X[, qrX$pivot[seq_len(qrX$rank)], drop = FALSE] # change
            # => old version
            # X <- X[, colMeans(X == 0) != 1, drop = FALSE]
            y <- y[obs]
            colnames_orig <- colnames(design)
            if (robust) {
                mod <- try(MASS::rlm(X, y, method = "M", maxit = maxitRob),
                    silent = TRUE)
                if (!is(mod, "try-error")) {
                    type <- "rlm"
                }
            }
            else {
                mod <- try(lm.fit(X, y))
                if ((!is(mod, "try-error")) & mod$rank == ncol(X)) {
                    type <- "lm"
                }
            }
            if (type == "rlm") {
                w <- mod$w
                sigma <- sqrt(sum(mod$w * mod$resid^2)/(sum(mod$w) -
                    mod$rank))
                df.residual <- sum(mod$w) - mod$rank
                if (df.residual < 2L)
```

```r
                    type <- "fitError"
            }
            if (type == "lm") {
                w <- NULL
                sigma <- sqrt(sum(mod$residuals^2/mod$df.residual))
                df.residual <- mod$df.residual
                if (df.residual < 2L)
                    type <- "fitError"
            }
            if (type != "fitError") {
                coef <- rep(NA, length(colnames_orig))
                names(coef) <- colnames_orig
                coef[names(mod$coef)] <- mod$coef
                vcovUnscaled <- matrix(NA, nrow = length(colnames_orig),
                    ncol = length(colnames_orig))
                rownames(vcovUnscaled) <- colnames(vcovUnscaled) <- colnames_orig
                vcovUnscaled[names(mod$coef), names(mod$coef)] <- msqrob2:::.vcovUnscaled(mod)
                model <- list(coefficients = mod$coef, vcovUnscaled = msqrob2:::.vcovUnscaled(mod),
                    sigma = sigma, df.residual = df.residual,
                    w = w)
            }
        }
        .StatModel(type = type, params = model, varPosterior = as.numeric(NA),
            dfPosterior = as.numeric(NA))
    }, design = myDesign)
    hlp <- limma::squeezeVar(var = vapply(models, getVar, numeric(1)),
        df = vapply(models, getDF, numeric(1)))
    for (i in seq_len(length(models))) {
        mydf <- hlp$df.prior + getDF(models[[i]])
        models[[i]]@varPosterior <- as.numeric(hlp$var.post[i])
        models[[i]]@dfPosterior <- as.numeric(mydf)
    }
    return(models)
}


### msqrob2::msqrob with the new msqrobLm

msqrob_fix <-  .local <- function (object, formula, modelColumnName = "msqrobModels",
        overwrite = FALSE, robust = TRUE, ridge = FALSE, maxitRob = 1,
        tol = 1e-06, doQR = TRUE, lmerArgs = list(control = lmerControl(calc.derivs = FALSE)))
    {
        if (ncol(colData(object)) == 0)
            stop("colData is empty")
        if ((modelColumnName %in% colnames(rowData(object))) &
            !overwrite) {
            stop("There is already a column named '", modelColumnName,
                "' in the rowData of the SummarizedExperiment object, set the argument overwrite=TRUE t
        }
        if (length(formula) == 3) {
            formula <- formula[-2]
        }
        if (any(all.vars(formula) %in% colnames(rowData(object)))) {
            stop("Use the msqrobAggregate function to use rowData variables")
        }
        check_vars <- all.vars(formula) %in% colnames(colData(object))
        if (!all(check_vars)) {
            if (sum(!check_vars) > 1) {
                vars_not_found <- paste0(all.vars(formula)[!check_vars],
                    collapse = ", ")
                stop(paste("Variables", vars_not_found, "are not found in coldata"),
                    sep = "")
            }
            else {
                vars_not_found <- all.vars(formula)[!check_vars]
                stop(paste0("Variable ", vars_not_found, " is not found in coldata"),
                    sep = "")
            }
        }
        if (!ridge & is.null(findbars(formula))) {
            rowData(object)[[modelColumnName]] <- msqrobLm_fix(y = assay(object),
                formula = formula, data = droplevels(colData(object)),
                robust = robust, maxitRob = maxitRob)
        }
        else {
            rowData(object)[[modelColumnName]] <- msqrobLmer(y = assay(object),
                formula = formula, data = droplevels(colData(object)),
                rowdata = NULL, robust = robust, maxitRob = maxitRob,
                tol = tol, doQR = doQR, ridge = ridge, lmerArgs = lmerArgs)
        }
        rowData(object)[[modelColumnName]] <- rowData(object)[[modelColumnName]][rownames(rowData(objec
        return(object)
    }
```

```
.createMockBalanced <- function(k, runs) {
    out <- split(runs, runs)
    out <- lapply(out, function(x) {
        index <- 1:length(x)
        x[sample(index)] <- rep(1:k, length.out = length(index))
        x
    })
    out <- S4Vectors::unname(do.call(c, out))
    as.factor(out)
}

sce <- readRDS("data/leduc_mock.rds")
set.seed(1234)
sce$Mock <- .createMockBalanced(2, sce$Set)

num_features <- 2000  # Number of features to sample
total_features <- nrow(sce)

subset <- sce[sample(1:total_features, num_features),]
colData(subset)$Set <- as.factor(colData(subset)$Set)
colData(subset) <- droplevels(colData(subset))
```

```r
subset_msqrob <-  msqrob(subset,
                         formula = ~ 1 + Set,
                         robust = FALSE,
                         ridge = FALSE)
subset_msqrob_fix <- msqrob_fix(subset,
                         formula = ~ 1 + Set,
                         robust = FALSE,
                         ridge = FALSE)
subset_limma <- limma::lmFit(assay(subset),
                          model.matrix(~ 1 + Set,
                                       colData(subset)))
```

```
## Warning: Partial NA coefficients for 1840 probe(s)
```

```r
subset_scp <- scp::scpModelWorkflow(subset,
                                  formula = ~ 1 + Set,
                                  verbose = FALSE)
```

```
##   |                                                                              |
```

```r
design <- model.matrix(~ 1 + Set, colData(subset))

L_limma <- makeContrasts(SeteAL00220, levels = design)
```

```
## Warning in makeContrasts(SeteAL00220, levels = design): Renaming (Intercept) to
## Intercept
```

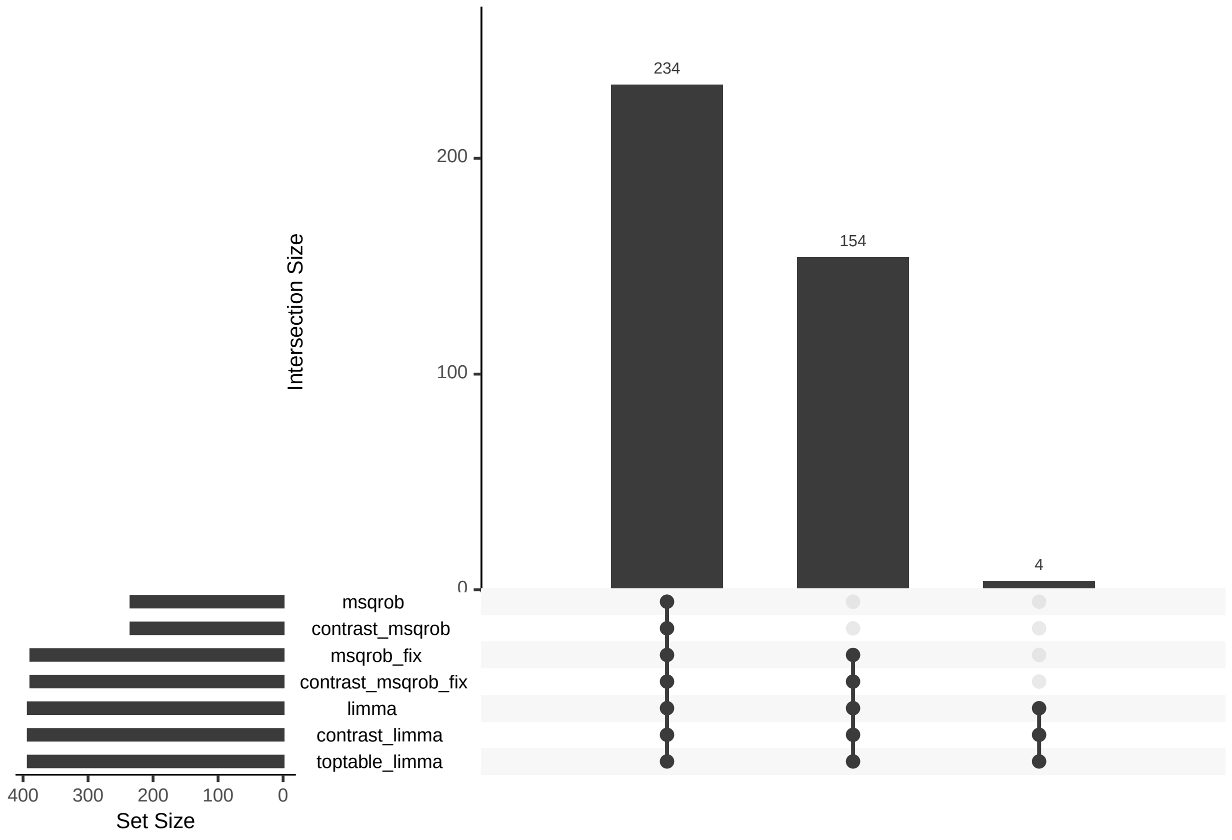
```r
contrast_limma <- contrasts.fit(subset_limma, L_limma)
toptable <- eBayes(contrast_limma)
toptable <- topTable(toptable, sort.by = "none", number = 2000)

L_msqrob <- makeContrast(c("SeteAL00220 = 0"), parameterNames = colnames(design))
contrast_msqrob <- hypothesisTest(subset_msqrob, contrast = L_msqrob)
contrast_msqrob <- rowData(contrast_msqrob)$SeteAL00220

contrast_msqrob_fix <- hypothesisTest(subset_msqrob_fix, contrast = L_msqrob)
contrast_msqrob_fix <- rowData(contrast_msqrob_fix)$SeteAL00220

try(contrast_scp <- scp::scpDifferentialAnalysis(subset_scp, contrasts = list(c("Set", "eAL00220", "eAL
```

```
## Error in t(l) %*% mm : non-conformable arguments
```

```r
# contrast_scp <- contrast_scp[[1]] %>%
#   as.data.frame()
# rownames(contrast_scp) <- contrast_scp$feature
```

```r
retrieve_msqrob_coef <- function(msqrob_model, coef_name){
    msqrob_object <- rowData(msqrob_model)$msqrobModels
  coef <- unlist(lapply(msqrob_object, function(x) {
      tryCatch(x@params$coefficients[coef_name],
               error = function(e) NA)
  }))
  names(coef) <- rownames(msqrob_model)
  coef
}
summary_df <- data.frame(
  limma = subset_limma$coefficients[,"SeteAL00220"],
  msqrob = retrieve_msqrob_coef(subset_msqrob, "SeteAL00220"),
  msqrob_fix = retrieve_msqrob_coef(subset_msqrob_fix, "SeteAL00220"),
  contrast_limma = contrast_limma$coefficients[,"SeteAL00220"],
  toptable_limma = toptable$logFC,
  contrast_msqrob = contrast_msqrob$logFC,
  contrast_msqrob_fix = contrast_msqrob_fix$logFC,
  row.names = rownames(subset)
)
```

```
sets <- list(
  limma = row.names(summary_df)[!is.na(summary_df$limma)],
  contrast_limma = row.names(summary_df)[!is.na(summary_df$contrast_limma)],
  toptable_limma = row.names(summary_df)[!is.na(summary_df$toptable_limma)],
  msqrob = row.names(summary_df)[!is.na(summary_df$msqrob)],
  contrast_msqrob = row.names(summary_df)[!is.na(summary_df$contrast_msqrob)],
  msqrob_fix = row.names(summary_df)[!is.na(summary_df$msqrob_fix)],
  contrast_msqrob_fix = row.names(summary_df)[!is.na(summary_df$contrast_msqrob_fix)]
)
upset(fromList(sets), nsets = 7, order.by = "freq")
```



```
to_test_msqrob <- rownames(summary_df)[!is.na(summary_df$msqrob)]

identical(summary_df[to_test_msqrob, "limma"], summary_df[to_test_msqrob, "msqrob"])
```

```
## [1] TRUE
```

```r
identical(summary_df[to_test_msqrob, "limma"], summary_df[to_test_msqrob, "contrast_msqrob"])
```

```
## [1] TRUE
```

```r
identical(summary_df[to_test_msqrob, "limma"], summary_df[to_test_msqrob, "msqrob_fix"])
```

```
## [1] TRUE
```

```r
identical(summary_df[to_test_msqrob, "limma"], summary_df[to_test_msqrob, "contrast_msqrob_fix"])
```

```
## [1] TRUE
```

```r
identical(summary_df[to_test_msqrob, "limma"], summary_df[to_test_msqrob, "contrast_limma"])
```

```
## [1] TRUE
```

```r
identical(summary_df[to_test_msqrob, "limma"], summary_df[to_test_msqrob, "toptable_limma"])
```

```
## [1] TRUE
```

```r
to_test_fix <- rownames(summary_df)[!is.na(summary_df$msqrob_fix)]

identical(summary_df[to_test_fix, "limma"], summary_df[to_test_fix, "msqrob_fix"])
```

```
## [1] TRUE
```

```r
identical(summary_df[to_test_fix, "limma"], summary_df[to_test_fix, "contrast_msqrob_fix"])
```

```
## [1] TRUE
```

```r
identical(summary_df[to_test_fix, "limma"], summary_df[to_test_fix, "contrast_limma"])
```

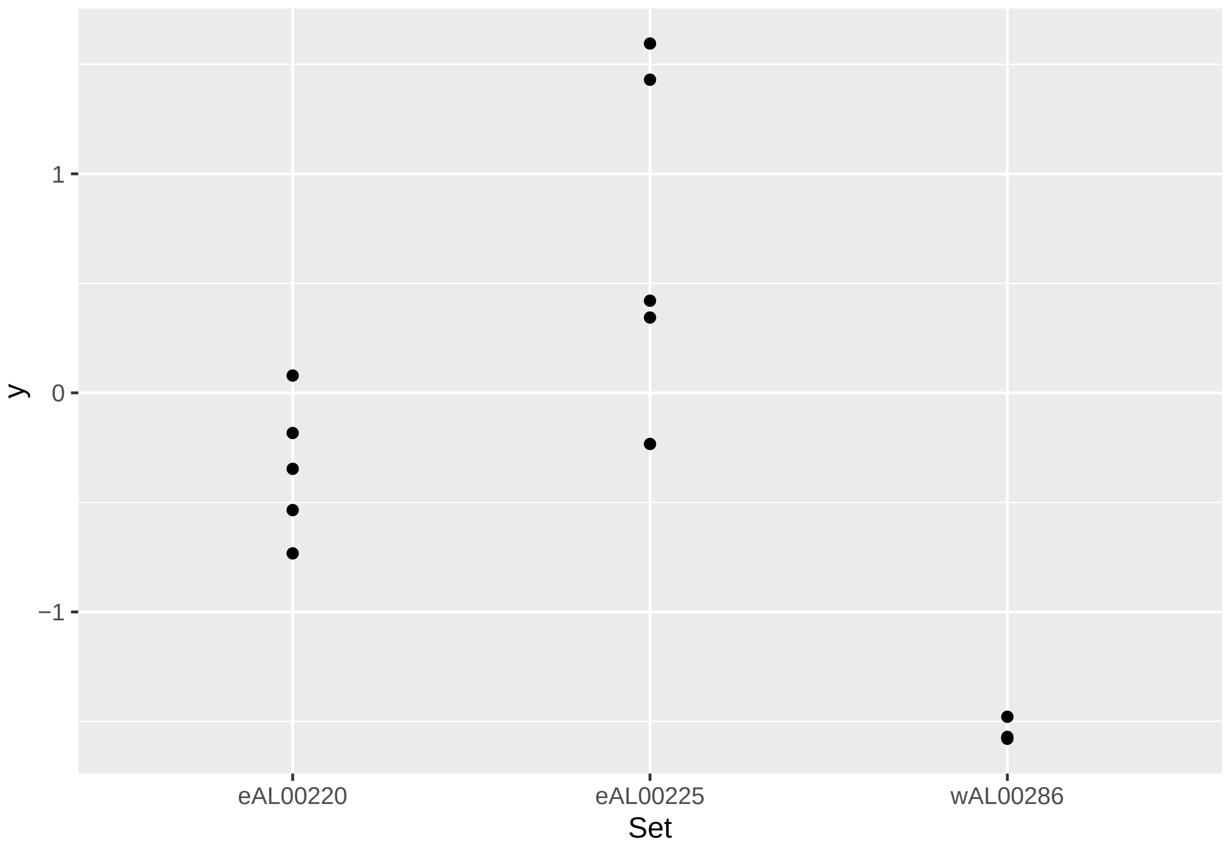
```
## [1] TRUE
```

```r
identical(summary_df[to_test_fix, "limma"], summary_df[to_test_fix, "toptable_limma"])
```

```
## [1] TRUE
```

```r
test <- subset_limma$coefficients
test <- test[,"SeteAL00220"]
testthat::expect_identical(as.numeric(test), as.numeric(subset_limma[,"SeteAL00220"]$coefficients))
```

```r
pe <- setdiff(to_test_fix, to_test_msqrob)[5]
limma_coef_pe <- subset_limma$coefficients[pe,]

df <- data.frame(y = assay(subset)[pe,], colData(subset)) %>%
    dplyr::filter(!is.na(y)) %>%
    droplevels()
ref_class <- levels(df$Set)[1]
new_ref_class <- tail(levels(df$Set),1)
test_class <- levels(df$Set)[2]

df <- df %>%
    dplyr::filter(Set %in% c(ref_class, test_class, new_ref_class))
ggplot(df,aes(x = Set, y = y))+geom_point()
```



```
limma_coef_pe_test_class <- limma_coef_pe[paste0("Set", test_class)]
df_mean <- df %>%
    group_by(Set) %>%
    summarise(mean = mean(y))
df_mean$mean[2]-df_mean$mean[3]
```

```
## [1] 2.253972
```