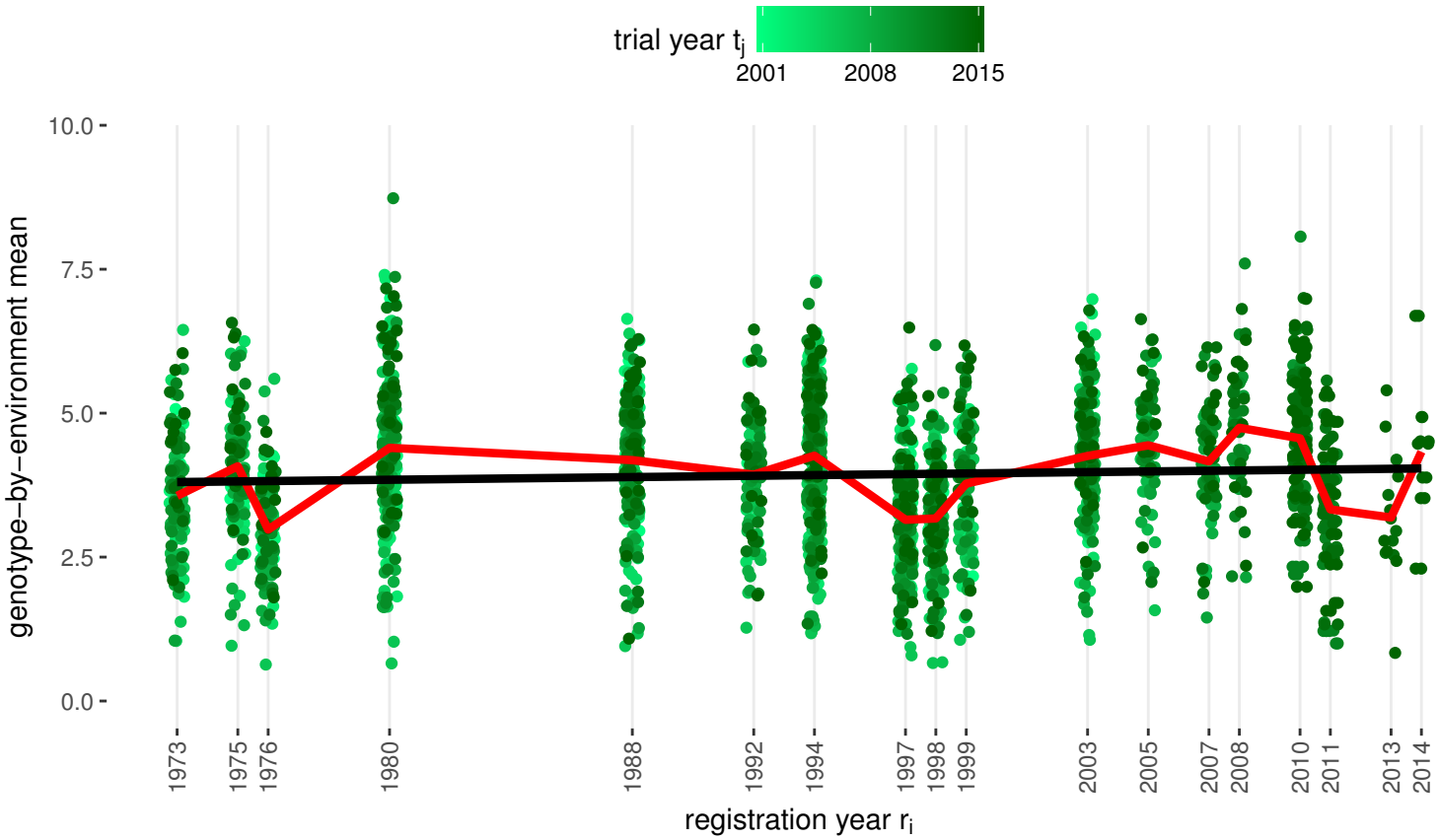
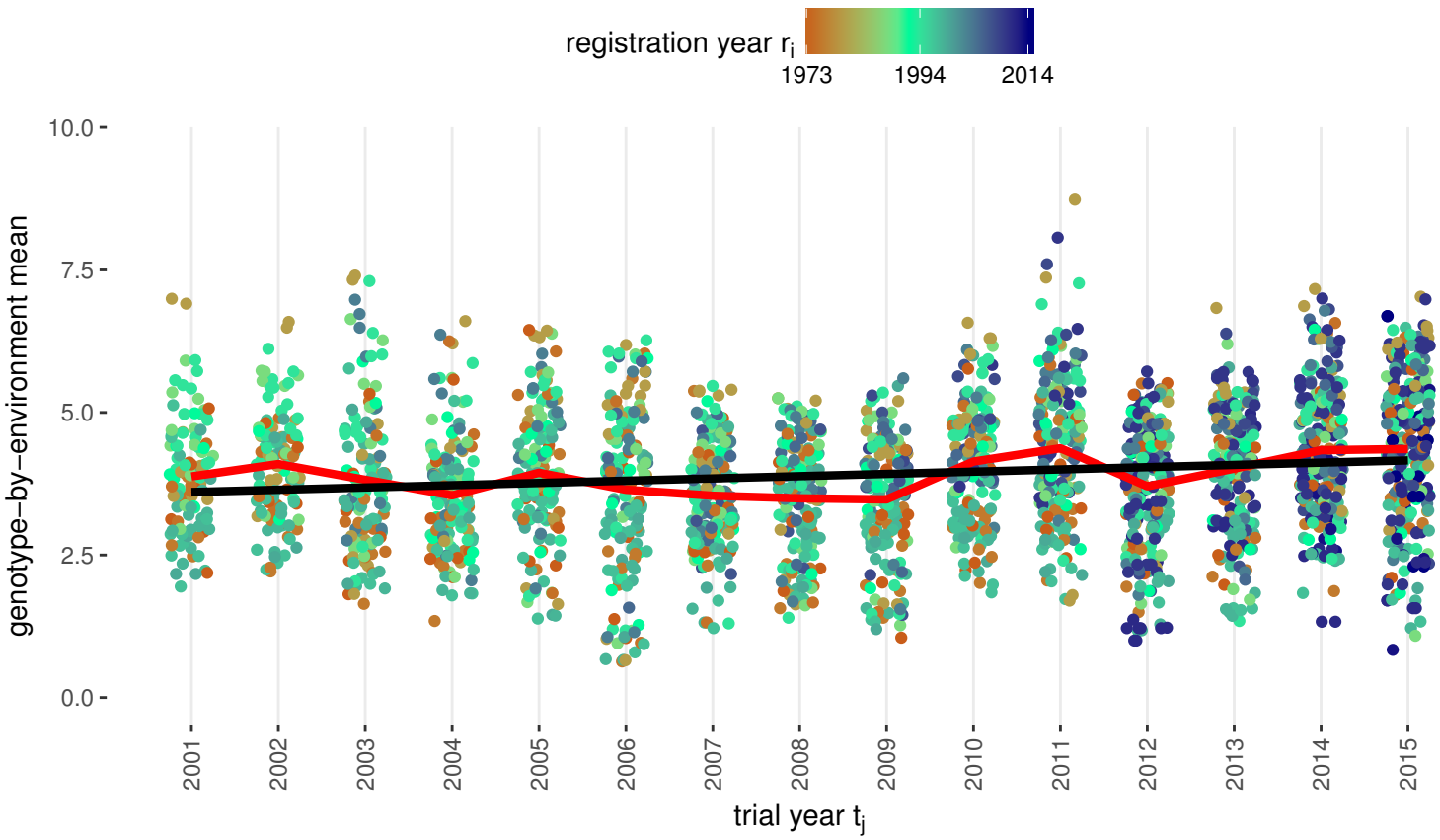
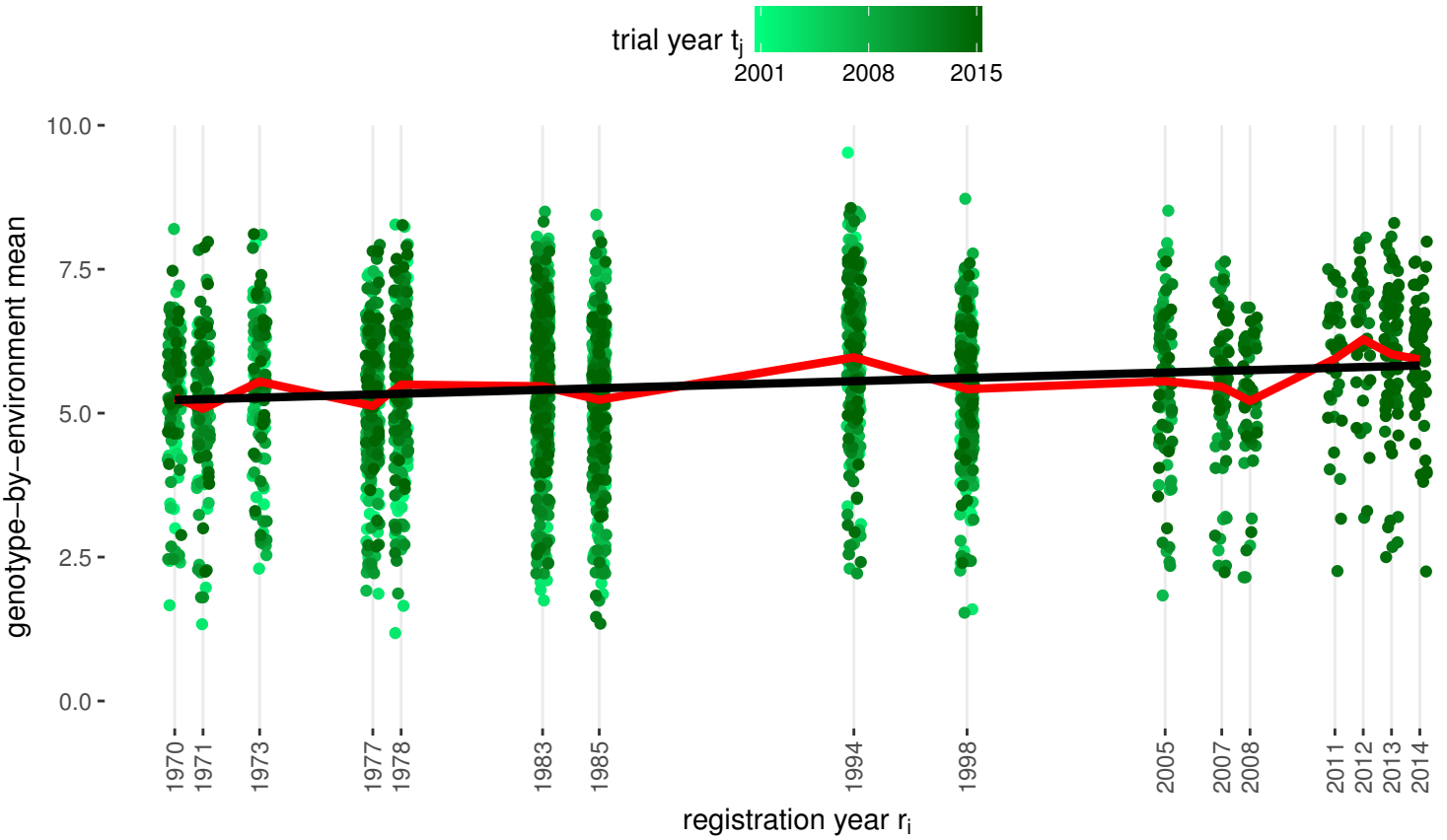


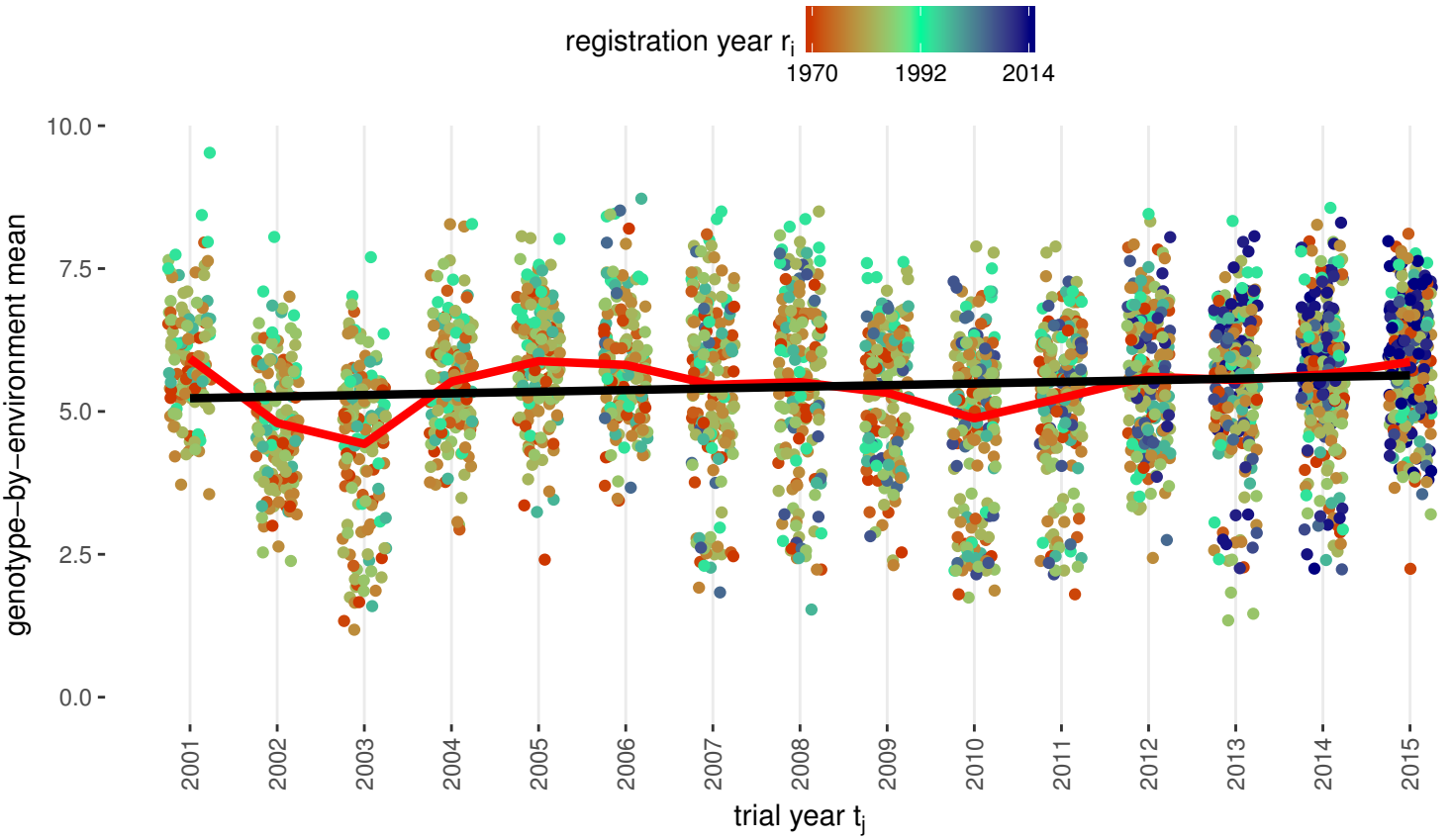
Genotype−by−environment means from aman dataset.
Red line: arithmetic mean per year. Black line: linear Regression y = a + bx.



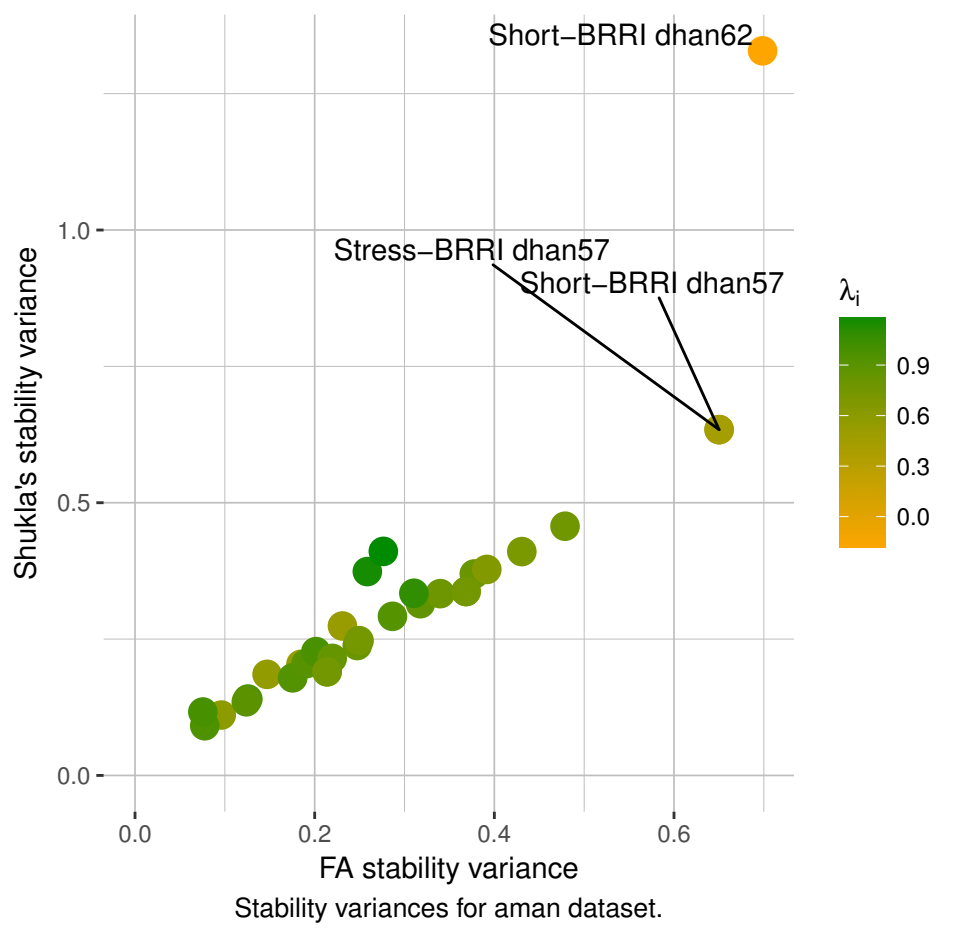
Genotype−by−environment means from aman dataset.
Red line: arithmetic mean per year. Black line: linear Regression y = a + bx.



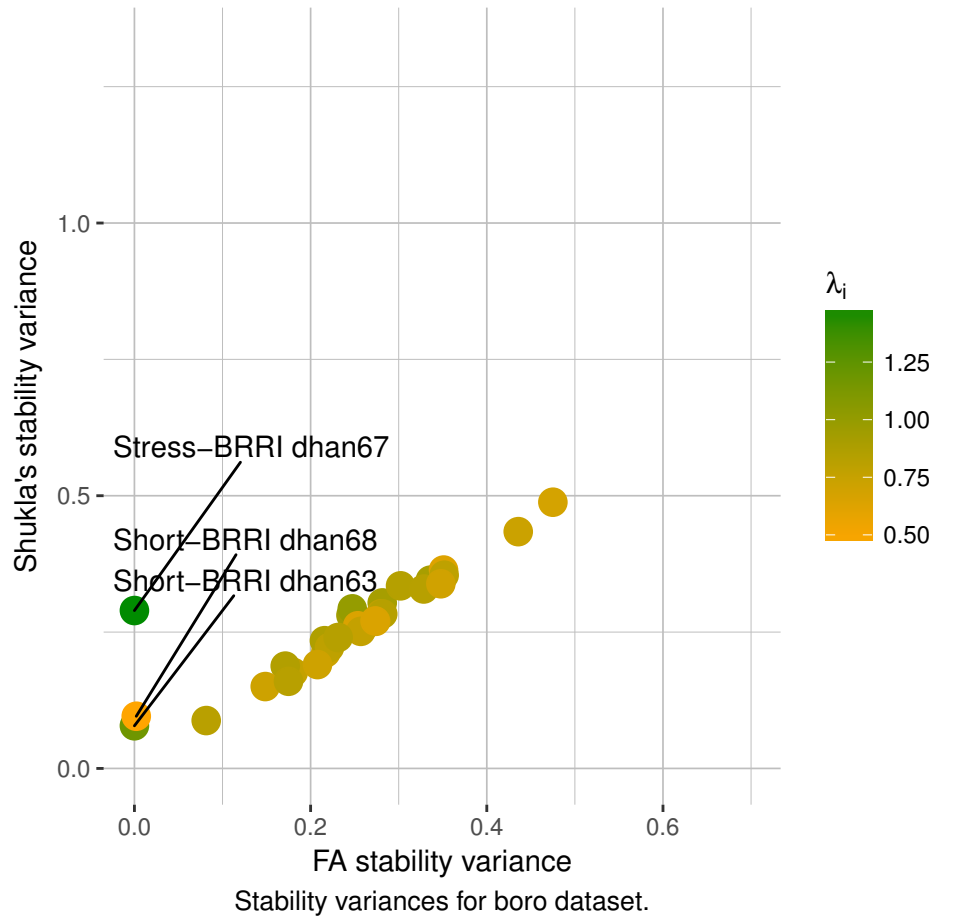
Genotype–by–environment means from boro dataset.
Red line: arithmetic mean per year. Black line: linear Regression y = a + bx.



Genotype–by–environment means from boro dataset.
Red line: arithmetic mean per year. Black line: linear Regression y = a + bx.



Stability variances for aman dataset.



Stability variances for boro dataset.

01_import.R

Paul Schmidt

April 23rd, 2018

```
# make sure to correctly set working directory with setwd() #

### Import
amanraw <- read.delim("20171212 Corrected Final GG Aman 09.08.2017.txt")
bororaw <- read.delim("20171212 Corrected Final GG Boro 09.08.2017.txt")

### individual changes
amanraw$t.j <- as.numeric(amanraw$Year)
bororaw$t.j <- as.numeric(substr(bororaw$Year,1,4))

### prepare lists for loop
dat.names <- c("aman", "boro")
list.data <- setNames(as.list(c(NA, NA)), dat.names)
set.names <- c("plot data raw", "plot data formatted", "plot data info", "GxE means")
datasets  <- setNames(as.list(c(NA, NA, NA, NA)), set.names)
list.data[["aman"]] <- datasets; list.data[["aman"]][["plot data raw"]] <- amanraw
list.data[["boro"]] <- datasets; list.data[["boro"]][["plot data raw"]] <- bororaw

### loop through aman & boro
for (sel.dat in dat.names) {

  d <- list.data[[sel.dat]][["plot data raw"]]

  # format raw data
    d$r.i   <- as.numeric(d$Year.of.release)
    d$Y     <- as.factor(d$Year)
    d$L     <- as.factor(d$Location)
    d$Env   <- as.factor(paste(d$Y,d$L,sep="-"))
    d$Rep   <- as.factor(d$Rep)
    d$G     <- as.factor(d$Variety)
    d$Group <- as.factor(d$Group)
    d$GG  <- as.factor(paste(d$Group,d$G,sep="-"))
    d <- d[,c("Y","t.j","r.i","L","Env","G","Rep","Group","GG","Yield")]
    d <- subset(d, is.na(d$Yield)==F)

    list.data[[sel.dat]][["plot data formatted"]] <- d

  # additinally export formatted data as text file
  write.table(x = list.data[[sel.dat]][["plot data formatted"]], sep="\t",
              file = paste0(sel.dat," formatted.txt"))

  # aggregate information on dataset's dimensions and levels
    columns <- names(d)[names(d)!="Yield"]
    info    <- matrix(list(),length(columns),2)
    rownames(info) <- columns
    for (x in c(1:length(columns))){
      info[[x,1]] <-    length(unique(as.factor(d[,c(x)])))
      info[[x,2]] <- data.frame(table(as.factor(d[,c(x)])))
      colnames(info) <- c("number","freq_list")
    }

    list.data[[sel.dat]][["plot data info"]] <- info

}

### Save output
saveRDS(list.data, file = "list_data.rds")
```

02a_get_GxE_means.R

Paul Schmidt

April 23rd, 2018

```
# load required packages
library(data.table)
library(emmeans)
```

```
## Warning: replacing previous import by 'stats::coef' when loading 'emmeans'
```

```
library(plyr)

list.data <- readRDS("list_data.rds")

### loop through aman and boro ##########
# Obtain genotype means per environment #
# 1: arithmetic means                   #
# 2: adjusted means                     #

for (sel.dat in names(list.data)) {

  dat <- as.data.table(list.data[[sel.dat]][["plot data formatted"]])

  ### arithmetic means
  dat[, ari.mean := mean(Yield), by=c("Env","GG")] # calc. mean yield per Env-GG combination
  e.mean.ari <- unique(dat[, -c("Rep","Yield")])   # reduce dataset - eliminate duplicate rows

  ### adjusted means

  # prepare list
  n.e         <- length(levels(dat$Env))
  e.mean.adj <- matrix(list(), n.e, 1)
  rownames(e.mean.adj) <- unique(dat$Env)

  # loop through and analyze each environment
  for (sel.env in c(1:n.e)){
    mod <- lm(formula = Yield ~ GG + Rep,
              data    = subset(dat, Env==dat$Env[sel.env]))
    m       <- emmeans(mod, "GG")
    means   <- as.data.frame(m)[,c("GG", "emmean")]
    names(means) <- c("GG", "adj.mean")
    means$w <- diag(solve(vcov(m)))

    e.mean.adj[[sel.env]] <- means
  }

  # bring together all adj. means from different environments
  e.mean.adj <- data.table(plyr::ldply(e.mean.adj[,1], data.frame, .id="Env"))

  # combine arithmetic and adjusted means
  e.means <- data.table(merge(e.mean.ari, e.mean.adj, by=c("Env","GG")))
  list.data[[sel.dat]][["GxE means"]] <- e.means[, c("Y", "L", "Env", "t.j",
                                                     "r.i", "Group", "G", "GG",
                                                     "ari.mean", "adj.mean", "w")]

  # additionally export GxE means as text file
  write.table(x = list.data[[sel.dat]][["GxE means"]], sep="\t",
              file = paste0(sel.dat," GxE means.txt"))

}

### Check for data discrepancies
list.data[["aman"]][["GxE means"]][ ari.mean - adj.mean > 0.000001]
```

```
##       Y      L         Env  t.j  r.i  Group           G                GG
## 1: 2007 Kustia 2007-Kustia 2007 1997  Short BRRI dhan33  Short-BRRI dhan33
## 2: 2015 Bhanga 2015-Bhanga 2015 2005 Stress BRRI dhan44 Stress-BRRI dhan44
##    ari.mean adj.mean         w
## 1:     3.04 2.989104  3.338907
## 2:     5.30 5.294583 16.891927
```

```
list.data[["boro"]][["GxE means"]][ ari.mean - adj.mean > 0.000001]
```

```
## Empty data.table (0 rows) of 11 cols: Y,L,Env,t.j,r.i,Group...
```

```
### Save output
saveRDS(list.data, file = "list_data.rds")
```

03_genetic_gain_m1-3.R

Paul Schmidt

April 23rd, 2018

```
# load required packages
library(asreml)
library(nadiv)
library(stringr)
library(data.table)

list.data <- readRDS("list_data.rds")

list.results <- setNames(as.list(c(NA, NA)), names(list.data))

# loop through aman & boro
for (sel.dat in names(list.data)) {
  dat      <- list.data[[sel.dat]][["GxE means"]]
  setorder(dat, G, Env)

  # basic model
  Basic <- asreml(fixed   = adj.mean ~ r.i + t.j,
                  random  = ~ GG + L + Y + Y:L + L:GG + Y:GG +
                              GG:Y:L,
                  weights = w,
                  family  = asreml.gaussian(dispersion=1.0),
                  data    = dat, ran.order = "user")

  # shukla's stability variances
  Shukla <- asreml(fixed   = adj.mean ~ r.i + t.j,
                   random  = ~ GG + L + Y + Y:L + L:GG + Y:GG +
                               at(GG):Y:L,
                   weights = w,
                   family  = asreml.gaussian(dispersion=1.0),
                   data    = dat, ran.order = "user")

  # factor-analytic
  FA <- asreml(fixed   = adj.mean ~ r.i + t.j,
               random  = ~ GG + L + Y + L:GG + Y:GG +
                           fa(GG):Env, #note: fa(G):Y:L does not converge
               weights = w,
               family  = asreml.gaussian(dispersion=1.0),
               control = asreml.control(maxiter=100),
               data    = dat, ran.order = "user")

  #### aggregate results
  m.names <- c("Basic", "Shukla", "FA")
  output  <- c("model object", "model fit", "fixed effect solutions", "fixed effect F-tests",
               "variance components raw", "variance components formatted")
  results <- matrix(list(), length(output), length(m.names))
    rownames(results) <- output
    colnames(results) <- m.names
  results[[1,1]] <- Basic
  results[[1,2]] <- Shukla
  results[[1,3]] <- FA

  for (i in c(1:3)){
    # fixed effect solutions
    results[[3,i]] <- summary(results[[1,i]], all=T)$coef.fixed

    # fixed effect F-tests
    results[[4,i]] <- wald(results[[1,i]], denDF=c("numeric"), ssType=c("conditional"))$Wald

    # variance components raw
      vc1 <- data.table(summary(results[[1,i]])$varcomp,  keep.rownames="CovParm")
      vc2 <- data.table(   aiCI(results[[1,i]]),          keep.rownames="CovParm")
    results[[5,i]] <- vc2[vc1, .(CovParm, LCL, estimate, UCL, std.error), on="CovParm"]
    rm(vc1, vc2)

    # AIC
    loglik    <- summary(results[[1,i]])$loglik
```

```
    vc        <- data.table(summary(results[[1,i]])$varcomp)
    n.par.est <- dim(vc[constraint!="Fixed" & constraint!="Constrained"])[1]
    m2logLik  <- -2 * loglik
    AIC       <- -2 * loglik + 2*(n.par.est)
    out <- data.frame(n.par.est, loglik, m2logLik, AIC)
    results[[2,i]] <- out
    rm(out, vc, loglik, n.par.est, m2logLik, AIC)
  }

  # variance component formatting
  for (i in c(1:3)){
    vc <- results[[5,i]]
    vc[, names(vc)[2:5]:=round(.SD,4), .SDcols=names(vc)[2:5]]
      is.var   <- str_detect(vc$CovParm,"\\.var|R!variance")
      is.at    <- str_detect(substr(vc$CovParm,1,3), "at\\(")
      is.FA    <- str_detect(substr(vc$CovParm,1,3), "fa\\(")
      is.basic <- is.at==F & is.FA==F
    if (T %in% is.at) {
      vc$CovParm[is.at] <- str_match(vc$CovParm[is.at], levels(dat$GG))
    }
    vc$CovParm[is.basic] <- tstrsplit(vc$CovParm[is.basic],"!",fixed=T)[[1]]
    results[[6,i]] <- vc
    if (T %in% is.FA) {
      vc$CovParm[is.FA] <- str_match(vc$CovParm[is.FA], levels(dat$GG))
      vc[, type:= ifelse(is.var, "sigma", "lambda")]
      FA.wide.all <- vc[is.FA, c("CovParm", "type", "LCL", "estimate", "UCL", "std.error")]
      FA.wide     <- dcast(FA.wide.all, CovParm ~ type, value.var="estimate")
      top <- vc[is.FA==F, c("CovParm", "type", "estimate")]
      names(top)[2:3] <- c("lambda", "sigma"); top$lambda <- NA
      FA.wide <- rbind(top, FA.wide); rm(top)
      FA.wide[, lambda:= as.numeric(paste(lambda))]
      results[[6,i]] <- FA.wide
    }
    rm(is.var, is.FA, is.at, is.basic)
  }

  list.results[[sel.dat]] <- results

}

### Save output
saveRDS(list.results, file = "list_results.rds")
```
```
```

03_genetic_gain_m4.R

Paul Schmidt

April 23rd, 2018

```
list.data <- readRDS("list_data.rds")

list.stabtrend <- list()

### Set up model construction set
  RP1 <- data.frame(part1 ="random=~ GG + L + Y + Y:L + GG:L + GG:Y + GG:Env",
                    riGL  ="+ sc.ri:GG:L",
                    riGY  ="+ sc.ri:GG:Y",
                    tjGY  ="+ sc.tj:GG:Y",
                    riGYL ="+ sc.ri:GG:Env",
                    tjGYL ="+ sc.tj:GG:Env")
  RP <- as.formula(paste0(RP1[,1],RP1[,2],RP1[,3],RP1[,4],RP1[,5],RP1[,6]))
  cols <- c(2,3,4,5,6)
  comblist <- matrix(list(),length(cols)+1,1)
  for(i in 1:length(cols)){comblist[[i]] <- combn(cols,i)}
  out_list  <- matrix(list(),6,10)

for (sel.dat in names(list.data)) {
  dat     <- list.data[[sel.dat]][["GxE means"]]
  setorder(dat, G, Env)

  dat$sc.ri <- sqrt(dat$r.i-min(dat$r.i))
  dat$sc.tj <- sqrt(dat$t.j-min(dat$t.j))

### Get and manage covparm constraints
  Dmod <- asreml(start.values = T,
                 fixed   = adj.mean ~ r.i + t.j,
                 random  = RP,
                 weights = w,
                 family  = asreml.gaussian(dispersion=1.0),
                 data    = dat, ran.order = "user")
  Gpar <- Dmod$gammas.table
  Gpar[str_detect(Gpar$Gamma,"sc\\."),"Constraint"] <- "U"

### Fit full model
  Dmod <- asreml(fixed   = adj.mean ~ r.i + t.j,
                 random  = RP,
                 G.param = Gpar,
                 weights = w,
                 family  = asreml.gaussian(dispersion=1.0),
                 control = asreml.control(maxiter = 100),
                 data    = dat, ran.order = "user")

### Extract VCs
  VC         <- data.table(summary(Dmod)$varcomp[c(2,3,5)], keep.rownames="CovParm")
  VC$CovParm <- tstrsplit(VC$CovParm,"!",fixed=T)[[1]]

  VCwide <- dcast(VC[str_detect(CovParm,"sc.")], . ~ CovParm, value.var = "component")[,-1]
  VCwide$Conv <- Dmod$converge
  VCwide$loglik <- Dmod$loglik
  VCwide$n.par  <- length(Dmod$gammas) - dim(VC[constraint=="Fixed"|constraint=="Constrained"])[1]
  VCwide$AIC    <- -2 * Dmod$loglik + 2*(VCwide$n.par)
  VCwide$t.j <- round(summary(Dmod, all=T)$coef.fixed["t.j",1],4)
  VCwide$t.j_p <- wald(Dmod, denDF=c("numeric"), ssType=c("conditional"))$Wald["t.j","Pr"]
  VCwide$r.i <- round(summary(Dmod, all=T)$coef.fixed["r.i",1],4)
  VCwide$r.i_p <- wald(Dmod, denDF=c("numeric"), ssType=c("conditional"))$Wald["r.i","Pr"]
  out_list[[6,1]] <- VCwide

  regnames <- c("sc.ri:GG:L","sc.ri:GG:Y","sc.tj:GG:Y","sc.ri:GG:Env","sc.tj:GG:Env")
  #names(out_list[[6,1]]) <- regnames

### Run models with all other random effect combinations
  for (i in 1:length(cols)){
    for (j in 1:dim(comblist[[i]])[2]){

      # Construng Random Part of Model
```

```
      loopRP1 <- RP1
      loopRP1[,comblist[[i]][,j]] <- ""
      loopRP <- as.formula(paste0(loopRP1[,1],loopRP1[,2],
                                  loopRP1[,3],loopRP1[,4],
                                  loopRP1[,5],loopRP1[,6]))

      # Get Starting Values
      Dmod <- asreml(start.values = T,
                     fixed   = adj.mean ~ r.i + t.j,
                     random  = loopRP,
                     weights = w,
                     family  = asreml.gaussian(dispersion=1.0),
                     control = asreml.control(maxiter=100),
                     data    = dat, ran.order = "user")

      # Manage Constraints
      Gpar <- Dmod$gammas.table
      Gpar[str_detect(Gpar$Gamma,"sc."),"Constraint"] <- "U"
      Gpar[1:6,"component"] <- c(0.06,0.51,0.16,0.03,0.01,0.82) # starting values

      # Fit Model
      Dmod <- asreml(fixed   = adj.mean ~ r.i + t.j,
                     random  = loopRP,
                     weights = w,
                     G.param = Gpar,
                     family  = asreml.gaussian(dispersion=1.0),
                     control = asreml.control(maxiter=100),
                     data    = dat, ran.order = "user")

      VC         <- data.table(summary(Dmod)$varcomp[c(2,3,5)], keep.rownames="CovParm")
      VC$CovParm <- tstrsplit(VC$CovParm,"!",fixed=T)[[1]]

      out <- data.frame(riGL  =if(loopRP1[,2]!=""){"yes"}else{""},
                        riGY  =if(loopRP1[,3]!=""){"yes"}else{""},
                        tjGY  =if(loopRP1[,4]!=""){"yes"}else{""},
                        riGYL =if(loopRP1[,5]!=""){"yes"}else{""},
                        tjGYL =if(loopRP1[,6]!=""){"yes"}else{""})

      for (v in 1:5){
        out[v] <- if(out[v]=="yes"){round(VC[CovParm==regnames[v],"component"],5)}else{""}
      }

      out$Conv   <- Dmod$converge
      vc         <- data.table(summary(Dmod)$varcomp)
      out$loglik <- Dmod$loglik
      out$n.par  <- dim(vc[constraint!="Fixed" & constraint!="Constrained"])[1]
      out$AIC    <- -2 * Dmod$loglik + 2*(out$n.par)

      out$t.j   <- round(summary(Dmod, all=T)$coef.fixed["t.j",1],4)
      out$t.j_p <- round(wald(Dmod, denDF=c("numeric"), ssType=c("conditional"))$Wald["t.j","Pr"],4)
      out$r.i   <- round(summary(Dmod, all=T)$coef.fixed["r.i",1],4)
      out$r.i_p <- round(wald(Dmod, denDF=c("numeric"), ssType=c("conditional"))$Wald["r.i","Pr"],4)

      out_list[[i,j]] <- out
    }
  }

  # Reformat full model output
    names(out_list[[6,1]]) <- names(out_list[[1,1]])
    out_list[[6,1]][,c(1:5,9:13)] <- round(out_list[[6,1]][,c(1:5,9:13)], 4)

  modcomp <- data.table(plyr::ldply(out_list, data.frame))
  modcomp[Conv==F, AIC:=NA]
  setorder(modcomp, AIC)

  list.stabtrend[[sel.dat]] <- modcomp
}

### Save output
saveRDS(list.stabtrend, file="list_stabtrend.rds")
```