Permutation tests

August 27, 2021
12:07

Seiro Ito

Contents

Use the 'trimmed' sample (has all 800 members) rather than the 'initial' sample (has only 776 members after dropping members who received loans only twice). To set to the trimmed sample, set the parameter UseTrimmedSample to T.

```
UseTrimmedSample ← T
TestMedian ← F
```

There are 92 members who attrited.

```
asv ← readRDS(paste0(pathsaveHere, "DestatData.rds"))
addmargins(table0(asv[!grepl("tw|dou", TradGroup), .(Arm, Attrited)]))
```

Attrition of members who were not affected by floods nor rejected.

```
addmargins(table0(asv[!grepl("flo", BStatus) & Rejected == 0, .(Arm, Attrited)]))
```

```
             Attrited
Arm             0   1 Sum
  traditional  83   2  85
  large       164   7 171
  large grace 160   7 167
  cow         147   6 153
  Sum         554  22 576
```

```
# these are HHs with two disbursements under traditional; read_admin_data.rnw(472)
# adw[(loanamount1st == 5600 & loanamount2nd == 5600 & loanamount3rd == 5600) |
#   (!is.na(DisDate1) & !is.na(DisDate2) & !is.na(DisDate3)),
#   TradGroup := "planned"]
# adw[loanamount1st == 5600 & loanamount2nd == 11200,
#   TradGroup := "double"]
# adw[(loanamount1st == 7840 & loanamount2nd == 8960) |
#   (!is.na(DisDate1) & !is.na(DisDate2) & is.na(DisDate3)),
#   TradGroup := "twice"]
# adw[, TradGroup := factor(TradGroup, levels = c("planned", "twice", "double"))]
```

```
# data to use in each tests: TradNonTradAttrited, AttritedInTrad, TradNonTradRejected, IRe
# drop 2 loan receivers
asv1 ← asv[!grepl("tw|dou", TradGroup), ]
# drop group rejecters
asv2 ← asv[!grepl("gr", BStatus), ]
# drop 2 loan receivers and group rejecters
asv3 ← asv[!grepl("gr", BStatus) & !grepl("tw|dou", TradGroup), ]
asvT ← asv[grepl("tra", Arm), ]
asvNT ← asv[!grepl("tra", Arm), ]
# data to be used for each tested variable
datalist ← rep("asv", length(vartobetested))
datalist1 ← paste0(datalist, 1) # drop 2 loan receivers
```

```
datalist2 ← paste0(datalist, 2) # drop group rejecters
datalist3 ← paste0(datalist, 3) # drop 2 loan receivers and group rejecters
datasets ← "asv"
datasets1 ← paste0(datasets, 1)
datasets2 ← paste0(datasets, 2)
datasets3 ← paste0(datasets, 3)
for (k in 1:3) {
  addchar ← c("f", "t", "j")[k]
  Datasets ← get(paste0("datasets", c("", 1, 2)[k]))
  for (dd in Datasets) {
    xdd ← get(dd)
    # all members all arms: attrited vs. nonattrited
    xa ← xdd
    assign(paste0(dd, "a", addchar), xa)
    # all in trad: attrition vs. nonattrition
    xTa ← xdd[grepl("trad", Arm), ]
    assign(paste0(dd, "Ta", addchar), xTa)
    # all in nontrad: attrition vs. nonattrition
    xNTa ← xdd[!grepl("trad", Arm), ]
    assign(paste0(dd, "NTa", addchar), xNTa)
    # attrited members in all arms: trad vs. nontrad
    xTNTa ← xdd[Attrited == 1L, ]
    xTNTa[, TradArm := 1L]; xTNTa[!grepl("trad", Arm), TradArm := 0L]
    assign(paste0(dd, "TNTa", addchar), xTNTa)
    # all members except flood victims: attrited vs. nonattrited
    xNFa ← xdd[!grepl("floo", BStatus), ]
    assign(paste0(dd, "NFa", addchar), xNFa)
    # all except flood victims in trad: attrition vs. nonattrition
    xNFTa ← xdd[!grepl("floo", BStatus) & grepl("trad", Arm), ]
    assign(paste0(dd, "NFTa", addchar), xNFTa)
    # all except flood victims in nontrad: attrition vs. nonattrition
    xNFNTa ← xdd[!grepl("floo", BStatus) & !grepl("trad", Arm), ]
    assign(paste0(dd, "NFNTa", addchar), xNFNTa)
    # attrited members except flood victims in all arms: trad vs. nontrad
    xNFTNTa ← xdd[!grepl("floo", BStatus) & Attrited == 1L, ]
    xNFTNTa[, TradArm := 1L]; xNFTNTa[!grepl("trad", Arm), TradArm := 0L]
    assign(paste0(dd, "NFTNTa", addchar), xNFTNTa)
    # attrited members except flood victims in all arms: cow vs. noncow
    xNFCNCa ← xdd[!grepl("floo", BStatus) & Attrited == 1L, ]
    xNFCNCa[, CowArm := 1L]; xNFCNCa[!grepl("cow", Arm), CowArm := 0L]
    assign(paste0(dd, "NFCNCa", addchar), xNFCNCa)
    # attrited members except flood victims: cow vs. large grace
    xNFCGa ← xdd[!grepl("floo", BStatus) & grepl("cow|gr", Arm) & Attrited == 1L, ]
    xNFCGa[, CowArm := 1L]; xNFCGa[!grepl("cow", Arm), CowArm := 0L]
    assign(paste0(dd, "NFCGa", addchar), xNFCGa)
    # being active (neither attrited nor rejected) members except flood victims
    # (these people are considered not fit for the offered program)
    # active in all arms
    xs ← xdd
    assign(paste0(dd, "s", addchar), xs)
    # active in trad: attrition vs. nonattrition
    xTs ← xdd[grepl("trad", Arm), ]
    assign(paste0(dd, "Ts", addchar), xTs)
    # active in nontrad: attrition vs. nonattrition
    xNTs ← xdd[!grepl("trad", Arm), ]
    assign(paste0(dd, "NTs", addchar), xNTs)
```

```
    # active members in all arms: trad vs. nontrad
    xTNTs ← xdd[Active == 1L, ]
    xTNTs[, TradArm := 1L]; xTNTs[!grepl("trad", Arm), TradArm := 0L]
    assign(paste0(dd, "TNTs", addchar), xTNTs)
    # active members: cow vs. noncow
    xCNCs ← xdd[!grepl("floo", BStatus) & Active == 1L, ]
    xCNCs[, CowArm := 1L]; xCNCs[!grepl("cow", Arm), CowArm := 0L]
    assign(paste0(dd, "CNCs", addchar), xCNCs)
    # active members: cow vs. lsge grace
    xCGs ← xdd[!grepl("floo", BStatus) & grepl("cow|gr", Arm) & Active == 1L, ]
    xCGs[, CowArm := 1L]; xCGs[!grepl("cow", Arm), CowArm := 0L]
    assign(paste0(dd, "CGs", addchar), xCGs)
    # all rejection all arms: rejected vs. nonrejected
    xr ← xdd
    assign(paste0(dd, "r", addchar), xr)
    # all rejection in trad: rejected vs. nonrejected
    xTr ← xdd[grepl("trad", Arm), ]
    assign(paste0(dd, "Tr", addchar), xTr)
    # all rejection in nontrad: rejected vs. nonrejected
    xNTr ← xdd[!grepl("trad", Arm), ]
    assign(paste0(dd, "NTr", addchar), xNTr)
    # all rejection: trad rejecetd vs. nontrad rejected
    xTNTr ← xdd[Rejected == 1L, ]
    xTNTr[, TradArm := 1L]; xTNTr[!grepl("trad", Arm), TradArm := 0L]
    assign(paste0(dd, "TNTr", addchar), xTNTr)
    # all rejection: cow rejecetd vs. noncow rejected
    xCNCr ← xdd[Rejected == 1L, ]
    xCNCr[, CowArm := 1L]; xCNCr[!grepl("cow", Arm), CowArm := 0L]
    assign(paste0(dd, "CNCr", addchar), xCNCr)
    # all rejection: cow rejecetd vs. large grace rejected
    xCLGr ← xdd[grepl("cow|gr", Arm) & Rejected == 1L, ]
    xCLGr[, CowArm := 1L]; xCLGr[!grepl("cow", Arm), CowArm := 0L]
    assign(paste0(dd, "CLGr", addchar), xCLGr)
    # all acceptance: cow accepted vs. noncow accepted
    xCNCa ← xdd[Rejected == 0L, ]
    xCNCa[, CowArm := 1L]; xCNCa[!grepl("cow", Arm), CowArm := 0L]
    assign(paste0(dd, "CNCa", addchar), xCNCa)
    # all acceptance: cow accepted vs. large grace accepted
    xCLGa ← xdd[grepl("cow|gr", Arm) & Rejected == 0L, ]
    xCLGa[, CowArm := 1L]; xCLGa[!grepl("cow", Arm), CowArm := 0L]
    assign(paste0(dd, "CLGa", addchar), xCLGa)
    # group rejection in all arms: rejected vs. nonrejected
    xgr ← xdd
    assign(paste0(dd, "gr", addchar), xgr)
    # group rejection in trad: rejecters vs. nonrejecters
    xTgr ← xdd[grepl("tra", Arm), ]
    assign(paste0(dd, "Tgr", addchar), xTgr)
    # group rejection in nontrad: rejecters vs. nonrejecters
    xNTgr ← xdd[!grepl("tra", Arm), ]
    assign(paste0(dd, "NTgr", addchar), xNTgr)
    # group rejection: trad rejecters vs. nontrad rejecters
    xTNTgr ← xdd[GRejected == 1L, ]
    xTNTgr[, TradArm := 1L]; xTNTgr[!grepl("trad", Arm), TradArm := 0L]
    assign(paste0(dd, "TNTgr", addchar), xTNTgr)
    # individual rejection in all arms: rejected vs. nonrejected
    # individual rejecters vs. all except group rejecters
```

```
    # group rejecters are excluded because they preceded indiv rejection
    xir ← xdd[!grepl("gr", BStatus), ]
    assign(paste0(dd, "ir", addchar), xir)
    # individual rejection in trad: rejecters vs. nonrejecters
    xTir ← xdd[grepl("tra", Arm) & !grepl("gr", BStatus), ]
    assign(paste0(dd, "Tir", addchar), xTir)
    # individual rejection in nontrad: rejecters vs. nonrejecters
    xNTir ← xdd[!grepl("tra", Arm) & !grepl("gr", BStatus), ]
    assign(paste0(dd, "NTir", addchar), xNTir)
    # individual rejection: trad rejecters vs. nontrad rejecters
    xTNTir ← xdd[!grepl("gr", BStatus) & Rejected == 1L, ]
    xTNTir[, TradArm := 1L]; xTNTir[!grepl("trad", Arm), TradArm := 0L]
    assign(paste0(dd, "TNTir", addchar), xTNTir)
    # trad group rejecters vs. nontrad participants
    xTNTgrp ← xdd[(grepl("gr", BStatus) & grepl("trad", Arm) & Rejected == 1L) |
      (grepl("bo", BStatus) & !grepl("trad", Arm)), ]
    xTNTgrp[, TradArm := 1L]; xTNTgrp[!grepl("trad", Arm), TradArm := 0L]
    assign(paste0(dd, "TNTgrp", addchar), xTNTgrp)
    # trad group vs. nontrad group
    xTNTrandom ← xdd
    xTNTrandom[, TradArm := 1L]; xTNTrandom[!grepl("trad", Arm), TradArm := 0L]
    assign(paste0(dd, "TNTrandom", addchar), xTNTrandom)
  }
}

# data names: ..af, ..rf  (full), ..at, ..rt (drop 2 loan receivers), ..aj, ..rj (drop gro
# data to use: datalist (full), datalist1 (drop 2 loan receivers), datalist2 (drop group m
library(coin)
PM ← vector(mode = "list", length = 3)
for (k in 1:3) {
  addchar ← c("f", "t", "j")[k]
  dataList ← eval(parse(text=paste0("datalist", c("", 1:2))[k]))
  if (addchar == "j") M ← 9 else M ← length(selection.criteria)
  Pm ← vector(mode = "list", length = M)
  for (m in 1:M) {
    set.seed(100+m)
    if (grepl("^Attrited$", addtofilename[m]))
      DataList ← gsub("$", paste0("a", addchar), dataList) else
    if (grepl("^AttritedInTrad", addtofilename[m]))
      DataList ← gsub("$", paste0("Ta", addchar), dataList) else
    if (grepl("^AttritedInNonTrad", addtofilename[m]))
      DataList ← gsub("$", paste0("NTa", addchar), dataList) else
    if (grepl("^TradNonTradAttrited$", addtofilename[m]))
      DataList ← gsub("$", paste0("TNTa", addchar), dataList) else
    if (grepl("^NonFloodAttrited$", addtofilename[m]))
      DataList ← gsub("$", paste0("NFa", addchar), dataList) else
    if (grepl("^NonFloodAttritedInTrad$", addtofilename[m]))
      DataList ← gsub("$", paste0("NFTa", addchar), dataList) else
    if (grepl("^NonFloodAttritedInNonTrad$", addtofilename[m]))
      DataList ← gsub("$", paste0("NFNTa", addchar), dataList) else
    if (grepl("^NonFloodTradNonTradAttrited$", addtofilename[m]))
      DataList ← gsub("$", paste0("NFTNTa", addchar), dataList) else
    if (grepl("^NonFloodAttritedCowN", addtofilename[m]))
      DataList ← gsub("$", paste0("NFCNCa", addchar), dataList) else
    if (grepl("^NonFloodAttritedCowL", addtofilename[m]))
      DataList ← gsub("$", paste0("NFCGa", addchar), dataList) else
```

```
  if (grepl("^Active$", addtofilename[m]))
    DataList ← gsub("$", paste0("s", addchar), dataList) else
  if (grepl("^ActiveInTrad", addtofilename[m]))
    DataList ← gsub("$", paste0("Ts", addchar), dataList) else
  if (grepl("^ActiveInNonTrad", addtofilename[m]))
    DataList ← gsub("$", paste0("NTs", addchar), dataList) else
  if (grepl("^ActiveTradNonTrad", addtofilename[m]))
    DataList ← gsub("$", paste0("TNTs", addchar), dataList) else
  if (grepl("^ActiveCowN", addtofilename[m]))
    DataList ← gsub("$", paste0("CNCs", addchar), dataList) else
  if (grepl("^ActiveCowL", addtofilename[m]))
    DataList ← gsub("$", paste0("CGs", addchar), dataList) else
  if (grepl("^Random", addtofilename[m]))
    DataList ← gsub("$", paste0("TNTrandom", addchar), dataList) else
  if (grepl("^Rejected$", addtofilename[m]))
    DataList ← gsub("$", paste0("r", addchar), dataList) else
  if (grepl("^Rej.*InTrad$", addtofilename[m]))
    DataList ← gsub("$", paste0("Tr", addchar), dataList) else
  if (grepl("^Rej.*InNonTrad$", addtofilename[m]))
    DataList ← gsub("$", paste0("NTr", addchar), dataList) else
  if (grepl("^TradNonTradR", addtofilename[m]))
    DataList ← gsub("$", paste0("TNTr", addchar), dataList) else
  if (grepl("^GRejected$", addtofilename[m]))
    DataList ← gsub("$", paste0("gr", addchar), dataList) else
  if (grepl("^GRej.*InTrad$", addtofilename[m]))
    DataList ← gsub("$", paste0("Tgr", addchar), dataList) else
  if (grepl("^GRej.*InNonTrad$", addtofilename[m]))
    DataList ← gsub("$", paste0("NTgr", addchar), dataList) else
  if (grepl("^TradNonTradGR", addtofilename[m]))
    DataList ← gsub("$", paste0("TNTgr", addchar), dataList) else
  if (grepl("^IRejected$", addtofilename[m]))
    DataList ← gsub("$", paste0("ir", addchar), dataList) else
  if (grepl("^IRej.*InTrad$", addtofilename[m]))
    DataList ← gsub("$", paste0("Tir", addchar), dataList) else
  if (grepl("^IRej.*InNonTrad$", addtofilename[m]))
    DataList ← gsub("$", paste0("NTir", addchar), dataList) else
  if (grepl("^TradNonTradIR", addtofilename[m]))
    DataList ← gsub("$", paste0("TNTir", addchar), dataList) else
  if (grepl("^GRejectedTradPar", addtofilename[m]))
    DataList ← gsub("$", paste0("TNTgrp", addchar), dataList) else
  if (grepl("^RejectedCowN", addtofilename[m]))
    DataList ← gsub("$", paste0("CNCr", addchar), dataList) else
  if (grepl("^RejectedCowLa", addtofilename[m]))
    DataList ← gsub("$", paste0("CLGr", addchar), dataList) else
  if (grepl("^AcceptedCowN", addtofilename[m]))
    DataList ← gsub("$", paste0("CNCa", addchar), dataList) else
  if (grepl("^AcceptedCowLa", addtofilename[m]))
    DataList ← gsub("$", paste0("CLGa", addchar), dataList) else
    DataList ← gsub("$", addchar, dataList)
  pmresults ← permmedian ← vector(mode = "list", length(vartobetested))
  for (i in 1:length(vartobetested)) {
    # if specific arm is selected, Arm is not compared in permutation
    if (grepl("Trad$|TradArm|Cow", addtofilename[m]) &
      vartobetested[i] == "Arm") next
    pmdata ← get(DataList[i])
    # drop NAs in vartobetested[i]
```

```
    pmdata ← pmdata[!is.na(eval(parse(text=vartobetested[i]))), ]
    # NULL if vartobetested[i] has uniform values (otherwise returns an error)
    if (length(unique(unlist(pmdata[, vartobetested[i], with = F]))) == 1)
      pmresults[[i]] ← NULL else
      pmresults[[i]] ← independence_test(eval(parse(text=
        paste(vartobetested[i], "~ as.factor(", selection.criteria[m], ")")
        )),
        data = pmdata,
        distribution = approximate(nresample=PermRepTimes))
    if (!TestMedian) next
    if (vartobetested[i] == "Arm" | length(unique(unlist(pmdata[, vartobetested[i], with
      permmedian[[i]] ← NULL else
      permmedian[[i]] ← median_test(eval(parse(text=
        paste(vartobetested[i], "~ as.factor(", selection.criteria[m], ")"))),
        data = pmdata,
        mid.score = "0.5",
        distribution = approximate(nresample=PermRepTimes))
  }
  #pmresults[[1]]@statistic@teststatistic
  Pmtresults ← NULL
  for (i in 1:length(vartobetested))
  {
    if (grepl("Trad$|TradArm|Cow", addtofilename[m]) &
      vartobetested[i] == "Arm") next
    z ← get(DataList[i])
    z ← z[!is.na(eval(parse(text=vartobetested[i]))), ]
    if (vartobetested[i] == "Arm") {
      Pmtresults ← rbind(Pmtresults,
        c(vartobetested[i],
          sum(!grepl("trad", unlist(z[eval(parse(text = selection.criteria[m])) == 0L,
            vartobetested[i], with = F])))/
            nrow(z[eval(parse(text = selection.criteria[m])) == 0L, ]),
          sum(!grepl("trad", unlist(z[eval(parse(text = selection.criteria[m])) == 1L,
            vartobetested[i], with = F])))/
            nrow(z[eval(parse(text = selection.criteria[m])) == 1L, ]),
          midpvalue(pmresults[[i]]),
          pvalue_interval(pmresults[[i]])))
    } else if (length(unique(unlist(z[, vartobetested[i], with = F]))) == 1)
    {
      # if both groups have no different values,
      # use 0 for all zero entries or 1 for unique nonzero entries
      if (allzerovalues ← unique(unlist(z[, vartobetested[i], with = F])) == 0)
        Pmtresults ← rbind(Pmtresults,
          c(vartobetested[i], 0, 0, rep(NA, 3))) else
        Pmtresults ← rbind(Pmtresults,
          c(vartobetested[i], 1, 1, rep(NA, 3)))
    } else {
      Pmtresults ← rbind(Pmtresults,
        c(vartobetested[i],
          mean(unlist(z[eval(parse(text = selection.criteria[m])) == 0L,
            vartobetested[i], with = F]), na.rm = T),
          mean(unlist(z[eval(parse(text = selection.criteria[m])) == 1L,
            vartobetested[i], with = F]), na.rm = T),
          midpvalue(pmresults[[i]]),
          pvalue_interval(pmresults[[i]])))
      if (TestMedian)
```

```
          Pmtresults ← rbind(Pmtresults,
            c("",
              median(unlist(z[eval(parse(text = selection.criteria[m])) == 0L,
                vartobetested[i], with = F]), na.rm = T),
              median(unlist(z[eval(parse(text = selection.criteria[m])) == 1L,
                vartobetested[i], with = F]), na.rm = T),
              midpvalue(permmedian[[i]]),
              pvalue_interval(permmedian[[i]])
            ))
        }
    }
    Pmtresults ← data.table(Pmtresults)
    setnames(Pmtresults, c("variables", paste0(c("Non", ""), selection.criteria[m]),
      "p-value.mid", "p-value.lower", "p-value.upper"))
    Pmtresults[grepl("Impute", variables),
      variables := gsub("To.*", "LivestockValue", variables)]
    cols ← grepout("p|er|ttr|eje|TradArm|CowA|Acti", colnames(Pmtresults))
    Pmtresults[, (cols) := lapply(.SD, as.numeric), .SDcols = cols]
    Pmtresults[, (cols) := lapply(.SD, formatC, digits = 3, format = "f"), .SDcols = cols
    cols ← grepout("ed$|TradArm|CowA", colnames(Pmtresults))
    Pmtresults[grepl("Ass|Liv|NetV", variables),
      (cols) := lapply(.SD, function(x) formatC(as.numeric(x), digits = 0, format = "f")),
      .SDcols = cols]
    setcolorder(Pmtresults, c("variables", paste0(c("Non", ""), selection.criteria[m]),
      "p-value.lower", "p-value.mid", "p-value.upper"))
    obs0L ← nrow(get(DataList[1])[eval(parse(text = selection.criteria[m])) == 0L, ])
    obs1L ← nrow(get(DataList[1])[eval(parse(text = selection.criteria[m])) == 1L, ])
    nobs ← t(c(NA, obs0L, obs1L, NA, obs1L/(obs0L+obs1L), NA))
    Pmtresults[, variables := paste0("\\makebox[2.5cm]{\\hfill ", variables, "}")]
    Pmtresults0 ← rbind(Pmtresults, nobs, use.names = F)
    Pmtresults0[nrow(Pmtresults0), variables := "\\makebox[2.5cm]{\\hfill n}"]
    Pm[[m]] ← Pmtresults0
    if (grepl("InNon|InTra|^TradNon|Cow", addtofilename[m]))
      Pmtresults ← Pmtresults[!grepl("Arm", variables), ]
    pmt ← latextab(as.matrix(Pmtresults),
      hleft = "\\scriptsize\\hfil$",
      hcenter = c(3, rep(1.5, ncol(Pmtresults)-1)),
      hright = "$",
      headercolor = "gray80", adjustlineskip = "-.2ex", delimiterline= NULL,
      alternatecolor = "gray90")
    pmt ← rbind(pmt[1:(nrow(pmt)-1), , drop = F],
        paste(c("\\makebox[2.5cm]{\\hfill n}",
         obs0L, obs1L, paste0("\\multicolumn{3}{1}{\\makebox[4.5cm]{\\scriptsize (rate: "
         formatC(obs1L/(obs0L+obs1L), digits = 3, format = "f"), ")\\hfill}}")),
         collapse = " & "),
        pmt[nrow(pmt), , drop = F]
      )
    write.tablev(pmt,
      paste0(pathsaveHere, addtofilename[m],
        c("Full", "", "DropGroupRejecters")[k], "PermutationTestResultso800.tex")
    , colnamestrue = F)
  }
  names(Pm) ← addtofilename[1:M]
  PM[[k]] ← Pm
}
names(PM) ← c("Full", "Drop2LoanReceivers", "DropGroupRejecters")
```

```
saveRDS(PM, paste0(pathsaveHere, "AllPermutationTestResults.rds"))
PM ← readRDS(paste0(pathsaveHere, "AllPermutationTestResults.rds"))
# indiv rejecters
Irej ← c("IRejectedInTrad", "IRejectedInNonTrad", "^IRejected$")
ir12 ←  cbind(
    PM[[2]][[grep(Irej[1], addtofilename)]][, c(1:3, 5)],
    PM[[2]][[grep(Irej[2], addtofilename)]][, c(2:3, 5)])
setnames(ir12, c("variables", 1:(ncol(ir12)-1)))
ir3 ← PM[[2]][[grep(Irej[3], addtofilename)]][, c(1:3, 5)]
setnames(ir3, c("variables", 10+1:(ncol(ir3)-1)))
ir3rows ← data.table(variables = ir3[, variables])
setkey(ir12, variables)
setkey(ir3, variables)
ir123 ← ir12[ir3]
ir123 ← ir123[ir3rows]
setnames(ir123, c("variables", paste0("v", 1:(ncol(ir123)-1))))
for (i in paste0("v", c(3, 6, 9)))
  ir123[nrow(ir123), (i) :=
    paste0("(\\mbox{rate }", formatC(as.numeric(eval(parse(text=i))), digits = 3, format =
#cnm ← t(c("\\makebox[3cm]{\\hfil variables}",
#  paste0("\\makebox[1.5cm]{\\hfil ", rep(c("Yes", "No", "$p$ value"), 3), "}")))
cnm ← t(c("\\makebox[2.5cm]{\\hfil }",
  paste0("\\makebox[1.2cm]{(", 1:(ncol(ir123)-1), ")}")))
irj ← as.matrix(rbind(cnm, ir123, use.names = F))
irj[is.na(irj)] ← ""
colnames(irj) ← c("variables", rep(c("Not rejected", "Rejected", "$p$ value"), 3))
irj ← latextab(irj,
  hleft = "\\scriptsize\\hfil$",
  hcenter = c(2.5, rep(1.2, ncol(Pmtresults)-1)),
  hright = "$",
  headercolor = "gray80", adjustlineskip = "-.2ex", delimiterline= NULL,
  alternatecolor = "gray90",
  addseparatingcols = c(3, 6),
  separatingcolwidth = rep(.1, 2),
  separatingcoltitle = c("\\textsf{Traditional} arm", "non-\\textsf{Traditional} arms", "A
  addsubcoltitlehere = T
  )
write.tablev(irj,
  paste0(pathsaveHere, "IndividualRejectionTestResults.tex")
, colnamestrue = F)
# active
Suv ← c("Acc.*NonCow", "Act.*NonCow")
sv12 ←  cbind(
    PM[[2]][[grep(Suv[1], addtofilename)]][, c(1, 3, 2, 5)],
    PM[[2]][[grep(Suv[2], addtofilename)]][, c(3, 2, 5)])
setnames(sv12, c("variables", paste0("v", 1:(ncol(sv12)-1))))
for (i in paste0("v", c(3, 6)))
  sv12[nrow(sv12), (i) :=
    paste0("(\\mbox{rate }", formatC(as.numeric(eval(parse(text=i))), digits = 3, format =
cnm ← t(c("\\makebox[2.5cm]{\\hfil }",
  paste0("\\makebox[1.2cm]{(", 1:(ncol(sv12)-1), ")}")))
suv ← as.matrix(rbind(cnm, sv12, use.names = F))
colnames(suv) ← c("variables", rep(c("Cattle arm", "Other arms", "$p$ value"), 2))
suv ← latextab(suv,
  hleft = "\\scriptsize\\hfil$",
  hcenter = c(2.5, rep(1.2, ncol(suv)-1)),
```

```
  hright = "$",
  headercolor = "gray80", adjustlineskip = "-.2ex", delimiterline= NULL,
  alternatecolor = "gray90",
  addseparatingcols = 3,
  separatingcolwidth = .1,
  separatingcoltitle = c("Borrowers", "Non-attriting borrowers"),
  addsubcoltitlehere = T
  )
write.tablev(suv,
  paste0(pathsaveHere, "CowVsNonCowTestResults.tex")
, colnamestrue = F)

HeaderDescription[11:16] ← c(
   "Permutation test results of attrition among borrowers",
     "Permutation test results of attrition among traditional arm borrowers",
     "Permutation test results of attrition among non-traditional arm borrowers",
     "Permutation test results of non-attriting members of traditional and non-traditional
     "Permutation test results of non-attriting members of cattle and all other arm borrov
     "Permutation test results of non-attriting members of cattle and large grace arm borr
    )
TabLabel1 ← paste("tab1", HeaderDescription)
% TabLabelStrings is from RejectionTestsMainText4.rnw
% I will modify the table headers for survival/surviving members to "non-attriting borrowe
%   "survival$", "surviving.*race$", "surv.*other"

TabLabelStrings ←
c(
  "Per.* of rejection$", "of rej.*ng traditional arm$", "of rej.*ng non-traditional arm$"
  "of rej.*l vs", "p rejection$", "p rej.* traditional arm$",
  "p rej.*g non-traditional arm$", "p rejecters.*vs", "l rejection$",
  "l rej.*g tra", "l rej.*g non-", "l rej.*vs",
  "bo.*non-ca",
  "of attrition$", "of attri.* traditional arm$",
  "of attri.* non-traditional arm$", "of attri.* of",
  "active status$", "active.*race$", "active.*other"
  )

tb1 ← "\\hfil\\begin{minipage}[t]{14cm}\\hfil\\textsc{\\normalsize Table \\refstepcounter
tb2 ← "}\\\\\\\\setlength{\\tabcolsep}{.5pt}\\setlength{\\baselineskip}{8pt}\\renewcommand{
tb3 ← "}};\\end{tikzpicture}\\\\\\\\begin{tabular}{>{\\hfill\\scriptsize}p{1cm}<{}>{\\hfill
#tb4 ← ". Step-down method is used to adjust for multiple testing of a multi-factor group
\\\\& 3. & See the footnote of \\textsc{Table \\ref{tab MainTextIRjecters}} for descriptio
tb42 ← ". Step-down method is used to adjust for multiple testing of a multi-factor group
tb41 ← paste(tb42, "See the footnote of \\textsc{Table \\ref{tab MainTextIRjecters}} for
tb43 ← ".\\\\& 2. &  See footnotes of \\textsc{Table \\ref{tab1 Permutation test results
tb44 ← ".\\\\& 2. &  See footnotes of \\textsc{Table \\ref{tab1 Permutation test results
for (k in 1:3)
  for (i in 1:length(HeaderDescription))
    assign(paste0("Tb", k, i),
      paste0(
        tb1
        ,
        HeaderDescription[i]
        ,
        paste0("\\label{", get(paste0("TabLabel", k))[i], "}")
        ,
```

```
        tb2
        ,
        paste0(pathsavePerm, addtofilename[i], c("", "Full", "DropGroupRejecters")[k],
          "PermutationTestResultso800.tex")
        ,
        tb3
        ,
        PermRepTimes
        ,
        #if (i %in% c(1, 5, 11, 17, 21, 25)) tb42 else tb41
        if (i==17)
        paste0(tb42, TabVariableDescription, PrefTestsDefinitions1, "\\end{tabular}\\end{r
        #if (i==18) tb41 else
        if (i %in% c(1, 5, 11, 18, 21, 25)) tb43 else
        tb44
        )
      )

TabLabelStrings[19:20] ← c("active.*race arms$", "activ.*other arms$")

for (i in 14:length(TabLabelStrings)) {
  ii ← grep(TabLabelStrings[i], TabLabel1)
  if (grepl("active", TabLabelStrings[i])){
   tblatt ← eval(parse(text=paste0("Tb1", ii)))
   cat(gsub("active\\\\", "attrited or rejected (NonActive) and other (Active) borrowers\\
     tblatt))
   rm(tblatt)
  } else
  if (grepl("active|activ.*o", TabLabelStrings[i])) {
   tblatt ← eval(parse(text=paste0("Tb1", ii)))
   cat(gsub("active members", "non-attriting borrowers",
     tblatt))
    rm(tblatt)
  } else
   cat(eval(parse(text=paste0("Tb1", ii))))
}
```

TABLE 1: PERMUTATION TEST RESULTS OF ATTRITION

variables	NonAttrited	Attrited	p-value.lower	p-value.mid	p-value.upper
HeadLiteracy	0.115	0.130	(60.9)	(67.0)	(73.1)
HeadAge	37.996	38.598	(59.1)	(59.3)	(59.5)
HHsize	4.178	4.272	(54.2)	(55.5)	(56.8)
Arm	0.789	0.652	(0.0)	(0.0)	(0.0)
FloodInRd1	0.493	0.527	(50.2)	(54.0)	(57.7)
HAssetAmount	763	741	(83.3)	(83.4)	(83.6)
PAssetAmount	1109	2181	(10.5)	(10.5)	(10.5)
LivestockValue	5124	5000	(92.4)	(96.2)	(100.0)
NumCows	0.256	0.250	(92.3)	(96.2)	(100.0)
NetValue	6141	5960	(90.7)	(90.7)	(90.7)
BroadNetValue	6780	6652	(93.5)	(93.5)	(93.5)
RiskPrefVal	110	128	(0.0)	(0.0)	(0.0)
TimePref1Val	382	404	(28.2)	(29.4)	(30.6)
TimePref2Val	490	486	(82.5)	(86.8)	(91.2)
PresentBias	0.459	0.531	(30.0)	(33.7)	(37.4)
n	684	92	(rate: 0.119)		

Source: Estimated with GUK administrative and survey data.

Notes: 1. R's package coin is used for baseline mean covariates to conduct approximate permutation tests. Number of repetition is set to 100000.

2. See footnotes of TABLE **??**.

TABLE 2: PERMUTATION TEST RESULTS OF ATTRITION AMONG TRADITIONAL ARM

variables	NonAttrited	Attrited	p-value.lower	p-value.mid	p-value.upper
HeadLiteracy	0.118	0.000	(1.8)	(3.2)	(4.6)
HeadAge	38.497	38.125	(84.8)	(85.2)	(85.6)
HHsize	4.167	3.750	(13.7)	(14.7)	(15.6)
FloodInRd1	0.479	0.387	(32.6)	(37.7)	(42.8)
HAssetAmount	702	842	(47.0)	(47.3)	(47.5)
PAssetAmount	997	926	(81.3)	(81.3)	(81.4)
LivestockValue	4722	2581	(28.3)	(33.6)	(38.8)
NumCows	0.236	0.129	(28.5)	(33.6)	(38.8)
NetValue	5625	3633	(41.0)	(41.0)	(41.0)
BroadNetValue	6206	4343	(44.5)	(44.5)	(44.5)
RiskPrefVal	113	131	(1.2)	(1.5)	(1.8)
TimePref1Val	371	391	(49.8)	(54.5)	(59.3)
TimePref2Val	485	470	(47.8)	(54.1)	(60.5)
PresentBias	0.462	0.522	(50.3)	(57.9)	(65.6)
n	144	32	(rate: 0.182)		

Source: Estimated with GUK administrative and survey data.

Notes: 1. R's package coin is used for baseline mean covariates to conduct approximate permutation tests. Number of repetition is set to 100000.

2. See footnotes of TABLE **??**.

TABLE 3: PERMUTATION TEST RESULTS OF ATTRITION AMONG NON-TRADITIONAL ARM

variables	NonAttrited	Attrited	p-value.lower	p-value.mid	p-value.upper
HeadLiteracy	0.115	0.200	(3.6)	(5.1)	(6.5)
HeadAge	37.862	38.850	(47.0)	(47.2)	(47.4)
HHsize	4.181	4.550	(6.1)	(6.4)	(6.7)
FloodInRd1	0.497	0.600	(10.2)	(12.0)	(13.8)
HAssetAmount	779	688	(47.3)	(47.5)	(47.7)
PAssetAmount	1139	2829	(9.3)	(9.3)	(9.3)
LivestockValue	5232	6531	(49.8)	(53.0)	(56.3)
NumCows	0.262	0.327	(49.9)	(53.1)	(56.4)
NetValue	6278	7162	(65.2)	(65.2)	(65.2)
BroadNetValue	6933	7846	(64.7)	(64.7)	(64.7)
RiskPrefVal	110	125	(2.2)	(2.8)	(3.3)
TimePref1Val	385	415	(27.4)	(29.3)	(31.2)
TimePref2Val	491	500	(66.1)	(71.5)	(77.0)
PresentBias	0.458	0.538	(43.0)	(48.8)	(54.7)
n	540	60	(rate: 0.100)		

Source: Estimated with GUK administrative and survey data.

Notes: 1. R's package coin is used for baseline mean covariates to conduct approximate permutation tests. Number of repetition is set to 100000.

2. See footnotes of TABLE **??**.

TABLE 4: PERMUTATION TEST RESULTS OF ATTRITERS OF TRADITIONAL AND NON-TRADITIONAL ARMS

variables	NonTradArm	TradArm	p-value.lower	p-value.mid	p-value.upper
HeadLiteracy	0.200	0.000	(0.3)	(0.5)	(0.7)
HeadAge	38.850	38.125	(76.8)	(77.2)	(77.6)
HHsize	4.550	3.750	(2.1)	(2.3)	(2.6)
FloodInRd1	0.600	0.387	(4.8)	(6.2)	(7.5)
HAssetAmount	688	842	(52.2)	(52.5)	(52.8)
PAssetAmount	2829	926	(83.4)	(83.4)	(83.4)
LivestockValue	6531	2581	(17.0)	(20.3)	(23.7)
NumCows	0.327	0.129	(17.1)	(20.4)	(23.7)
NetValue	7162	3633	(45.5)	(45.5)	(45.5)
BroadNetValue	7846	4343	(46.1)	(46.1)	(46.1)
RiskPrefVal	125	131	(39.1)	(48.5)	(57.9)
TimePref1Val	415	391	(50.7)	(58.0)	(65.3)
TimePref2Val	500	470	(29.7)	(36.1)	(42.5)
PresentBias	0.538	0.522	(77.8)	(88.9)	(100.0)
n	60	32	(rate: 0.348)		

Source: Estimated with GUK administrative and survey data.

Notes: 1. R's package coin is used for baseline mean covariates to conduct approximate permutation tests. Number of repetition is set to 100000.

2. See footnotes of TABLE **??**.

TABLE 5: PERMUTATION TEST RESULTS OF ACTIVE STATUS

variables	NonActive	Active	p-value.lower	p-value.mid	p-value.upper
HeadLiteracy	0.104	0.123	(38.9)	(42.7)	(46.5)
HeadAge	37.835	38.159	(68.8)	(69.0)	(69.1)
HHsize	4.072	4.236	(14.9)	(15.3)	(15.7)
Arm	0.581	0.850	(0.0)	(0.0)	(0.0)
FloodInRd1	0.548	0.477	(6.6)	(7.2)	(7.9)
HAssetAmount	707	781	(32.1)	(32.1)	(32.2)
PAssetAmount	1440	1154	(55.0)	(55.0)	(55.0)
LivestockValue	3714	5642	(5.2)	(5.6)	(6.0)
NumCows	0.186	0.282	(5.0)	(5.4)	(5.8)
NetValue	4616	6719	(5.6)	(5.6)	(5.6)
BroadNetValue	5290	7353	(6.4)	(6.4)	(6.4)
RiskPrefVal	120	109	(0.0)	(0.0)	(0.0)
TimePref1Val	388	382	(60.3)	(61.4)	(62.5)
TimePref2Val	476	494	(13.7)	(14.6)	(15.5)
PresentBias	0.520	0.446	(7.9)	(8.7)	(9.6)
n	222	554	(rate: 0.714)		

Source: Estimated with GUK administrative and survey data.
Notes: 1. R's package coin is used for baseline mean covariates to conduct approximate permutation tests. Number of repetition is set to 100000.
2. See footnotes of TABLE **??**.

TABLE 6: PERMUTATION TEST RESULTS OF ACTIVE MEMBERS OF CATTLE AND LARGE GRACE ARMS

variables	NonCowArm	CowArm	p-value.lower	p-value.mid	p-value.upper
HeadLiteracy	0.106	0.150	(23.6)	(27.1)	(30.6)
HeadAge	38.481	37.973	(64.4)	(64.7)	(64.9)
HHsize	4.181	4.102	(57.3)	(58.9)	(60.4)
FloodInRd1	0.352	0.459	(4.6)	(5.5)	(6.3)
HAssetAmount	798	785	(90.5)	(90.6)	(90.7)
PAssetAmount	1480	753	(0.3)	(0.3)	(0.3)
LivestockValue	5375	3425	(12.6)	(13.9)	(15.2)
NumCows	0.269	0.171	(12.4)	(13.7)	(15.0)
NetValue	6740	4118	(5.4)	(5.4)	(5.4)
BroadNetValue	7448	4688	(4.4)	(4.4)	(4.4)
RiskPrefVal	112	108	(24.4)	(26.6)	(28.8)
TimePref1Val	373	412	(2.1)	(2.2)	(2.3)
TimePref2Val	479	515	(2.1)	(2.4)	(2.7)
PresentBias	0.462	0.466	(90.9)	(95.5)	(100.0)
n	160	147	(rate: 0.479)		

Source: Estimated with GUK administrative and survey data.
Notes: 1. R's package coin is used for baseline mean covariates to conduct approximate permutation tests. Number of repetition is set to 100000.
2. See footnotes of TABLE **??**.

TABLE 7: PERMUTATION TEST RESULTS OF NON-ATTRITING BORROWERS OF CATTLE AND ALL OTHER ARMS

variables	NonCowArm	CowArm	p-value.lower	p-value.mid	p-value.upper
HeadLiteracy	0.113	0.150	(24.6)	(27.5)	(30.4)
HeadAge	38.226	37.973	(78.6)	(78.8)	(79.0)
HHsize	4.285	4.102	(16.6)	(17.1)	(17.7)
FloodInRd1	0.484	0.459	(56.1)	(59.5)	(62.9)
HAssetAmount	780	785	(95.6)	(95.6)	(95.7)
PAssetAmount	1298	753	(2.8)	(2.8)	(2.8)
LivestockValue	6437	3425	(1.5)	(1.6)	(1.8)
NumCows	0.322	0.171	(1.5)	(1.6)	(1.8)
NetValue	7658	4118	(0.7)	(0.7)	(0.7)
BroadNetValue	8315	4688	(0.7)	(0.7)	(0.7)
RiskPrefVal	109	108	(64.9)	(68.3)	(71.6)
TimePref1Val	371	412	(0.5)	(0.5)	(0.5)
TimePref2Val	486	515	(2.7)	(3.0)	(3.3)
PresentBias	0.439	0.466	(55.8)	(59.1)	(62.4)
n	407	147	(rate: 0.265)		

Source: Estimated with GUK administrative and survey data.
Notes: 1. R's package coin is used for baseline mean covariates to conduct approximate permutation tests. Number of repetition is set to 100000.
2. See footnotes of TABLE **??**.

TABLE 1 shows results from tests of independence between attriters and nonattriters. Attrition is defined as attrition from household surveys, not from the lending program. We see the moderate rate of attrition is not correlated with household level characteristicsat the conventional p value level. Productive asset amounts seem to differ between attriters and nonattriters at $p = .105$, with the

former being larger than the latter. This positive attrition selection can cause underestimation of impacts, if the asset values are positively correlated with entrepreneurial capacity. We also see that the attriters are less risk tolerant in terms of minimum expected payoff to choose a risky option in RiskPrefVal. Table 2 shows attrition in the traditional arm. Household heads of attriters are relatively less literate than nonattriters. We observe the traditional arm attriters are less risk tolerant the nonattriters. Table 3 compares attriters and nonattriters in the non-traditional arm. Unlike traditional arm attriters, non-traditional arm attriters have more literate household heads, have a larger household size, are more exposed to floods, and have larger productive assets. The traditional arm attriters may be less entrepreneurial, if anything, so their attrition may upwardly bias the positive gains of the arm, hence understate the impacts of non-traditional arm. These are explicitly shown in Table 4 where we compare attriters of traditional and non-traditional arms. Overall, attrition may have attenuated the impacts but is not likely to have inflated them.[†] We observe the non-traditional arm attriters are also less risk tolerant than the nonattriters.

For the microfinance institutions (MFIs), attrition of the loan receiving members poses a threat to their business continuation. Financial institutions often use observable characteristics, such as collateralisable assets, and easily surveyed chracteristics, such as job experiences and schooling of borrowers, and are likely to lend if the assets levels are greater and the borrowers have relevant job experiences and more schooling. We first examine if such screening variables have any predictive power in terms of loan rejection or borrower attrition under our lending. Table 5 compares potential MFI targets (nonattriting borrowers, noted as Active) vs. non-targets (attriting borrowers or loan rejecters, noted as NonSurvived) in all arms. It shows potential targets at the baseline have larger values in livestock and greater number of cattle, and are less affected by the flood, which conforms the conventional wisdom of lenders in using these aspects in their loan decisions. We also see that more risk torelant members are likely to be borrowers and do not attrit. Next, we examine if the relationship of having "less favourable" values in these characteristics and attrition is mitigated under various loan characteristics. In Table 6, we restrict our attention to the potential MFI targets, or the nonattriting borrowers, and compare between cattle and large grace arms, whose difference is efffectively the presence of managerial supports that the former provides. Comparing against the large grace arm, nonattriting borrowers of the cattle arm are more exposed to the flood (p = .055), have less productive assets (p = .003), have lower net asset values (p = .046), and have fewer livestock (p = .139). This shows that the smaller livestock holders or individuals with less experienced in livestock are encouraged to participate and continue to operate in the cattle arm that has a managerial support program, with all other features being equal. This is consistent with our analysis of participation in Table **??** which weakly hints that the cattle arm's managerial support programs may have encouraged participation of inexperienced or lower asset holders. This also underscores our interpretation that the current impact estimates may be downwardly biased, if any, as people who would otherwise attrit or reject in the cattle arm stayed on. This result is confirmed with lower p values due to a larger sample size when we compare the nonattriting borrowers between cattle arm with all other arms in Table **??**. At the baseline, cattle arm nonattriting borrowers have smaller baseline livestock holding (p value = .016) and smaller baseline net asset holding (p value = .007) than other arms' nonattriting borrowers.

[†] So one can employ the Lee bounds for stronger results, but doing so will give us less precision and require more assumptions. We will not use the Lee bounds [we can show them if necessary].