swfdr_base.R

Nathan (Nat) Goodman

June 1, 2017

Contents

run

```
run=function(...) {
  init(...);                        # process parameters & other initialization
  doit();                           # do it!
  saveit(save.rdata,save.txt);      # optionally save parameters and results
  fini();                           # final cleanup if any
}
```

init

```
init=function(
  ## simulation parameters
  prop.true=seq(.1,.9,by=.2),    # fraction of cases where where there is a real effect
                                 #   ie, alternative hypothesis is true
  m=1e4,                         # number of iterations
  n=16,                          # sample size
  d=c(.25,.50,.75,1,2),          # standardized effect size (aka Cohen's d)
  pwr=NA,                        # power. if set, n or d adjusted to achieve power
  sig.level=0.05,                # for power calculations
  pval.plot=c(.001,.01,.03,.05,.1), # pvalues for which we plot results
  ## program parameters, eg, for output files, error messages, etc.
  scriptname='swfdr_base',       #
  datadir=file.path('data',scriptname), # directory for data files
  figdir=file.path('figure',scriptname), # directory for plots
  ## program control
  save=F,                        # shorthand for save.rdata & save.plot
  save.rdata=save,               # save params and results in RData format
  save.txt=F,                    # save results in txt format - big & slow!
```

```
  save.plot=save,                # save plots
  clean=F,                       # remove contents of datadir and figdir and start fresh
  clean.data=clean,              # remove contents of datadir and start fresh
  clean.fig=clean,               # remove contents of figdir and start fresh
  end=NULL                       # placeholder for last parameter
  ) {
  ## make sure m>=2. program crashes for smaller values
  if (m<2) stop("m must be 2 or more");
  ## if pwr is set, program computes d. error if user sets d, too
  if (!any(is.na(pwr))) {
    if (!missing(d))
      stop("if pwr set, 'd' must not be set");
    d=NA;                              # override default value
  }
  ## compute parameter grid, tqking into account which power-related parameters are set
  cases=expand.grid(prop.true=prop.true,m=m,n=n,d=d,pwr=pwr,sig.level=sig.level);
  cases=power_grid(cases);
  ## clean and create output directories as needed
  if (clean.data) unlink(datadir,recursive=T);
  dir.create(datadir,recursive=TRUE,showWarnings=FALSE);
  if (clean.fig) unlink(figdir,recursive=T);
  dir.create(figdir,recursive=TRUE,showWarnings=FALSE);
  ## at end, assign global parameters to global variables
  ##   don't do it earlier because it's too easy to confuse local and
  ##   global variables with the same names
  prop.true<<-prop.true;
  m<<-m;
  n<<-n;
  d<<-d;
  pwr<<-pwr;
  sig.level<<-sig.level;
  pval.plot<<-pval.plot;
  scriptname<<-scriptname;
  datadir<<-datadir;
  figdir<<-figdir;
  save.rdata<<-save.rdata;
  save.txt<<-save.txt;
  save.plot<<-save.plot;
  clean.data<<-clean.data;         # not really needed, since only used in init
  clean.fig<<-clean.fig;           # not really needed, since only used in init
  cases<<-cases;
  invisible(cases);
}
```

doit

```
doit=function() {
  dosim();                       # do simulation
  dointerp();                    # interpolate at fixed pvals
  doplot(save.plot);             # plot results and optionally save plots
}
```

fini

```
fini=function() {}
```

dosim

```
dosim=function() {
  ## call sim_one on each case and combine results into data frame
  ## the complicated one-liner below is a good idiom for doing this
  sim=do.call(rbind,apply(cases,1,
    function(case) do.call("sim_one",as.list(case)[c('prop.true','m','n','d')])));
  sim<<-sim;
  invisible(sim);
}
```

sim_one

```
sim_one=function(prop.true,m,n,d) {
  ## draw m pairs of samples
  ## each group is matrix whose rows are cases and columns are samples
  ## group0 is control
  group0=replicate(m,rnorm(n,mean=0));
  ## group1 contains num.false samples with effect=0, and num.true with effect=1
  num.true=round(m*prop.true);
  num.false=m-num.true;
  ## be careful when num.false or num.true is 0: replicate(0,...) produces empty list!
  if (num.false==0) group10=NULL else group10=replicate(num.false,rnorm(n,mean=0));
  if (num.true==0) group11=NULL else group11=replicate(num.true,rnorm(n,mean=d));
  group1=cbind(group10,group11);
  ## set vector of true effects
  d.true=c(rep(FALSE,num.false),rep(TRUE,num.true));
  ## apply t-test to all m pairs of samples
  pval=sapply(1:m,function(j) t.test(group0[,j],group1[,j])$p.value);
  ## get means and standard deviations of samples and differences between means
  mean0=colMeans(group0);
  mean1=colMeans(group1);
  sd0=sapply(1:m,function(j) sd(group0[,j]));
  sd1=sapply(1:m,function(j) sd(group1[,j]));
  diff=mean1-mean0;
  ## calculate theoretical and simulated (empirical) fdr
  fdr.theo=fdr_theo(pval,num.true,num.false,n,d);
  fdr.empi=fdr_empi(pval,d.true);
  ## store it all in data frame
  sim=data.frame(prop.true,m,n,d,d.true,pval,mean0,mean1,diff,sd0,sd1,
    fdr.theo=fdr.theo,fdr.empi=fdr.empi);
  invisible(sim);
}
```

Interpolation Functions

```
dointerp=function() {
  ## call interp_one on each case and combine results into data frame
  ## the complicated one-liner is a good idiom for doing this
  interp=do.call(rbind,apply(cases,1,
    function(case) do.call("interp_one",as.list(case)[c('prop.true','m','n','d')])));
  interp<<-interp;
  invisible(interp);
}
interp_one=function(prop.true,m,n,d) {
  sim1=sim[(sim$prop.true==prop.true&sim$m==m&sim$n==n &sim$d==d),];
  fun_theo=with(sim1,approxfun(pval,fdr.theo,rule=2));
  fdr.theo=fun_theo(pval.plot);
  fun_empi=with(sim1,approxfun(pval,fdr.empi,rule=2));
  fdr.empi=fun_empi(pval.plot);
  interp=data.frame(prop.true,m,n,d,pval=pval.plot,fdr.theo,fdr.empi);
  invisible(interp);
}
```

doplot

```
doplot=function(save.plot=F) {
    plot_byprop(save.plot);             # plot fdr by prop.true
    plot_byd(save.plot);                # plot fdr by d
    plot_vsprop(save.plot);             # plot fdr for fixed pvals vs prop.true
    plot_vsd(save.plot);                # plot fdr for fixed pvals vs d
  }
```

Plot Functions

```
plot_byprop=function(save.plot=F,d1=1,sig.level=.05) {
  if (!(d1 %in% cases$d)) d1=max(cases$d); # if desired value of d not in d, set to max
  sim=sim[sim$d==d1,];                     # select desired d
  sim.byprop=split(sim,sim$prop.true);     # split into groups
  ## initialize plot
  dev.new();
  title=paste(sep='','FDR vs. pval by prop.true for d=',round(d1,3));
  plot(x=NULL,y=NULL,type='n',xlim=c(0,.1),ylim=c(0,.5),xlab='pval',ylab='fdr',main=title);
  col=colramp(length(sim.byprop));
  ## plot fdr.theo and fdr.empi for each group
  sapply(1:length(sim.byprop),function(i) {
    sim1=sim.byprop[[i]];
    sim1=sim1[order(sim1$pval),];
    with(sim1,matlines(pval,cbind(fdr.theo,fdr.empi),col=col[i]))
  });
  ## add grid and dashed lines at sig.level
  abline(v=sig.level,col='grey',lty='dashed');
  abline(h=sig.level,col='grey',lty='dashed');
  grid();
```

```
  ## add legend
  legend.title='prop.true';
  legend.text=names(sim.byprop);
  legend.col=col;
  legend('topleft',bty='n',legend=legend.text,col=legend.col,pch=19,cex=1,
         title=legend.title,title.col='black')
  ## optionally save plot
  if (save.plot) savePlot(filename=file.path(figdir,'plot_byprop.png'));
}
plot_byd=function(save.plot=F,prop.true1=0.5,sig.level=.05) {
                                      # if desired value of d not in d, set to 1st value
  if (!(prop.true1 %in% prop.true)) prop.true1=prop.true[1];
  sim=sim[sim$prop.true==prop.true1,];  # select desired prop.true
  sim.byd=split(sim,sim$d);             # split into groups
  ## initialize plot
  dev.new();
  title=paste(sep='','FDR vs. pval by d for prop.true=',prop.true1);
  plot(x=NULL,y=NULL,type='n',xlim=c(0,.1),ylim=c(0,.5),xlab='pval',ylab='fdr',main=title);
  col=colramp(length(sim.byd));
  ## plot fdr.theo and fdr.empi for each group
  sapply(1:length(sim.byd),function(i) {
    sim1=sim.byd[[i]];
    sim1=sim1[order(sim1$pval),];
    with(sim1,matlines(pval,cbind(fdr.theo,fdr.empi),col=col[i]))
  });
  ## add grid and dashed lines at sig.level
  abline(v=sig.level,col='grey',lty='dashed');
  abline(h=sig.level,col='grey',lty='dashed');
  grid();
  ## add legend
  legend.title='d';
  legend.text=round(as.numeric(names(sim.byd)),3);
  legend.col=col;
  legend('topleft',bty='n',legend=legend.text,col=legend.col,pch=19,cex=1,
         title=legend.title,title.col='black')
  ## optionally save plot
  if (save.plot) savePlot(filename=file.path(figdir,'plot_byd.png'));
}
plot_vsprop=function(save.plot=F,d1=1,sig.level=.05) {
  if (!(d1 %in% cases$d)) d1=max(cases$d); # if desired value of d not in d, set to max
  interp=interp[interp$d==d1,];            # select desired d
  interp.bypval=split(interp,interp$pval); # split into groups
  ## initialize plot
  dev.new();
  title=paste(sep='','FDR vs. prop.true by pval for d=',round(d1,3));
  plot(x=NULL,y=NULL,type='n',xlim=range(prop.true),ylim=c(0,.5),
       xlab='prop.true',ylab='fdr',main=title);
  col=colramp(length(interp.bypval));
  ## plot fdr.theo and fdr.empi for each group
  sapply(1:length(interp.bypval),function(i) {
    interp1=interp.bypval[[i]];
    interp1=interp1[order(interp1$prop.true),];
    if (nrow(interp1)>1) with(interp1,matlines(prop.true,cbind(fdr.theo,fdr.empi),col=col[i]))
```

```
      else    # if interp1 has only 1 row, plot points insted of lines
        with(interp1,matpoints(prop.true,cbind(fdr.theo,fdr.empi),pch=c(19,21),col=col[i]));
  });
  ## add grid and dashed lines at sig.level
  abline(h=sig.level,col='grey',lty='dashed');
  abline(v=0.5,col='grey',lty='dashed');
  grid();
  ## add legend
  legend.title='pval';
  legend.text=names(interp.bypval);
  legend.col=col;
  legend('topright',bty='n',legend=legend.text,col=legend.col,pch=19,cex=1,
         title=legend.title,title.col='black')
  ## optionally save plot
  if (save.plot) savePlot(filename=file.path(figdir,'plot_vsprop.png'));
}
plot_vsd=function(save.plot=F,prop.true1=0.5,sig.level=.05) {
                                                  # if desired value of d not in d, set to 1st value
  if (!(prop.true1 %in% prop.true)) prop.true1=prop.true[1];
  interp=interp[interp$prop.true==prop.true1,]; # select desired prop.true
  interp.bypval=split(interp,interp$pval);      # split into groups
  ## initialize plot
  dev.new();
  title=paste(sep='','FDR vs. d by pval for prop.true=',prop.true1);
  plot(x=NULL,y=NULL,type='n',xlim=range(interp$d),ylim=c(0,.5),xlab='d',ylab='fdr',main=title);
  col=colramp(length(interp.bypval));
  ## plot fdr.theo and fdr.empi for each group
  sapply(1:length(interp.bypval),function(i) {
    interp1=interp.bypval[[i]];
    interp1=interp1[order(interp1$d),];
    if (nrow(interp1)>1) with(interp1,matlines(d,cbind(fdr.theo,fdr.empi),col=col[i]))
      else    # if interp1 has only 1 row, plot points insted of lines
        with(interp1,matpoints(d,cbind(fdr.theo,fdr.empi),pch=c(19,21),col=col[i]));
  });
  ## add grid and dashed lines at sig.level
  abline(h=sig.level,col='grey',lty='dashed');
  abline(v=1,col='grey',lty='dashed');
  grid();
  ## add legend
  legend.title='pval';
  legend.text=names(interp.bypval);
  legend.col=col;
  legend('topright',bty='n',legend=legend.text,col=legend.col,pch=19,cex=1,
         title=legend.title,title.col='black')
  ## optionally save plot
  if (save.plot) savePlot(filename=file.path(figdir,'plot_vsd.png'));
}
```

Save and Load

```
saveit=function(save.rdata=T,save.txt=F) {
  if (save.rdata) {
```

```
    ## get names of global variables that aren't functions
    vars=Filter(function(x) !is.function(get(x,envir=.GlobalEnv)),ls(envir=.GlobalEnv));
    save(list=vars,envir=.GlobalEnv,file=file.path(datadir,'globals.RData'));
  }
  if (save.txt) {
    write.table(sim,file=file.path(datadir,'sim.txt'),sep='\t',quote=F,row.names=F);
    write.table(interp,file=file.path(datadir,'interp.txt'),sep='\t',quote=F,row.names=F);
  }
}
loadit=function(file='data/swfdr_base/globals.RData') {
  if (!file.exists(file))
    stop(paste(sep=' ',"Cannot load saved parameters and results: file",file,"does not exist"));
  load(file=file,envir=.GlobalEnv);
}
```

FDR Calculations

```
fdr_theo=function(pval,num.true,num.false,n=16,d=1) {
  pwr=sapply(pval,function(p) power.t.test(n,delta=d,sd=1,sig.level=p)$power);
  num.tp=pwr*num.true;
  num.fp=pval*num.false;
  num.fp/(num.tp+num.fp);
}
fdr_empi=function(pval,d.true) {
  order=order(pval);
  pval=pval[order];
  d.true=d.true[order];
  m=length(pval);
  neg=cumsum(ifelse(d.true,0,1));
  fdr=neg/(1:m);        # because pval is sorted, index is number of entries with smaller pval
                        # equals number of entries that would be accepted at this pval
  ## deal with ties if necessary
  if (anyDuplicated(pval)) {
    ## split fdr by pval, then set fdr to max fdr of group.
    ## CAUTION: have to deal with approximate equality of pvals
    digits=ceiling(-log10(min(pval)));  # number of significant digits in smallest pval
    pval.approx=signif(pval,digits);
    fdr.by.pval=split(fdr,pval.approx);
    max.by.pval=sapply(fdr.by.pval,function(g) if (all(is.na(g))) NA else max(g,na.rm=T));
    fdr=sapply(pval.approx,function(p) max.by.pval[as.character(p)]);
  }
  unorder=order(order); # restore original order
  fdr[unorder];
}
```

Utility Functions

```
power_grid=function(cases) {
  cases=do.call(rbind,apply(cases,1,function(case) {
    case=as.list(case);
```

```
    case=with(case,{
      if (is.na(pwr)) pwr=NULL else d=NULL;
      power=power.t.test(n=n,delta=d,sd=1,power=pwr,sig.level=sig.level);
      case[c('n','d','pwr')]=power[c('n','delta','power')];
      data.frame(case);
    })}));
  cases;
}
colramp=colorRampPalette(c('red','green','blue'))
```