**SAS/IML Program Code (using SAS version 9.1)**

The following code defines all of the modules to implement the numeric solutions

defined in the paper. No output is produced by running this code.

```
****invoke the IML program and define the modules WJGLM and BOOTCOM****;
proc iml;
reset noname;

**define module to check initial specifications****;
start initial(c,u,y,opt1,per,opt2,numsim_b,numsim_bc,opt3,numsim_es,loc1,loc2,scale,
alpha,seed,r,x) global(nx,ntot,bobs,wobs,wobs1);
if nrow(u)=0 then u=i(ncol(y));
  if ncol(u)>nrow(u) then print
    'Possible Error: Number Of Columns Of U Exceeds Number Of Rows';
if opt1=1 then if nrow(per)=0 then per=.20;
  if opt1=1 then if per > .49 then print
       'Error: Percentage Of Trimming Exceeds Upper Limit';
if opt2=1 then if nrow(numsim_b)=0 then numsim_b=999;
if opt3=1 then if nrow(numsim_es)=0 then numsim_es=999;
if nrow(alpha)=0 then alpha=.05;
if opt3=1 then if nrow(loc1)=0 then loc1=1;
if opt3=1 then if nrow(loc2)=0 then loc2=1;
if nrow(scale)=0 then scale=1;
if nrow(seed)=0 then seed=0;
if nrow(numsim_bc)=0 then numsim_bc=699;

do i=1 to ncol(nx);
  x1=j(nx[i],1,i);
  if i=1 then x=x1;
  else x=x//x1;
end;

x=design(x);
ntot=nrow(y);
wobs=ncol(y);
bobs=ncol(x);
wobs1=wobs-1;
r=c@u`;
finish;
****define module to compute least squares or trimmed means****;
start mnmod (y,x,opt1,per,bhat,bhatw,muhat,yt,df) global(bobs,wobs,wobs1,ntot,nx);
if opt1=0 then do;
  bhat=inv(x`*x)*x`*y;
  bhatw=bhat;
```

```
  yt=y;
  df=nx-1;
end;
if opt1=1 then do;
  bhat=j(bobs,wobs,0);
  bhatw=bhat;
  yt=j(ntot,wobs,0);
  df=j(1,bobs,0);
  f=1;
  m=0;
  do j=1 to ncol(nx);
    samp=nx[j];
    l=m+samp;
    g=int(per#samp);
    df[j]=samp-2#g-1;
    do k=1 to ncol(y);
    temp=y[f:l,k];
    nv=temp;
    temp[rank(nv),]=nv;
    trimy=temp[g+1:samp-g,];
    trimmn=sum(trimy)/(df[j]+1);
    bhat[j,k]=trimmn;
    mint=min(trimy);
    maxt=max(trimy);
    do p=1 to nrow(nv);
          if nv[p]<=mint then nv[p]=mint;
          if nv[p]>=maxt then nv[p]=maxt;
        end;
    yt[f:l,k]=nv;
    winmn=sum(nv)/samp;
    bhatw[j,k]=winmn;
   end;
   m=l;
   f=f+nx[j];
  end;
end;
muhat=shape(bhat,bobs#wobs);
finish;
***define module to compute sigma matrix****;
start sigmod (yt,x,bhatw,df,sigma,stdizer) global(bobs,wobs,wobs1,ntot,nx);
sigma=j(wobs#bobs,wobs#bobs,0);
stdizer=sigma;
do i=1 to bobs;
  sigb=(yt#x[,i]-x[,i]*bhatw[i,])`*(yt#x[,i]-x[,i]*bhatw[i,])/((df[i]+1)#df[i]);
  f=i#wobs-wobs1;
  l=i#wobs;
```

```
  sigma[f:l,f:l]=sigb;
  stdizer[f:l,f:l]=sigb#((df[i]+1)#df[i])/(nx[i]-1);
end;
finish;
****define module to compute test statistic****;
start testmod(sigma,muhat,r,df,fstat,df1,df2) global(bobs,wobs,wobs1,ntot,nx,x);
t=(r*muhat)`*inv(r*sigma*r`)*(r*muhat);
a=0;
imat=i(wobs);
do i=1 to bobs;
  qmat=j(bobs#wobs,bobs#wobs,0);
  f=i#wobs-wobs1;
  l=i#wobs;
  qmat[f:l,f:l]=imat;
  prod=(sigma*r`)*inv(r*sigma*r`)*r*qmat;
  a=a+(trace(prod*prod)+trace(prod)**2)/df[i];
end;
a=a/2;
df1=nrow(r);
df2=df1#(df1+2)/(3#a);
cval=df1+2#a-6#a/(df1+2);
fstat=t/cval;
finish;
****define modules to perform bootstrap****;
***define module to generate bootstrap data****;
start bootdat (y,yb1,seed)  global(bobs,wobs,wobs1,ntot,nx);
  f=1;
  m=0;
  do j=1 to bobs;
   l=m+nx[j];
   temp=y[f:l,];
   bval=temp;
   do p=1 to nrow(temp);
     rval=uniform(seed);
     bval[p,]=temp[ceil(nrow(temp)#rval),];
   end;
   if j=1 then yb1=bval;
   else yb1=yb1//bval;
   m=l;
   f=f+nx[j];
 end;
 finish;
****define module to centre the bootstrap data****;
start bootcen (yb1,bhat,yb) global(bobs,wobs,wobs1,ntot,nx,r);
m=0;
f=1;
```

```
yb=yb1;
do i=1 to bobs;
  l=m+nx[i];
  mval=bhat[i,];
  do q=f to l by 1;
    yb[q,]=yb1[q,]-mval;
  end;
  m=l;
  f=f+nx[i];
end;
finish;
****define module to compute bootstrap test statistic****;
start bootstat(yb,x,opt1,per,r,fstatb) global(bobs,wobs,wobs1,ntot,nx);
  call mnmod(yb,x,opt1,per,bhatb,bhatbw,muhatb,ytb,dfb);
  call sigmod(ytb,x,bhatbw,dfb,sigmab,stdizerb);
  call testmod(sigmab,muhatb,r,dfb,fstatb,df1b,df2b);
finish;
****define module to compute bootstrap effect size****;
start bootes(yb,x,loc1,loc2,scale,opt1,per,r,multp,effszb)
global(bobs,wobs,wobs1,ntot,nx);
  call mnmod(yb,x,opt1,per,bhatb,bhatbw,muhatb,ytb,dfb);
  call sigmod(ytb,x,bhatbw,dfb,sigmab,stdizerb);
  call wjeffsz(loc1,loc2,scale,opt1,per,r,muhatb,stdizerb,multp,effszb);
finish;
****compute measure of effect size and bootstrap confidence interval****;
start wjeffsz(loc1,loc2,scale,opt1,per,r,muhat,stdizer,multp,effsz)
global(bobs,wobs,wobs1,ntot,nx);
  if opt1=0 then multp=1;
  if opt1=1 then if scale=0 then multp=1;
  if opt1=1 then if scale=1 then do;
    cut=probit(per);
         **fun defines the probability density function of the standard normal distribution**;
    start fun(z);
    v=(z**2)*(1/(sqrt(2*3.141592653589793))*(exp(-(1/2)*z**2)));
         return(v);
    finish;
    a=j(1,3,0);
    a[1,1]=cut;
    a[1,3]=-1*cut;
    call quad(int,"fun",a);
    b=sum(int);
    winvar=b+per*(2*(cut**2));
    multp=sqrt(winvar);
  end;
num=r*muhat;
if loc1=99 then stdz=1;
```

```
if loc1=0 then do;
  r2=r##2;
  rvec=shape(r2,bobs#wobs)`;
  stdz1=sqrt(diag(stdizer));
  stdz=(rvec*stdz1*rvec`)/sum(rvec);
end;
if  loc1 > 0 then if loc1 < 99 then do;
  loc=loc1*wobs-(wobs-loc2);
  stdz1 = stdizer[loc,loc];
  if stdz1> 0 then stdz=sqrt(stdz1);
  if stdz1=0 then stdz=.00001;
end;
effsz=multp#(num/stdz);
finish;
****compute Welch-James statistic****;
start wjglm;
call initial(c,u,y,opt1,per,opt2,numsim_b,numbsim_bc,opt3,numsim_es,loc1,loc2,scale,
alpha,seed,r,x);
call mnmod(y,x,opt1,per,bhat,bhatw,muhat,yt,df);
call sigmod(yt,x,bhatw,df,sigma,stdizer);
call testmod(sigma,muhat,r,df,fstat,df1,df2);
if opt3=1 then call wjeffsz (loc1,loc2,scale,opt1,per,r,muhat,stdizer,multp,effsz);
if opt2=1 then do;
   do simloop=1 to numsim_b;
      call bootdat(y,yb1,seed);
            call bootcen(yb1,bhat,yb);
      call bootstat(yb,x,opt1,per,r,fstatb);
         if simloop=1 then fmat=fstatb;
        else fmat=fmat//fstatb;
   end;
   tempb=fmat;
fmat[rank(fmat)]=tempb;
end;
if opt3=1 then do;
  do simloop=1 to numsim_es;
    call bootdat(y,yb1,seed);
          call bootes(yb1,x,loc1,loc2,scale,opt1,per,r,multp,effszb);
            if simloop=1 then esmat=effszb;
            else esmat=esmat//effszb;
    end;
          tempc=esmat;
  esmat[rank(esmat)]=tempc;
end;
***calculate significance level for welch-james statistic****;
results=j(4,1,0);
results[1]=fstat;
```

```
results[2]=df1;
results[3]=df2;
if opt2=0 then results[4]=1-probf(results[1],df1,df2);
if opt2=1 then do;
  avec=(fstat<=fmat);
  pval=sum(avec)/numsim_b;
  results[4]=pval;
end;
if opt3=1 then do;
ind1=int(numsim_es#(alpha/2))+1;
ind2=numsim_es-int((numsim_es#(alpha/2)));
lcl=esmat[ind1];
ucl=esmat[ind2];
end;
****print results****;
print 'Welch-James Approximate DF Solution';
if opt1=0 then print 'Least Squares Means & Variances';
if opt1=1 then print 'Trimmed Means & Winsorized Variances';
if opt1=1 then print 'Percentage of Trimming:'per[format=4.2];
if opt2=0 then if opt3=0 then print 'F Distribution Critical Value';
if opt2=1 then if opt3=0 then print 'Bootstrap Critical Value for Single Test Statistic';
if opt2=1 then if opt3=0 then do;
  print 'Number of Bootstrap Samples:' numsim_b[format=5.0],;
  print 'Starting Seed:' seed[format=15.0],;
end;
print'Contrast Matrix:';
print r[format=4.1],;
muhat=muhat`;
print 'Mean Vector:';
print muhat[format=10.4],;
print 'Sigma Matrix:';
print sigma[format=10.4],;
reslab={"Test Statistic" "Numerator DF" "Denominator DF" "p-value"};
if opt3=0 then do;
  print 'Significance Test Results:';
  print results[rowname=reslab format=10.4]/;
end;
if opt3=1 then do;
   cilev = (1 - alpha)*100;
   print 'Effect Size:' effsz[format=10.4];
   if scale=0 then print 'No Scaling Factor';
   if scale=1 then print 'Scaling Factor is:' multp[format=6.3];
   if loc1=0 then print 'Standardizer Is Square Root of Average Variance';
   if loc1 > 0 then if loc1 < 99 then do;
     print 'Standardizer is:';
     print 'LOC1:' loc1[format=3.0];
```

```
     print 'LOC2:' loc2[format=3.0];
   end;
   if loc1=99 then print 'No Standardizer';
   print 'Number of Bootstrap Samples:' numsim_es[format=5.0];
   print 'Starting Seed:' seed[format=15.0];
   print 'Confidence Level (%):' cilev[format=5.1];
   print 'Lower Confidence Limit:' lcl[format=10.4];
   print 'Upper Confidence Limit:' ucl[format=10.4]/;
end;
finish;
****define module to compute bootstrap critical value for FWR control of ADF
statistics****;
start bootcom;
call initial(c,u,y,opt1,per,opt2,numsim_b,numsim_bc,opt3,numsim_es,loc1,loc2,scale,
alpha,seed,r,x);
call mnmod(y,x,opt1,per,bhat,bhatw,muhat,yt,df);
call sigmod(yt,x,bhatw,df,sigma,stdizer);
do i=1 to nrow(r);
cm=r[i,];
call testmod(sigma,muhat,cm,df,fstat,df1,df2);
if i=1 then do;
   cmmat=fstat;
   df1mat=df1;
   df2mat=df2;
end;
if i>1 then do;
    cmmat=cmmat||fstat;
    df1mat=df1mat||df1;
    df2mat=df2mat||df2;
end;
end;
do simloop=1 to numsim_bc;
call bootdat(y,yb1,seed);
call bootcen(yb1,muhat,yb);
do q=1 to nrow(r);
  cm=r[q,];
  call bootstat(yb,x,opt1,per,cm,fstatb);
  if q=1 then frow=fstatb;
    else frow=frow||fstatb;
end;
if simloop=1 then fmat=frow;
  else fmat=fmat//frow;
end;
fmax=j(numsim_bc,1,0);
do k=1 to numsim_bc;
 fmax[k]=max(fmat[k,]);
```

```
end;
tempb=fmax;
if ncol(alpha) = 0 then alpha = .05;
fmax[rank(fmax)]=tempb;

results=j(3,nrow(r),0);
results[1,]=cmmat;
results[2,]=df1mat;
results[3,]=df2mat;
qcrit=round((1-alpha)#numsim_bc);
critv=fmax[qcrit];
****print results****;
print 'Welch-James Approximate DF Solution';
if opt1=0 then print 'Least Squares Means & Variances';
if opt1=1 then print 'Trimmed Means & Winsorized Variances';
if opt1=1 then do;
  print 'Percentage of Trimming:';
  print per[format=4.2];
end;
if opt2=1 then print 'Bootstrap Critical Value for FWR Control';
if opt2=1 then print 'Number of Bootstrap Samples for Test Statistic Critical Value:';
if opt2=1 then print numsim_bc[format=4.0];
if opt2=1 then do;
  print 'Starting Seed:';
  print seed[format=15.0],;
end;
print 'Contrast Matrix:';
print r[format=4.1],;
muhat=muhat`;
print 'Mean Vector:';
print muhat[format=10.4],;
print 'Sigma Matrix:';
print sigma[format=10.4],;
reslab={"Test Statistic" "Numerator DF" "Denominator DF" "Significance"};
print 'Significance Test Results:';
print results[rowname=reslab format=10.4],;
if opt2=1 then print 'Critical Value:';
if opt2=1 then print critv[format=5.2]/;
finish;
```