
S4: Fitting power-laws in empirical data with estimators that work for all exponents

Rudolf Hanel¹, Bernat Corominas-Murtra¹, Bo Liu¹, Stefan Thurner^{1,2,3,4},

1 Section for Science of Complex Systems, Medical University of Vienna, Spitalgasse 23, 1090 Vienna, Austria

2 Santa Fe Institute, 1399 Hyde Park Road, Santa Fe, NM 87501, USA

3 IIASA, Schlossplatz 1, 2361 Laxenburg, Austria

4 Complexity Science Hub Vienna, Josefstadtstrasse 39, A-1090 Vienna, Austria

✉ Current Address: Section for Science of Complex Systems, CeMSIIS, Medical University of Vienna, Spitalgasse 23, Bauteil 86, A-1090, Vienna, Austria

* stefan.thurner@meduniwien.ac.at

APPENDIX D: Code

For printing the code all unnecessary comments have been removed.

APPENDIX D1: r_plfit

```
1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 % Author: R Hanel: 1.6.2016; last modification on 10.08.2016
3 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
4 function out = r_plfit(z, varargin)
5 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
6 % Main begin
7 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
8 if length(z)==length('help'), if strcmp(z, 'help') ==1,
9     msg=['Using function out = r_plfit(data, varargin)\n'];
10    msg=[msg 'r_plfit can be used to fit the exponent out.exponent of power
laws\n'];
11    msg=[msg 'in three basic different modes\n'];
12    msg=[msg '(1) out = r_plfit(x) with data being samples x=[x1,...,xN]
from N experiments\n'];
13    msg=[msg '\t in this mode out.exponent returns the estimated exponent
lambda of \n'];
14    msg=[msg '\t the distribution p(z_i|lambda) one has sampled from\n'];
15    msg=[msg '(2) out = r_plfit(k, 'hist') with data being histograms
k=[x1,...,xW]\n'];
16    msg=[msg '\t of W distinct events i=1...W with magnitudes
z=[z1,...,zW]\n'];
17    msg=[msg '\t here out.exponent also returns lambda\n'];
18    msg=[msg '(3) out = r_plfit(k) with data being histograms
k=[x1,...,xW]\n'];
19    msg=[msg '\t works equivalent to (1) but with k as data one computes
the\n'];
20    msg=[msg '\t frequency distribution of x with exponent
alpha^-1+1/lambda\n'];
21    msg=[msg '\n'];
22    msg=[msg 'Fitting with data x:\n'];
23    msg=[msg '\t By default r_plfit(x) assumes that x consists of natural
numbers and\n'];
24    msg=[msg '\t the sample space (magnitudes) z={i | min(x)<=i<=max(x)}\n'];
25    msg=[msg '\t if magnitudes z_i are not of this form one either uses\n'];
```

```

26     msg=[msg '\t if magnitudes z_i are not of this form one either
27         uses:\n'];
28     msg=[msg '\t out = r_plfit(x, ''urange'') ... which sets
29         z=[z_1,...,z_W]=unique(x)\n'];
30     msg=[msg '\t where W is then the number of unique values of data points
31         in x\n'];
32     msg=[msg '\t out =
33         r_plfit(x, ''urange'', ''rangemin'',zmin, ''rangemax'',zmax) can be
34         used\n'];
35     msg=[msg '\t to specify a fit range zmax <= z_i <= zmin ( default
36         zmin=min(x), zmax=max(x))\n'];
37     msg=[msg '\t To control the data range individually use out =
38         r_plfit(x, ''range'',z)\n'];
39     msg=[msg '\n'];
40     msg=[msg 'Fitting with histograms k\n'];
41     msg=[msg '\t Using k (mode 3) for fitting the frequenvcy distribution
42         with exponent alpha\n'];
43     msg=[msg '\t works in exactly the same way as the sample distribution
44         estimating the\n'];
45     msg=[msg '\t exponent lambda using x.\n'];
46     msg=[msg '\t out = r_plfit(k, ''hist'') works similar however one should
47         note that\n'];
48     msg=[msg '\t in this mode r_plfit assumes by default that
49         z=[1,2,...,W]\n'];
50     msg=[msg '\t The option ''urange'' has no effect in this mode and gets
51         ignored if set\n'];
52     msg=[msg '\t Otherwise use out = r_plfit(k, ''hist'', ''range'',z) to set
53         the event magnitudes z\n'];
54     msg=[msg '\t ''rangemin'' and ''rangemax'' options work in exactly the
55         same way as before\n'];
56     msg=[msg '\n'];
57     msg=[msg 'Automatic low frequency cutoff \n'];
58     msg=[msg '\t By default r_plfit runs an iterative search for an optimal
59         low frequency cutoff\n'];
60     msg=[msg '\t If cutoff is set at index i where |Nz_i^lambda/N-Nmin| is
61         minimal where Nmin=1\n'];
62     msg=[msg '\t by default and N the length of x=[x1,...,xN]; Nmin can be
63         set using the\n'];
64     msg=[msg '\t ''Nmin'' option. If maxrange is smaller than the predicted
65         if cutoff then the\n'];
66     msg=[msg '\t cutoff has no effect. Note that in the out = r_plfit(k)
67         mode the lf cutoff mechanism\n'];
68     msg=[msg '\t effectively acts as ahigh frequency cutoff with respect to
69         the data x\n'];
70     msg=[msg '\t the option ''nolf'' switches of the lf cutoff \n'];
71     msg=[msg '\n'];
72     msg=[msg 'out = r_plfit(data,..., ''plot'') displays the fit over the
73         data\n'];
74     msg=[msg 'in double logarithmic coordinates (loglog plot)\n'];
75     msg=[msg ''fig'' behaves like ''plot'' but explicitly opens a new
76         figure\n'];
77     msg=[msg ''exp_min'' can be used to specify the minimal search value
78         exp_min (default =0)\n'];
79     msg=[msg 'for out.exponent. Similarly
80         r_plfit(data,..., ''exp_max'',expmax) sets the minimal\n'];
81     msg=[msg ' search value (default =5) to expmax\n'];
82     msg=[msg ''eps'', ... set absolute error eps (default eps=1e-10) for
83         out.exponent \n'];
84     msg=[msg '\n'];
85     msg=[msg 'Other options to control the performance of the
86         algorithm:\n'];
87     msg=[msg ''Ncut'', ... specifies the number of values alpha in the
88         search range\n'];
89     msg=[msg 'of out.exponent. After m iterations the precision of
90         out.exponent becomes\n'];

```

```

63     msg=[msg ' (expmax-expmin)/(Ncut/2)^m =>
64         m(eps)^2*(log(expmax-expmin)-log(eps))/Ncut\n'];
65     msg=[msg '' Nimplicit '' sets the maximal number of implicit iteration
66         for finding \n the exponent (default 80) of iterations\n];
67     msg=[msg 'searching implicitly for out.exponent. one should use
68         Nimplicit>m(eps)\n'];
69     msg=[msg '' Ntail '' sets the maximal number of iterations for the lf
70         cutoff\n];
71     msg=[msg '' info '' if set will output info over the run stored in
72         out.info at runtime!\n];
73     msg=[msg 'Bug reports to: rudolf.hanel@meduniwien.ac.at\n'];
74     fprintf(1,msg); out=msg;
75     return;
76 end; end;
77 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
78 % Initialization & Default values
79 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
80 N=length(z); % number of data items
81 xflg=0; cleanxflg=0; histflg=0; cdatflg=0; rangemin=0; rangemax=Inf;
82 Nrangeitems_max=1e6; eps=1e-10; issmall=1e-50; alpha_min=0; alpha_max=5;
83 lfcutoff=1; plotflg=0; newfigflg=0; testflg=1; Nmin=1; Ninc=50; Ntail=80;
84 Nimplicit=80; Nrep_cdat=80; rep_min=3; Amlthres=0.1; infoflg=0;
85 runinfo='start ... \n';
86 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
87 % varargin
88 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
89 id=0;
90 while id<length(varargin),
91     id=id+1;
92     if ischar(varargin{id}),
93         switch varargin{id},
94             case 'exp_min', alpha_min = varargin{id+1}; id=id+1;
95             case 'exp_max', alpha_max = varargin{id+1}; id=id+1;
96             case 'Ntail', Ntail = varargin{id+1}; id=id+1;
97             case 'Nimplicit', Nimplicit = varargin{id+1}; id=id+1;
98             case 'Nmin', Nmin = varargin{id+1}; id=id+1;
99             case 'Ncut', Ninc = varargin{id+1}; id=id+1;
100            case 'range', xrange = varargin{id+1}; xflg=1; id=id+1;
101            case 'rangemin', rangemin = varargin{id+1}; id=id+1;
102            case 'rangemax', rangemax = varargin{id+1}; id=id+1;
103            case 'urange', xflg=2;
104            case 'cdat', cdatflg=1; lfcutoff=0; xflg=2;
105            case 'cdat2', cdatflg=1; testflg=2; lfcutoff=0; xflg=2;
106            case 'hist', histflg=1;
107            case 'nolf', lfcutoff=0;
108            case 'info', infoflg=1;
109            case 'plot', plotflg=1;
110            case 'fig', plotflg=1; newfigflg=1;
111            case 'cleanrange', cleanxflg=1;
112            case 'eps', eps = varargin{id+1}; id=id+1;
113            otherwise, msg=['no such argument [ ' varargin{id} ' ] ->skip
114                ... \n'];
115            runinfo=[runinfo msg]; if infoflg==1, fprintf(1,msg); end;
116        end
117    end
118 end
119 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
120 % handle hist property, data filtering etc ...
121 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
122 if histflg==1, %data is histogram data

```

```

123     msg=['handling histogram input case ' num2str(xflg), ' ...\\n'];
124     runinfo=[runinfo msg]; if infoflg==1, fprintf(1,msg); end;
125     kz; zz=z;
126 %... urange property is specified
127     if xflg==2,
128         msg=['r_plfit: urange property has no effect
129             together with hist input -> ', ...
130             'set default range case 0 ... \\n'];
131         runinfo=[runinfo msg]; if infoflg==1,
132             fprintf(1,msg); end; xflg=0;
133     end;
134     switch xflg,      % if input is histogram set range to...
135         case 0, %... default range if range is not specified
136             x=1:length(k); rangemin=max(rangemin,1);
137                 rangemax=min(rangemax,length(k));
138         case 1, %... range property is specified
139             if length(k)==length(xrange); x=xrange;
140                 rangemin=max(rangemin,min(x));
141                 rangemax=min(rangemax,max(x));
142             else
143                 fprintf(1,runinfo);
144                 error('r_plfit: range and histogram must have same
145                     length!!!');
146             end;
147         otherwise
148             fprintf(1,runinfo);
149             error('this should not happen 1 !!!');
150         end;
151     xidv=find(x>=rangemin & x<=rangemax);
152     W=length(x); N=sum(k); xx=x(xidv);
153     kk=k(xidv); WW=length(xx); NN=sum(kk);
154 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
155 % data=samples; clean data
156 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
157     msg=['cleaning data input ... \\n'];
158     runinfo=[runinfo msg];
159     if infoflg==1, fprintf(1,msg); end;
160     idv=find(z>0); zz=z(idv); idv=find(abs(log(zz))<Inf);
161     zz=zz(idv); idv=find(isnan(zz)==0); zz=zz(idv);
162     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
163     rangemin=max(rangemin,min(zz));
164     rangemax=min(rangemax,max(zz));
165     if rangemin>rangemax,
166         fprintf(1,runinfo);
167         error('r_plfit requires rangemin<rangemax!');
168     end;
169     zidv=find(zz>=rangemin & zz<=rangemax); zz=zz(zidv);
170     if ((max(zz)-min(zz))>Nrangeitems_max) & (xflg==0),
171         msg1=['Length of r_plfit default range rangemin:1:rangemax , ...
172             num2str((max(zz)-min(zz))) ' is
173             too large!\\n'];
174         msg2=['Maximum allowed: ' num2str(Nrangeitems_max) ...
175             '; => r_plfit tries the urange
176             option!\\n'];
177         msg3=['Otherwise, try again by setting the range property
178             manually with options: ...
179             ' range, maxrange, minrange!\\n'];
180         runinfo=[runinfo msg1 msg2 msg3];
181         if infoflg==1, fprintf(1,[msg1 msg2 msg3]); end; xflg=2;
182     end;
183     msg=['handling data input case ' num2str(xflg), ' ...\\n'];
184     runinfo=[runinfo msg];
185     if infoflg==1, fprintf(1,msg); end;
186     switch xflg ,

```

```

181                                     %... default range if range is not
182                                     specified
183             case 0,
184                 x=min(zz):max(zz);
185                 rangemin=max(rangemin,min(x));
186                 rangemax=min(rangemax,max(x));
187                                     %... range property is specified
188             case 1,
189                 x=xrange;
190                 rangemin=max(rangemin,min(x));
191                 rangemax=min(rangemax,max(x));
192                 % filter
193                 uz=unique(zz); uzon=zeros(size(uz));
194                 for uid=1:length(uz),
195                     if length(v)>0, uzon(uid)=1; end;
196                 end;
197                 uzoffidv=find(uzon==0);
198                 for uid=1:length(uzoffidv),
199                     v=find(uz(uzoffidv(uid))==zz);
200                     if length(v)>0, zz(v)=-1; end;
201                 end;
202                 v=find(zz>0); zz=zz(v);
203                                     %... urange property is specified
204             case 2,
205                 x=unique(zz);
206                 rangemin=max(rangemin,min(x));
207                 rangemax=min(rangemax,max(x));
208             otherwise
209                 fprintf(1,runinfo);
210                 error('this should not happen 1 !!!');
211             end;
212 %%%%%%
213 % sort data & filter sample-space
214 %%%%%%
215         zz=sort(zz);
216         if cleanxflg==1,
217             msg=['cleaning sample space ...\\n'];
218             runinfo=[runinfo msg];
219             if infoflg==1, fprintf(1,msg); end;
220             uz=unique(zz); xon=zeros(size(x));
221             for xid=1:length(uz),
222                 v=find(uz==x(xid));
223                 if length(v)>0, xon(xid)=1; end;
224             end;
225             x=x(find(xon==1));
226         end;
227 %%%%%%
228 % compute histogram
229 %%%%%%
230         msg=['computing histogram ...\\n'];
231         runinfo=[runinfo msg]; if infoflg==1, fprintf(1,msg); end;
232         x=sort(x); W=length(x); maxx=max(x);
233         while 1,
234             xidv=find(x>=rangemin & x<=rangemax);
235             if length(xidv)==0,
236                 rangemin=max(rangemin-1,issmall);
237                 rangemax=min(rangemax+5,maxx);
238             else break; end;
239         end;
240         k=zeros(size(x)); [k,xdummy]=hist(zz,x); N=sum(k);
241         xx=x(xidv); kk=k(xidv); WW=length(xx); NN=sum(kk);
242     end;
243     [a,b]=size(kk); if b==1, kk=kk'; end;
244     [a,b]=size(xx); if b==1, xx=xx'; end;
245

```

```

246 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
247 % Maximum likelihood estimate: find ML optimal alpha
248 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
249
250 xrangemax=max(xx); xrangemax2=xrangemax;
251 idrmax=WW; idrmax_1=0; idrmax_2=0; QQ=kk.* log(xx);
252 xmin=rangemin; xmax=rangemax; alphbest=0;
253
254 param.Nimplicit=Nimplicit; param.Ninc=Ninc;
255 param.alpha_max=alpha_max; param.alpha_min=alpha_min;
256 param.cdatflg=cdatflg; param.testflg=testflg;
257 param.infoflg=infoflg; param.issmall=issmall;
258
259 if lfcutoff==1,
260 %% optimize low frequency cut-off for ML optimal alpha
261 idrmax_1=WW;
262 msg=['find exponent (lf cut-off is on) ... \n'];
263 runinfo=[runinfo msg];
264 if infoflg==1, fprintf(1,msg); end;
265 for rep1=1:Ntail,
266     xridv=find(xx<=xrangemax2); idrmax_2=idrmax_1;
267     idrmax_1=idrmax; idrmax=max(xridv); xxx=xx(xridv);
268     kkk=kk(xridv); WWW=length(xridv); NNN=sum(kkk); QQQ=QQ(xridv);
269 %% optimize alpha for Ninc alpha values
270 loopmsg=['if-loop: ', num2str(rep1), ' '];
271 [alphbest, runinfo]=h_implicit(NNN,xxx,QQQ,runinfo,loopmsg,alphbest,param);
272 Aml=alphbest; yml=xx.^(-Aml); yml=yml/sum(yml);
273 gugu=abs(NN*yml-Nmin); v=find(gugu==min(gugu));
274 if Aml>Amlthres, idrmax=max(v); else idrmax=WW; end;
275 if (idrmax==idrmax_1 | idrmax==idrmax_2) & rep1>=rep_min, break;
276 end;
277 idrmax=max(idrmax,1); xrangemax2=xx(idrmax);
278 end;
279 else
280     msg=['find exponent (lf cut-off is off) ... \n'];
281     runinfo=[runinfo msg];
282     if infoflg==1, fprintf(1,msg); end;
283     [alphbest, runinfo]=h_implicit(NN,xx,QQ,runinfo,'',alphbest,param);
284     Aml=alphbest; idrmax=WW;
285 end
286 msg=['preparing output ... \n'];
287 runinfo=[runinfo msg];
288 if infoflg==1, fprintf(1,msg); end;
289
290 xlf cutoff=1:idrmax;
291
292 % Quality of fit;
293 xxx=xx(xlf cutoff);
294 plf=xxx.^(-Aml); %Aml,
295 plf=plf/sum(plf);
296 p=xx.^(-Aml); %Aml,
297 p=p/sum(p);
298
299 ylf=kk(xlf cutoff)/sum(kk(xlf cutoff)); y=kk/sum(kk);
300 if plotflg==1,
301     if newfigflg==1, figure; end;
302     loglog(xxx,ylf,'r')
303     hold on;
304     loglog(xxx,plf),
305     loglog(xxx(idrmax),plf(idrmax),'go'),
306     pause(0.01);
307 end;
308
309
310

```

```

311 out.runinfo=runinfo;
312 out.N = NN; out.K=kk; out.fit.data=zz;
313 out.fit.Nsamples=N; out.fit.Nmin = Nmin;
314 out.fit.range=xx; out.fit.lf_cutoff_on=lf cutoff;
315 out.fit.alpha_min=alpha_min; out.fit.alpha_max=alpha_max;
316 out.fit.Ntail=Ntail; out.fit.Nimplicit=Nimplicit;
317 out.fit.Ninc=Ninc;
318
319 out.exponent = Aml;
320 out.KSall = max(abs(cumsum(y)-cumsum(p)));
321 out.KStrange = max(abs(cumsum(ylf)-cumsum(plf)));
322 out.xmax = max(xx); out.xmin = min(xx);
323 out.rangemax = rangemax; out.rangemin = rangemin;
324 out.lf_rangemax = max(xxx); out.lf_cutoffid = idrmax;
325
326 return;
327 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
328 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
329 % MAIN END
330 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
331 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
332
333 function
[alphbest,runinfo]=h_implicit(NN,xx,QQ,runinfo,loopmsg,alphbest,param)
334
335 Nimplicit=param.Nimplicit; Ninc=param.Ninc;
336 alpha_max=param.alpha_max; alpha_min=param.alpha_min;
337 cdatflg=param.cdatflg; testflg=param.testflg;
338 infoflg=param.infoflg; issmall=param.issmall;
339 xmin=min(xx); xmax=max(xx);
340
341 dalph=(alpha_max-alpha_min)/Ninc;
342 alphav=alpha_min:dalph:alpha_max;
343 QQQ=sum(QQ);
344 for rep2=1:Nimplicit,
345 msg1=[ 'implicit: ', num2str(rep2)];
346 msg2=[ 'accuracy: ', num2str(log(dalph/eps)/log(10))];
347 msg3=[ ' alpha: ', num2str(alphbest), ' ... \n'];
348 runinfo=[runinfo loopmsg msg1 msg2 msg3];
349 if infoflg==1, fprintf(1,[loopmsg msg1 msg2 msg3]); end;
350 d=zeros(size(alphav));
351 count=0;
352 logxx=log(xx);
353 for alph=alphav,
354 count=count+1; px=xx.^(-alph); px=px/sum(px);
355 if cdatflg==0
356 FFF=NN*sum(px.*logxx);
357 else
358 if abs(alph-1)<issmall, if alph<1 alph=1-issmall; else
359 alph=1+issmall; end; end;
360 switch testflg,
361 case 1,
362 % PRIMITIVE ESTIMATOR FOR xmax and xmin ....
363 dhi=0; dlo=0;
364 xmax2=xmax+dhi;
365 xmin2=xmin-dlo;
366 hxmax=xmax2.^((1-alph));
367 hxmin=xmin2.^((1-alph));
368 if hxmax>hxmin, h1=max(hxmax-hxmin,issmall);
369 else h1=-max(hxmin-hxmax,issmall);
370 end;
371 FFF=NN*((hxmax*log(xmax2)-hxmin*log(xmin2))/h1-1/(1-alph));
372 case 2,
373 % EXPERIMENTAL ESTIMATOR FOR xmax and xmin ....
374 % REM: works quite badly!!!

```

```

375     lxx=length(xx);
376     hxhi=xx(3:lxx);
377     hxlo=xx(1:(lxx-2));
378     hxmlid=xx(2:(lxx-1));
379     hf1=(hxhi-hxlo).*hxmlid.^ (1-alph);
380     hf2=hf1.*log(hxmlid);
381     FFF=NN*sum(hf2)/sum(hf2);
382     otherwise
383         error('... this should never happen!');
384     end;
385     end;
386     val=abs(QQQ-FFF);
387     if isnan(val)==0; d(count)=val; else d(count)=Inf; end;
388 end;
389 if length(d)==0,
390     msg=['dalphi too small -> break loop: ', dalphi, ' !!!\n'];
391     runinfo=[runinfo msg];
392     if infoflg==1, fprintf(1,msg); end;
393     break;
394 end;
395 v=find(d==min(d));
396 alphbest=alphav(v(1));
397 new_dalphi=4*dalphi/Ninc;
398 alphav=(alphbest-2*dalphi):new_dalphi:(alphbest+2*dalphi);
399 if dalphi<eps, break; end;
400 dalphi=new_dalphi;
401 end;

```

APPENDIX D2: r_plhistfit

```

1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 % Author: R Hanel
3 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
4 function out = r_plhistfit(k,varargin)
5 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
6 % latest modification on 12.07.2016
7 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
8 % arg
9 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
10 % k ... is the recorded data as a histogram
11 % By default we assume a linear binning that
12 % uses bins b_i=i+1/2 for bin k_i=|{x|b_i>x>b_{i-1}}|
13 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
14 if length(k)==length('help'),
15 if strcmp(k,'help')==1,
16     msg=['Using function out = r_plfit(data,varargin)\n'];
17     msg=[msg 'r_plhistfit can be used to fit the exponent out.exponent of
18           power laws\n'];
19     msg=[msg 'where data is given as histogram k=[k_1, ..., k_W] \n'];
20     msg=[msg 'The histogram is associated with bin margins b=[b_0, ...
21           b_W] \n'];
22     msg=[msg 'such that each event x counted in the i''th bin fulfills
23           b_i>x>b_{i-1}\n'];
24     msg=[msg '\n'];
25     msg=[msg 'Fitting binned histograms:\n'];
26     msg=[msg '\t By default r_plhistfit(k) assumes that k is a
27           histogram\n'];
28     msg=[msg '\t over bins with margins b_i=i+1/2 \n'];
29     msg=[msg '\t if other margins get used one can set them by using \n'];
30     msg=[msg '\t out = r_plhistfit(x,'margins',margins) ... where
31           margins is a vector [b_0,...,b_W] \n'];
32     msg=[msg '\t To specify a fit range zmax <= z_i <= zmin (default
33           zmin=min(x), zmax=max(x))\n'];
34     msg=[msg '\t ''rangemin'' and ''rangemax'' options can be used to set
35           the fit range\n'];

```

```

30     msg=[msg '\n'];
31     msg=[msg 'Automatic low frequency handling \n'];
32     msg=[msg '\t By default r_plhistfit does not merge bins in such a way
33             that each bin has at least Nmin elements \n'];
34     msg=[msg '\t the option ''lf'' switches on this lf handling \n'];
35     msg=[msg '\n'];
36     msg=[msg 'out = r_plfit(data,...,'plot') displays the fit over the
37             data\n'];
38     msg=[msg 'in double logarithmic coordinates (loglog plot)\n'];
39     msg=[msg "'fig'" behaves like 'plot' but explicitly opens a new
40             figure\n'];
41     msg=[msg "'exp_min'" can be used to specify the minimal search value
42             exp_min (default =0)\n'];
43     msg=[msg 'for out.exponent. Similarly
44             r_plfit(data,...,'exp_max',expmax) sets the minimal\n'];
45     msg=[msg 'search value (default =5) to expmax\n'];
46     msg=[msg "'eps'" ... set absolute error eps (default eps=le-5) for
47             out.exponent \n'];
48     msg=[msg '\n'];
49     msg=[msg 'Other options to control the performance of the
50             algorithm:\n'];
51     msg=[msg "'Ncut'" ... specifies the number of values alpha in the
52             search range\n'];
53     msg=[msg 'of out.exponent. After m iterations the precision of
54             out.exponent becomes\n'];
55     msg=[msg '(exp.max-exp.min)/(Ncut/2)^m =>
56             m(eps)^~2*(log(expmax-expmin)-log(eps))/Ncut\n'];
57     msg=[msg "'Nimplicit'" sets the maximal number of implicit iteration
58             for finding \nthe exponent (default 80) of iterations\n'];
59     msg=[msg 'searching implicitly for out.exponent. one should use
60             Nimplicit>m(eps)\n'];
61     msg=[msg "'info'" if set will output info over the run stored in
62             out.info at runtime!\n'];
63     msg=[msg "'autocenter'" assumes that bins have an overall offset
64             that can be optimized!\n'];
65     msg=[msg "'eps2'" can be used to set the search accuracy of the
66             'autocenter' option (default eps2=le-7)!\n'];
67     msg=[msg 'Bug reports to: rudolf.hanel@meduniwien.ac.at\n'];
68     fprintf(1,msg);
69     out=msg;
70     return;
71 end;
72 end;
73 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
74 % Initialization
75 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
76 % number of data items
77 Nbin=length(k);
78 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
79 % Default values
80 rangemin=0; rangemax=Inf; eps=le-5; eps2=le-7;
81 issmall=le-10; smallaszero=le-50;
82 alpha_min=0; alpha_max=5; xflg=0; lfflg=0;
83 plotflg=0; newfigflg=0; infoflg=0; centerflg=0;
84 centerfac=0; autocenterflg=0;
85 Nmin=1; Ncut=100; Nimplicit=500; Nautoc=50;
86 acenterf=0.999;
87
88 runinfo=['start ... \n'];
89 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
90 % varargin
91 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
92 id=0;
93 while id<length(varargin),

```

```

81      id=id+1;
82      if ischar(varargin{id}),
83          switch varargin{id},
84              case 'alpha_min', alpha_min = varargin{id+1}; id=id+1;
85              case 'alpha_max', alpha_max = varargin{id+1}; id=id+1;
86              case 'Nimplicit', Nimplicit = varargin{id+1}; id=id+1;
87              case 'Nautoc', Nautoc = varargin{id+1}; id=id+1;
88              case 'Nmin', Nmin = varargin{id+1}; id=id+1;
89              case 'Ncut', Ncut = varargin{id+1}; id=id+1;
90              case 'margins', margin = varargin{id+1}; xflg=1; id=id+1;
91              case 'rangemin', rangemin = varargin{id+1}; id=id+1;
92              case 'rangemax', rangemax = varargin{id+1}; id=id+1;
93              case 'lf', lfflg = 1;
94              case 'plot', plotflg=1;
95              case 'plot2', plotflg=2;
96              case 'fig', plotflg=1; newfigflg=1;
97              case 'eps', eps = varargin{id+1}; id=id+1;
98              case 'eps2', eps2 = varargin{id+1}; id=id+1;
99              case 'autocenter', autocenterflg = 1;
100      if centerflg==1,
101          msg=['The option ''autocenter'' supersedes ...
102              ' the options ''center'' and ''centering''!!!\n'];
103          runinfo=[runinfo msg];
104          if infoflg==1, fprintf(1,msg); end;
105          centerflg = 0;
106      end;
107      case 'acf', acenterf = varargin{id+1}; id=id+1;
108      case 'center', % Ok
109      if autocenterflg==1,
110          msg=['The option ''autocenter'' supersedes ...
111              ' the options ''center'' and ''centering''!!!\n'];
112          runinfo=[runinfo msg];
113          if infoflg==1, fprintf(1,msg); end;
114          centerflg = 0;
115      else
116          centerflg = 1;
117      end;
118      case 'centering', % Ok
119      if autocenterflg==1,
120          msg=['The option ''autocenter'' supersedes ...
121              ' the options ''center'' and ''centering''!!!\n'];
122          runinfo=[runinfo msg];
123          if infoflg==1, fprintf(1,msg); end;
124          centerflg = 0; centerfac = 0;
125      else
126          centerflg = 1;
127          centerfac = varargin{id+1};
128          centerfac=min(max(centerfac,-1),1);
129      end;
130      id=id+1;
131      case 'info',
132          infoflg=1;
133          otherwise ,
134          msg=['no such argument [ ', varargin{id} ' ] ->skip\n'];
135          runinfo=[runinfo msg];
136          if infoflg==1, fprintf(1,msg); end;
137      end
138  end
139 end
140 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
141 % handle histogram, data filtering etc ...
142 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
143 % data=histogram
144 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
145 % k=z;
146 msg=['handling histogram ...\\n'];

```

```

147 runinfo=[runinfo msg];
148 if infoflg==1, fprintf(1,msg); end;
149
150 %if input is histogram set range to...
151 switch xflg,
152 %...default binning if only histogram available
153 case 0,
154     msg=['setting default bins ...\\n'];
155     runinfo=[runinfo msg];
156     if infoflg==1, fprintf(1,msg); end;
157     margin=(1:(Nbin+1));
158     if centerflg==1, margin=margin-1/2*(1+(1-is small)*centerfac); end;
159 %...binning if margins are available
160 case 1,
161     if length(margin)^(Nbin+1),
162         msg=['length(margin)=', num2str(length(margin)), ' needst to
163             equal length(k)+1=',
164             num2str(length(k)+1),
165             ', !!!'];
166         error(msg);
167     end;
168     margin=sort(margin, 'ascend');
169 otherwise
170     error('this should not happen 1 !!!');
171 end;
172 % remark here bin [margin(i),margin(i+1)] is associated with k(i+1).
173 rangemin=max(rangemin,margin(1));
174 rangemax=min(rangemax,max(margin)+is small);
175 xidv=find(margin>=rangemin & margin<rangemax);
176 xmargin=margin(xidv); xNmarg=length(xmargin);
177 xNbin=xNmarg-1; kidv=xidv(2:xNmarg)-1; zk=k(kidv);
178 N=sum(k); W=length(k); NN=sum(zk); WW=length(zk);
179
180 if autocenterflg==1,
181     msg=['handling autocenter option ...\\n'];
182     runinfo=[runinfo msg];
183     if infoflg==1, fprintf(1,msg); end;
184     acenterf=min(max(acenterf,0),1-is small);
185     dbfac=rangemin*acenterf;
186     deltacv=2/Neut;
187     acv=(-1:deltacv:1)*dbfac;
188     deltacv=deltacv*dbfac;
189 else
190     Nautoc=1;
191     deltacv=0;
192     acv=[0];
193 end;
194 lacv=length(acv);
195
196 % if handling
197 if lfflg==1,
198     msg=['handling low frequency option ...\\n'];
199     runinfo=[runinfo msg];
200     if infoflg==1, fprintf(1,msg); end;
201
202     binsum=0;
203     lfmarg=[margin(1)];
204     lfk=[];
205     mid=0;
206     for id=1:WW,
207         binsum=binsum+xk(id);
208         if binsum>=Nmin,
209             mid=id+1;
210             lfmarg=[lfmarg xmargin(mid)];

```

```

211         lfk=[lfk binsum];
212         binsum=0;
213     end;
214 end
215 % if the last bin margin is not xmarg(WW+1)
216 lfW=length(lfmarg)-1;
217 lfmarg(lfW+1)=xmargin(WW+1);
218 lfk(lfW)=lfk(lfW)+binsum;
219 xmargin=lfmarg; xk=lfk; WW=lfW;
220 end;
221 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
222 % Maximum likelihood estimate & find optimal alpha
223 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
224 for rep1=1:Nautoc,
225 xKSV=zeros(1,lacv);
226 xAmIV=zeros(1,lacv);
227 xcount=0;
228 for dcent=acv,
229     xcount=xcount+1; xxmargin=xmargin+dcent; idrmax=WW+1;
230     xmax=max(xxmargin); xmin=min(xxmargin);
231     dalph=(alpha_max-alpha_min)/Ncut;
232     alphav=alpha_min:dalph:alpha_max;
233     idrmax=length(xxmargin); p=zeros(1,WW); f=xk/NN;
234     %% optimize alpha for Ncut alpha values
235     %xmargin0=xmargin;
236     msg=['start optimizing exponent ... \n'];
237     runinfo=[runinfo msg];
238     if infoflg==1, fprintf(1,msg); end;
239     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
240     % Inner Iterations
241     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
242     for rep2=1:Nimplicit,
243         d=zeros(size(alphav)); count=0;
244         for alph=alphav,
245             count=count+1;
246             if abs(alph-1)<smallaszero,
247                 hmarg=log(xxmargin);
248                 hxmax=log(xmax);
249                 hxmin=log(xmin);
250             else
251                 hmarg=xxmargin.^(1-alph);
252                 hxmax=xmax.^(1-alph);
253                 hxmin=xmin.^(1-alph);
254             end;
255             p=hmarg(2:(WW+1))-hmarg(1:WW);
256             p=p/(hxmax-hxmin);
257             h1=hmarg.*log(xxmargin);
258             h2=h1(2:(WW+1))-h1(1:WW);
259             h3=hxmax*log(xmax)-hxmin*log(xmin);
260             p(find(p<smallaszero))=smallaszero;
261             p=p/sum(p);
262             m=h2/h3;
263             d(count)=abs(sum(f.*m./p)-1);
264         end;
265         if length(d)==0,
266             msg=['dalphi too small -> break loop: ', dalph, ' !!!\n'];
267             runinfo=[runinfo msg];
268             if infoflg==1, fprintf(1,msg); end;
269             break;
270         end;
271         v=find(d==min(d));
272         alphbest=alphav(v(1));
273         new_dalph=4*dalph/Ncut;
274         alphav=(alphbest-2*dalph):new_dalph:(alphbest+2*dalph);
275         if dalph<eps, break; end;
276         dalph=new_dalph;

```

```

277     msg1=[ 'repetition: ', num2str([rep1,rep2]) ];
278     msg2=[ 'accuracy: ',
279             , num2str(log([deltacv/eps2,dalph/eps])./log(10)) ];
280     msg3=[ 'alpha: ', num2str(alphbest), ' ... \n' ];
281     msg=[msg1 msg2 msg3];
282     runinfo=[runinfo msg];
283     if infoflg==1, fprintf(1,msg); end;
284 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
285 % End inner iterations
286 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
287     Aml=alphbest;
288     if abs(Aml-1)<smallaszero,
289         hmarg=log(xxmargin);
290         hxmax=log(xmax);
291         hxmin=log(xmin);
292     else
293         hmarg=xxmargin.^ (1-Aml);
294         hxmax=xmax.^ (1-Aml);
295         hxmin=xmin.^ (1-Aml);
296     end;
297     pml=hmarg(2:(WW+1))-hmarg(1:WW);
298     pml=pml/(hxmax-hxmin);
299 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
300 % Quality of fit;
301 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
302     fml=xk/sum(xk);
303     KS = max(abs(cumsum(pml)-cumsum(fml)));
304     xAmIV(xcount)=Aml;
305     xKSV(xcount)=KS;
306 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
307     if plotflg >1,
308         xdelta=diff(xxmargin);
309         xcentroid=zeros(1,WW);
310         xcentroid=(xxmargin(1:WW)+xxmargin(2:(WW+1)))/2;
311         hold off;
312         loglog(xcentroid,fml./xdelta,'r');
313         hold on;
314         loglog(xcentroid,pml./xdelta),
315         pause(0.01);
316     end;
317 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
318     end;
319     v=min(find(xKSV==min(xKSV)));
320     xid=v(1);
321     Aml=xAmIV(xid);
322     KSV=xKSV(xid);
323     db=acv(xid);
324     acvmin=db-2*deltacv;
325     acvmax=db+2*deltacv;
326     deltaxv=4*deltacv/Ncut;
327     acv=acvmin:deltacv:acvmax;
328     lacv=length(acv);
329     if deltaxv<eps2, break; end;
330 end;
331 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
332 % End outer iterations
333 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
334 msg=['preparing output ... \n'];
335 runinfo=[runinfo msg];
336 if infoflg==1, fprintf(1,msg); end;
337
338 xxmargin=xmargin+db;
339 xmax=max(xxmargin);
340 xmin=min(xxmargin);

```

```

342 if abs(Aml-1)<smallaszero ,
343     hmarg=log(xxmargin);
344     hxmax=log(xmax);
345     hxmin=log(xmin);
346 else
347     hmarg=xxmargin.^ (1-Aml) ;
348     hxmax=xmax.^ (1-Aml) ;
349     hxmin=xmin.^ (1-Aml) ;
350 end;
351 pml=hmarg(2:(WW+1))-hmarg(1:WW);
352 pml=pml/(hxmax-hxmin);
353 %%%%%%%%%%%%%%
354 xcentroid=zeros(1,WW);
355 xcentroid=(xxmargin(1:WW)+xxmargin(2:(WW+1)))/2;
356 fml=xk/sum(xk);
357 %%%%%%%%%%%%%%
358 % plot
359 %%%%%%%%%%%%%%
360 xdelta=diff(xmargin); %/(max(xmargin)-min(xmargin));
361 hold off;
362 if plotflg >0,
363     if newfigflg==1, figure; end;
364     loglog(xcentroid,fml./xdelta,'r')
365     hold on;
366     loglog(xcentroid,pml./xdelta),
367     pause(0.01);
368 end;
369 %%%%%%%%%%%%%%
370 % output
371 %%%%%%%%%%%%%%
372 out.runinfo=runinfo; out.exponent = Aml;
373 out.KS = KS; out.xmin = xmin; out.xmax = xmax;
374 out.pml = pml; out.f = fml; out.centers=xcentroid;
375 out.xmargin=xmargin; out.offset=db; out.keff=xk;
376 out.fit.N = N; out.fit.Neff=NN;
377 out.fit.alpha_min=alpha_min; out.fit.alpha_max=alpha_max;
378 out.fit.Nmin = Nmin; out.fit.Nimplicit=Nimplicit;
379 out.fit.Ncut=Ncut; out.fit.lfmode = lf_flg;
380 out.fit.margin=margin; out.fit.k=k;
381 out.fit.delta=dalph;
382
383 return;

```

APPENDIX D3: r_plfit_calibrate

```

1 %%%%%%%%%%%%%%
2 % Author: R Hanel
3 %%%%%%%%%%%%%%
4 function out = r_plfit_calibrate(alpha,W, Nsamples, Nrep)
5 %%%%%%%%%%%%%%
6 % USE FOR SAMPLING CALIBRATION CURVES FOR
7 % the r_plfit KS statistics.
8 % r_plfit_calibrate samples from a power-law with exponent
9 % alpha Nsamples (can be a vector) samples then uses r_plfit to predict
10 % alpha and measures the corresponding KS values. This is repeated
11 % for Nrep times. By sorting the KS vector of length Nrep
12 % in a descending order one gets a curve KS(id) over p=id/Nrep, which
13 % represents the p-value of the KS value KS(id) of the r_plfit estimate
14 % This means that for KS(id) there exists a chance p=id/Nrep that we
15 % might sample a KS value more extreme than KS(id). This allows us to
16 % determine the critical KS value KS(id) for which we may expect
17 % to reject only a fraction p=id/Nrep of samples of size Nsamples
18 % that actually have been drawn from a power-law with exponent alpha
19 %%%%%%%%%%%%%%
20 % out ... is a struct that returns a summary of the input

```

```

21 % out.alpha=alpha;
22 % out.W=W;
23 % out.Nsample=Nsamples;
24 % out.Nrep=Nrep;
25 % out.KSS=KSS; ... the Nrep x length(Nsamples) matrix of KS
26 % statistics values
27 % out.LAM=LAM; ... the Nrep x length(Nsamples) matrix of reconst.
28 % exponents
29 % out.KSSsorted=KSSsorted; ... the sorted matrix of KS statistics matrix
30 % out.LAMsorted=LAMsorted; ... the sorted matrix of exponents (ordered
31 % in the same way as KS)
32 % out.Pval=Pvalv; ... the vector p=(1:Nrep)/Nrep
33 %
34 %%%%
35 % define distribution to sample from
36 p=(1:W).^( -alpha); p=p/sum(p);
37 L=length(Nsamples);
38 Nsm=max(Nsamples);
39 %%%%
40 % produce Nrep data sets
41 for id=1:Nrep,
42     fprintf(1,[ '----- togo= ', num2str(Nrep-id),
43                 '\n' ]);
44     data=randi(p,'p', 'Nsamples', Nsm);
45     % look at sample-sizes N
46     for id2=1:L;
47         N=Nsamples(id2);
48         [pdf1,dummy]=hist(data(1:N),1:W);
49         pdf1=pdf1/sum(pdf1);
50         % make the ML estimate of the exponent
51         x=r_plfit(data(1:N), 'noks');
52         % and compute the estimated power-law distribution
53         pdf2=(1:W).^x.exponent;
54         pdf2=pdf2/sum(pdf2);
55         % compute the KS value statistics
56         KSS(id,id2)=max(abs(cumsum(pdf1)-cumsum(pdf2)));
57         LAM(id,id2)=x.exponent;
58     end;
59 end
60 KSSsorted=zeros(size(KSS));
61 LAMsorted=zeros(size(LAM));
62 Pvalv=(1:Nrep)/Nrep;
63 KSScrit=zeros(Nrep,L);
64 % compute the p-value landscape
65 for id2=1:L,
66     [KSSv,I]=sort(KSS(:,id2), 'descend');
67     LAMv=LAM(I,id2);
68     KSSsorted(:,id2)=KSSv;
69     LAMsorted(:,id2)=LAMv;
70 end
71 % set the output
72 out.alpha=alpha;
73 out.W=W;
74 out.Nrep=Nrep;
75 out.KSS=KSS;
76 out.LAM=LAM;
77 out.KSSsorted=KSSsorted;
78 out.LAMsorted=LAMsorted;
79 out.Pval=Pvalv;
80 out.Nsample=Nsamples;

```

APPENDIX D4: r_plfit_calib_eval

```

1 %%%%%%
2 % Author: R Hanel
3 %%%%%%
4 function out = r_plfit_calib_eval(in,confidence,samplesize,plotflg)
5 %%%%%%
6 % finds the critical KS value from the confidence p-value confidence
7 % for the samples-size samplesize from in=r_plfit_calibrate(...) and
8 % returns
9 % out.confidence
10 % out.samplesize
11 % out.Nsample ... vector of sample sizes (r_plfit_calibrate)
12 % out.Pval ... (r_plfit_calibrate)
13 % out.KS ... selected (by confidence) and interpolated KS vector of
14 % length(out.Nsamples)
15 % out.KScrit ... selected (by confidence and samplesize) and interpolated
16 % critical KS value
17 %%%%%%
18 plotflg=round(plotflg);
19 if plotflg >2, plotflg=2; end;
20 if plotflg <0, plotflg=0; end;
21 %plotflg=2; % plot a calibration curve 1D + 2D
22 %plotflg=1; % plot a calibration curve 1D
23 %plotflg=0; % no plots
24 %%%%%%
25 L=length(in.Nsample);
26 Nrep=in.Nrep;
27 out.confidence=confidence;
28 out.samplesize=samplesize;
29 out.Nsample=in.Nsample;
30 out.Pval=in.Pval;
31 v=find(abs(in.Pval-confidence)==min(abs(in.Pval-confidence)));
32 cid=min(v);
33 if in.Pval(cid)<confidence,
34 if cid>1,
35 a=(in.Pval(cid)-confidence)/(in.Pval(cid)-in.Pval(cid-1));
36 b=(confidence-in.Pval(cid-1))/(in.Pval(cid)-in.Pval(cid-1));
37 out.KS=a*in.KSSsorted(cid,:)+b*in.KSSsorted(cid-1,:);
38 else
39 out.KS=in.KSSsorted(cid,:);
40 end;
41 else
42 if cid<length(in.Pval),
43 a=(in.Pval(cid+1)-confidence)/(in.Pval(cid+1)-in.Pval(cid));
44 b=(confidence-in.Pval(cid))/(in.Pval(cid+1)-in.Pval(cid));
45 out.KS=a*in.KSSsorted(cid+1,:)+b*in.KSSsorted(cid,:);
46 else
47 out.KS=in.KSSsorted(cid,:);
48 end;
49 end;
50 v=find(abs(in.Nsample-samplesize)==min(abs(in.Nsample-samplesize)));
51 sid=min(v);
52 if in.Nsample(sid)>samplesize,
53 if sid>1,
54 a=(in.Nsample(sid)-samplesize)/(in.Nsample(sid)-in.Nsample(sid-1));
55 b=(samplesize-in.Nsample(sid-1))/(in.Nsample(sid)-in.Nsample(sid-1));
56 out.KScrit=a*out.KS(sid)+b*out.KS(sid-1);
57 else
58 out.KScrit=out.KS(sid);
59 end;
60 else
61 if sid<length(in.Nsample),
62 a=(in.Nsample(sid+1)-samplesize)/(in.Nsample(sid+1)-in.Nsample(sid));
63 b=(samplesize-in.Nsample(sid))/(in.Nsample(sid+1)-in.Nsample(sid));
64 out.KScrit=a*out.KS(sid+1)+b*out.KS(sid);
65 else

```

```

64         out.KScrit=out.KS(sid);
65     end;
66 end;
67
68 if plotflg==1,
69 figure;
70 L=length(out.KS);
71 plot(out.Nsample,out.KS);
72 xlabel('sample size');
73 ylabel('KS');
74 end;
75 if plotflg==2,
76 figure;
77 subplot(2,2,1);
78 mesh(in.Nsample,in.Pval,in.LAMsorted);
79 xlabel('sample size');
80 ylabel('confid.');
81 zlabel('\lambda');
82 subplot(2,2,2);
83 mesh(in.Nsample,in.Pval,in.KSSsorted);
84 xlabel('sample size');
85 ylabel('confid.');
86 zlabel('KS');
87 subplot(2,2,3);
88 for id=1:L
89 gval=id/L;
90 plot(in.Pval,in.KSSsorted(:,id),'Color',[1,gval,1-gval]);
91 hold on;
92 end;
93 hold off;
94 xlabel('confid.');
95 ylabel('KS');
96 subplot(2,2,4);
97 for id=1:Nrep
98 gval=id/Nrep;
99 plot(in.Nsample,in.KSSsorted(id,:),'Color',[1,1-gval,gval]);
100 hold on;
101 end;
102 hold off;
103 xlabel('sample size');
104 ylabel('KS');
105 end;
106 return;

```

APPENDIX D5: r_randi

```

1 %%%%%%
2 % Author: R Hanel
3 %%%%%%
4 function n=r_randi(X,varargin),
5 %%%%%%
6 % varargin:
7 % a) n=r_randi(X)
8 % X ... can be a vector of numbers > 0
9 % n ... returns a vector of randomnumbers with 1<=n(id)<=X(id)
10 % b) n=r_randi(X,'Nsamples',Nsamples)
11 % X ... is a number > 0
12 % Nsamples ... is a number > 0
13 % n ... returns a vector of length Nsamples with randomnumbers
14 % 1<=n(id)<=X
15 % c) n=r_randi(X,'p')
16 % X ... is a real vector > 0 representing a distribution function
17 % X gets normalized by r_randi to X/sum(X)
18 % n ... returns a number drawn randomly from id=1:length(X)
19 % with probability X(id)/sum(X) of;
20 % if the 'Nsamples' option is used n is a vector of length(X)

```

```

20 % c) n=r_randi(X, 'cp')
21 % X ... is a real monotonically increasing vector > 0 representing a
22 % cummulative
23 % distribution function; X gets normalized by r_randi to
24 % X/X(length(X))
25 % n ... returns a number drawn randomly from the distribution
26 % characterized by the cummulative distribution
26 id=0;
27 isdist=0;
28 Nsamples=1;
29 while id<length(varargin),
30     id=id+1;
31     if ischar(varargin{id}),
32         switch varargin{id},
33             case 'Nsamples',
34                 % X is distribution
35                 Nsamples = varargin{id+1};
36                 id=id+1;
37             case 'p',
38                 % X is distribution
39                 isdist = 1;
40             case 'cp',
41                 % X is cummulative distribution
42                 isdist = 2;
43             otherwise,
44                 fprintf(1,['no such argument [ ' varargin{id} ' ] ->skip\n']);
45             end
46         end
47     end
48 %
49 n=zeros(1,Nsamples);
50 if isdist==0,
51     W=max(round(X),1);
52     if length(X)>1,
53         Nsamples=length(X);
54         n=zeros(1,Nsamples);
55         n=min(1+floor(rand(size(W)).*W),W);
56     else
57         for id=1:Nsamples,
58             n(id)=min(1+floor(rand(size(W)).*W),W);
59         end;
60     end
61 elseif isdist==2
62     if X(length(X))~=1, X=X/X(length(X)); end;
63     for id=1:Nsamples,
64         v=find(rand<=X);
65         n(id)=min(v);
66     end;
67 else
68     p=X/sum(X);
69     csp=cumsum(p);
70     for id=1:Nsamples,
71         % if mod(id,1000)==0, Nsamples=id, pause(0.1); end;
72         v=find(rand<=csp);
73         n(id)=min(v);
74     end;
75 end;
76
77 return;

```