
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, Josefstädterstrasse 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
11         laws\n'];
12    msg=[msg 'in three basic different modes\n'];
13    msg=[msg '(1) out = r_plfit(x) with data being samples x=[x1,...,xN]
14         from N experiments\n'];
15    msg=[msg '\t in this mode out.exponent returns the estimated exponent
16         lambda of \n'];
17    msg=[msg '\t the distribution p(z_i|lambda) one has sampled from\n'];
18    msg=[msg '(2) out = r_plfit(k,\'hist\') with data being histograms
19         k=[x1,...,xW]\n'];
20    msg=[msg '\t of W distinct events i=1..W with magnitudes
21         z=[z1,...,zW]\n'];
22    msg=[msg '\t here out.exponent also returns lambda\n'];
23    msg=[msg '(3) out = r_plfit(k) with data being histograms
24         k=[x1,...,xW]\n'];
25    msg=[msg '\t works equivalent to (1) but with k as data one computes
26         the\n'];
27    msg=[msg '\t frequency distribution of x with exponent
28         alpha^-1+1/lambda\n'];
29    msg=[msg '\n'];
30    msg=[msg 'Fitting with data x:\n'];
31    msg=[msg '\t By default r_plfit(x) assumes that x consists of natural
32         numbers and\n'];
33    msg=[msg '\t the sample space (magnitudes) z={i|min(x)<=i<=max(x)}\n'];
34    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
      msg uses:\n'];
27 msg=[msg '\t out = r_plfit(x, ''urange'') ... which sets
      msg z=[z_1, ..., z_W]=unique(x)\n'];
28 msg=[msg '\t where W is then the number of unique values of data points
      msg in x\n'];
29 msg=[msg '\t out =
      msg r_plfit(x, ''urange'', ''rangemin'', zmin, ''rangemax'', zmax) can be
      msg used\n'];
30 msg=[msg '\t to specify a fit range zmax <= z_i <= zmin (default
      msg zmin=min(x), zmax=max(x))\n'];
31 msg=[msg '\t To control the data range individually use out =
      msg r_plfit(x, ''range'', z)\n'];
32 msg=[msg '\n'];
33 msg=[msg 'Fitting with histograms k\n'];
34 msg=[msg '\t Using k (mode 3) for fitting the frequency distribution
      msg with exponent alpha\n'];
35 msg=[msg '\t works in exactly the same way as the sample distribution
      msg estimating the\n'];
36 msg=[msg '\t exponent lambda using x.\n'];
37 msg=[msg '\t out = r_plfit(k, ''hist'') works similar however one should
      msg note that\n'];
38 msg=[msg '\t in this mode r_plfit assumes by default that
      msg z=[1, 2, ..., W]\n'];
39 msg=[msg '\t The option ''urange'' has no effect in this mode and gets
      msg ignored if set\n'];
40 msg=[msg '\t Otherwise use out = r_plfit(k, ''hist'', ''range'', z) to set
      msg the event magnitudes z\n'];
41 msg=[msg '\t ''rangemin'' and ''rangemax'' options work in exactly the
      msg same way as before\n'];
42 msg=[msg '\n'];
43 msg=[msg 'Automatic low frequency cutoff \n'];
44 msg=[msg '\t By default r_plfit runs an iterative search for an optimal
      msg low frequency cutoff\n'];
45 msg=[msg '\t If cutoff is set at index i where |Nz_i^lambda/N-Nmin| is
      msg minimal where Nmin=1\n'];
46 msg=[msg '\t by default and N the length of x=[x_1, ..., x_N]; Nmin can be
      msg set using the\n'];
47 msg=[msg '\t ''Nmin'' option. If maxrange is smaller than the predicted
      msg lf cutoff then the\n'];
48 msg=[msg '\t cutoff has no effect. Note that in the out = r_plfit(k)
      msg mode the lf cutoff mechanism\n'];
49 msg=[msg '\t effectively acts as a high frequency cutoff with respect to
      msg the data x\n'];
50 msg=[msg '\t the option ''nolf'' switches of the lf cutoff \n'];
51 msg=[msg '\n'];
52 msg=[msg 'out = r_plfit(data, ..., ''plot'') displays the fit over the
      msg data\n'];
53 msg=[msg 'in double logarithmic coordinates (loglog plot)\n'];
54 msg=[msg ''fig'' behaves like ''plot'' but explicitly opens a new
      msg figure\n'];
55 msg=[msg ''exp_min'' can be used to specify the minimal search value
      msg exp_min (default =0)\n'];
56 msg=[msg 'for out.exponent. Similarly
      msg r_plfit(data, ..., ''exp_max'', expmax) sets the minimal\n'];
57 msg=[msg 'search value (default =5) to expmax\n'];
58 msg=[msg ''eps'' ... set absolute error eps (default eps=1e-10) for
      msg out.exponent \n'];
59 msg=[msg '\n'];
60 msg=[msg 'Other options to control the performance of the
      msg algorithm:\n'];
61 msg=[msg ''Ncut'' ... specifies the number of values alpha in the
      msg search range\n'];
62 msg=[msg 'of out.exponent. After m iterations the precision of
      msg 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            % for cont dat we require the urange
105            case 'cdat', cdatflg=1; lfcutoff=0; xflg=2;
106            case 'cdat2', cdatflg=1; testflg=2; lfcutoff=0; xflg=2;
107            case 'hist', histflg=1;
108            case 'nolf', lfcutoff=0;
109            case 'info', infoflg=1;
110            case 'plot', plotflg=1;
111            case 'fig', plotflg=1; newfigflg=1;
112            case 'cleanrange', cleanxflg=1;
113            case 'eps', eps = varargin{id+1}; id=id+1;
114            otherwise, msg=['no such argument [' varargin{id} '] ->skip
115                            ... \n'];
116            runinfo=[runinfo msg]; if infoflg==1, fprintf(1,msg); end;
117        end
118    end
119 end
120 msg=['handling arguments ... \n'];
121 runinfo=[runinfo msg]; if infoflg==1, fprintf(1,msg); end;
122 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
123 % handle hist property, data filtering etc ...
124 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
125 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     k=z; 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 else
155 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
156 % data=samples; clean data
157 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
158     msg=['cleaning data input ... \n'];
159     runinfo=[runinfo msg];
160     if infoflg==1, fprintf(1,msg); end;
161     idv=find(z>0); zz=z(idv); idv=find(abs(log(zz))<Inf);
162     zz=zz(idv); idv=find(isnan(zz)==0); zz=zz(idv);
163     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
164     rangemin=max(rangemin,min(zz));
165     rangemax=min(rangemax,max(zz));
166     if rangemin>rangemax,
167         fprintf(1,runinfo);
168         error('r_plfit requires rangemin<rangemax!');
169     end;
170     zidv=find(zz>=rangemin & zz<=rangemax); zz=zz(zidv);
171     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
172     if (((max(zz)-min(zz))>Nrangeitems_max) & (xflg==0)),
173         msg1=['Length of r_plfit default range rangemin:1:rangemax ' ...
174             num2str((max(zz)-min(zz))) ' is
175             too large!\n'];
176         msg2=['Maximum allowed: ' num2str(Nrangeitems_max) ...
177             '\n']; => r_plfit tries the urange
178             option!\n'];
179         msg3=['Otherwise, try again by setting the range property
180             manually with options: ' ...
181             ' range, maxrange, minrange!\n'];
182         runinfo=[runinfo msg1 msg2 msg3];
183         if infoflg==1, fprintf(1,[msg1 msg2 msg3]); end; xflg=2;
184     end;
185     msg=['handling data input case ' num2str(xflg), ' ...\n'];
186     runinfo=[runinfo msg];
187     if infoflg==1, fprintf(1,msg); end;
188     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             v=find(uz(uid)==x);
196             if length(v)>0, uzon(uid)=1; end;
197         end;
198         uzoffidv=find(uzon==0);
199         for uid=1:length(uzoffidv),
200             v=find(uz(uzoffidv(uid))==zz);
201             if length(v)>0, zz(v)=-1; end;
202         end;
203         v=find(zz>0); zz=zz(v);
204                                     %... urange property is specified
205     case 2,
206         x=unique(zz);
207         rangemin=max(rangemin,min(x));
208         rangemax=min(rangemax,max(x));
209     otherwise
210         fprintf(1,runinfo);
211         error('this should not happen 1 !!!');
212     end;
213 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
214 % sort data & filter sample-space
215 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
216     zz=sort(zz);
217     if cleanxflg==1,
218         msg=['cleaning sample space ...\n'];
219         runinfo=[runinfo msg];
220         if infoflg==1, fprintf(1,msg); end;
221         uz=unique(zz); xon=zeros(size(x));
222         for xid=1:length(uz),
223             v=find(uz==x(xid));
224             if length(v)>0, xon(xid)=1; end;
225         end;
226         x=x(find(xon==1));
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 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
248 % Maximum likelihood estimate: find ML optimal alpha
249 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
250 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
251 xrangemax=max(xx); xrangemax2=xrangemax;
252 idrmax=WW; idrmax_1=0; idrmax_2=0; QQ=kk.*log(xx);
253 xmin=rangemin; xmax=rangemax; alphbest=0;
254
255 param.Nimplicit=Nimplicit; param.Ninc=Ninc;
256 param.alpha_max=alpha_max; param.alpha_min=alpha_min;
257 param.cdatflg=cdatflg; param.testflg=testflg;
258 param.infoflg=infoflg; param.issmall=issmall;
259
260 if lfcutoff==1,
261 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
262 %% optimize low frequency cut-off for ML optimal alpha
263     idrmax_1=WW;
264     msg=['find exponent (lf cut-off is on) ...\n'];
265     runinfo=[runinfo msg];
266     if infoflg==1, fprintf(1,msg); end;
267     for repl=1:Ntail,
268         xridv=find(xx<=xrangemax2); idrmax_2=idrmax_1;
269         idrmax_1=idrmax; idrmax=max(xridv); xxx=xx(xridv);
270         kkk=kk(xridv); WWW=length(xridv); NNN=sum(kkk); QQ=QQ(xridv);
271 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
272 %% optimize alpha for Ninc alpha values
273         loopmsg=['lf-loop: ', num2str(repl), ' '];
274         [alphbest, runinfo]=h_implicit(NNN,xxx,QQ,runinfo,loopmsg,alphbest,param);
275         Aml=alphbest; yml=xx.^(-Aml); yml=yml/sum(yml);
276         gugu=abs(NN*yml-Nmin); v=find(gugu==min(gugu));
277         if Aml>Amlthres, idrmax=max(v); else idrmax=WW; end;
278         if (idrmax==idrmax_1 | idrmax==idrmax_2) & repl>=rep_min, break;
279         end;
280         idrmax=max(idrmax,1); xrangemax2=xx(idrmax);
281     end;
282 else
283     msg=['find exponent (lf cut-off is off) ...\n'];
284     runinfo=[runinfo msg];
285     if infoflg==1, fprintf(1,msg); end;
286     [alphbest, runinfo]=h_implicit(NN,xx,QQ,runinfo,'',alphbest,param);
287     Aml=alphbest; idrmax=WW;
288 end
289 msg=['preparing output ...\n'];
290 runinfo=[runinfo msg];
291 if infoflg==1, fprintf(1,msg); end;
292
293 xlf cutoff=1: idrmax;
294
295 % Quality of fit;
296 xxx=xx(xlf cutoff);
297 plf=xxx.^(-Aml); %Aml,
298 p=plf/sum(plf);
299 p=xxx.^(-Aml); %Aml,
300 p=p/sum(p);
301
302 ylf=kk(xlf cutoff)/sum(kk(xlf cutoff)); y=kk/sum(kk);
303 if plotflg==1,
304     if newfigflg==1, figure; end;
305     loglog(xxx, ylf, 'r')
306     hold on;
307     loglog(xxx, plf),
308     loglog(xxx(idrmax), plf(idrmax), 'go'),
309     pause(0.01);
310 end;
311

```

```

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.KSrange = 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             %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
361             switch testflg ,
362                 case 1,
363                     % PRIMITIVE ESTIMATOR FOR xmax and xmin ....
364                     dhi=0; dlo=0;
365                     xmax2=xmax+dhi;
366                     xmin2=xmin-dlo;
367                     hxmax=xmax2.^(1-alpha);
368                     hxmin=xmin2.^(1-alpha);
369                     if hxmax>hxmin, h1=max(hxmax-hxmin, issmall);
370                     else h1=-max(hxmin-hxmax, issmall);
371                     end;
372                     FFF=NN*((hxmax*log(xmax2)-hxmin*log(xmin2))/h1-1/(1-alpha));
373                 case 2,
374                     % EXPERIMENTAL ESTIMATOR FOR xmax and xmin ....
375                     % REM: works quite badly!!!

```

```

375         lxx=length(xx);
376         hxhi=xx(3:lxx);
377         hxlo=xx(1:(lxx-2));
378         hxmid=xx(2:(lxx-1));
379         hf1=(hxhi-hxlo).*hxmid.^(1-alpha);
380         hf2=hf1.*log(hxmid);
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=['dalpha too small -> break loop: ', dalph, ' !!!\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_dalph=4*dalph/Ninc;
398 alphav=(alphbest-2*dalph):new_dalph:(alphbest+2*dalph);
399 if dalph<eps, break; end;
400 dalph=new_dalph;
401 end;

```

APPENDIX D2: r_plhistfit

```

1
2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3 % Author: R Hanel
4 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5 function out = r_plhistfit(k,varargin)
6 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
7 % latest modification on 12.07.2016
8 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
9 % arg
10 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
11 % k ... is the recorded data as a histogram
12 %     By default we assume a linear binning that
13 %     uses bins b_i=i+1/2 for bin k_i={x|b_i>x>b_{i-1}}
14 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
15 if length(k)==length('help'),
16     if strcmp(k,'help')==1,
17         msg=['Using function out = r_plfit(data,varargin)\n'];
18         msg=[msg 'r_plhistfit can be used to fit the exponent out.exponent of
19             power laws\n'];
20         msg=[msg 'where data is given as histogram k=[k_1, ... , k_W] \n'];
21         msg=[msg 'The histogram is associated with bin margins b=[b_0, ... ,
22             b_W] \n'];
23         msg=[msg 'such that each event x counted in the i''th bin fulfills
24             b_i>x>b_{i-1}\n'];
25         msg=[msg '\n'];
26         msg=[msg 'Fitting binned histograms:\n'];
27         msg=[msg '\t By default r_plhistfit(k) assumes that k is a
28             histogram\n'];
29         msg=[msg '\t over bins with margins b_i=i+1/2 \n'];
30         msg=[msg '\t if other margins get used one can set them by using \n'];
31         msg=[msg '\t out = r_plhistfit(x,''margins'',margins) ... where
32             margins is a vector [b_0,...,b_W] \n'];
33         msg=[msg '\t To specify a fit range zmax <= z_i <= zmin (default
34             zmin=min(x), zmax=max(x))\n'];
35         msg=[msg '\t ''rangemin'' and ''rangemax'' options can be used to set
36             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=1e-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=1e-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=1e-5; eps2=1e-7;
81 issmall=1e-10; smallaszero=1e-50;
82 alpha_min=0; alpha_max=5; xflg=0; lflg=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 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
91 % varargin
92 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
93 id=0;
94 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', lf_flg = 1;
94             case 'plot', plot_flg=1;
95             case 'plot2', plot_flg=2;
96             case 'fig', plot_flg=1; newfig_flg=1;
97             case 'eps', eps = varargin{id+1}; id=id+1;
98             case 'eps2', eps2 = varargin{id+1}; id=id+1;
99             case 'autocenter', autocenter_flg = 1;
100            if center_flg==1,
101                msg=['The option ''autocenter'' supersedes' ...
102                    ' the options ''center'' and ''centering''!!!\n'];
103                runinfo=[runinfo msg];
104                if info_flg==1, fprintf(1,msg); end;
105                center_flg = 0;
106            end;
107            case 'acf', acenterf = varargin{id+1}; id=id+1;
108            case 'center', % Ok
109                if autocenter_flg==1,
110                    msg=['The option ''autocenter'' supersedes' ...
111                        ' the options ''center'' and ''centering''!!!\n'];
112                    runinfo=[runinfo msg];
113                    if info_flg==1, fprintf(1,msg); end;
114                    center_flg = 0;
115                else
116                    center_flg = 1;
117                end;
118            case 'centering', % Ok
119                if autocenter_flg==1,
120                    msg=['The option ''autocenter'' supersedes' ...
121                        ' the options ''center'' and ''centering''!!!\n'];
122                    runinfo=[runinfo msg];
123                    if info_flg==1, fprintf(1,msg); end;
124                    center_flg = 0; centerfac = 0;
125                else
126                    center_flg = 1;
127                    centerfac = varargin{id+1};
128                    centerfac=min(max(centerfac,-1),1);
129                end;
130                id=id+1;
131                case 'info',
132                    info_flg=1;
133                otherwise,
134                    msg=['no such argument [ ' varargin{id} ' ] ->skip\n'];
135                    runinfo=[runinfo msg];
136                    if info_flg==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-issmall)*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             error(msg);
166         end;
167         margin=sort(margin, 'ascend');
168     otherwise
169         error('this should not happen 1 !!!');
170 end;
171 % remark here bin [margin(i),margin(i+1)] is associated with k(i+1).
172 rangemin=max(rangemin, margin(1));
173 rangemax=min(rangemax, max(margin)+issmall);
174 xidv=find(margin>=rangemin & margin<rangemax);
175 xmarg=xmargin(xidv); xNmarg=length(xmargin);
176 xNbin=xNmarg-1; kidv=xidv(2:xNmarg)-1; xk=k(kidv);
177 N=sum(k); W=length(k); NN=sum(xk); WW=length(xk);
178
179 if autocenterflg==1,
180     msg=['handling autocenter option ...\n'];
181     runinfo=[runinfo msg];
182     if infoflg==1, fprintf(1,msg); end;
183     acenterf=min(max(acynterf,0),1-issmall);
184     dbfac=rangemin*acynterf;
185     deltacv=2/Ncut;
186     acv=(-1:deltacv:1)*dbfac;
187     deltacv=deltacv*dbfac;
188 else
189     Nautoc=1;
190     deltacv=0;
191     acv=[0];
192 end;
193 lacv=length(acv);
194
195 % If handling
196 %~~~~~
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 repl=1:Nautoc,
225     xKSV=zeros(1,lacv);
226     xAmlV=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=['dalph 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     end;
285     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
286     % End inner iterations
287     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
288     Aml=alphbest;
289     if abs(Aml-1)<smallaszero ,
290         hmarg=log(xxmargin);
291         hxmax=log(xmax);
292         hxmin=log(xmin);
293     else
294         hmarg=xxmargin.^(1-Aml);
295         hxmax=xmax.^(1-Aml);
296         hxmin=xmin.^(1-Aml);
297     end;
298     pml=hmarg(2:(WW+1))-hmarg(1:WW);
299     pml=pml/(hxmax-hxmin);
300     % Quality of fit;
301     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
302     fml=xk/sum(xk);
303     KS = max(abs(cumsum(pml)-cumsum(fml)));
304     xAmlV(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=xAmlV(xid);
322     KSV=xKSV(xid);
323     db=acv(xid);
324     acvmin=db-2*deltacv;
325     acvmax=db+2*deltacv;
326     deltacv=4*deltacv/Ncut;
327     acv=acvmin:deltacv:acvmax;
328     lacv=length(acv);
329     if deltacv<eps2, break; end;
330 end;
331 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
332 % End outer iterations
333 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
334
335 msg=['preparing output ...\n'];
336 runinfo=[runinfo msg];
337 if infoflg==1, fprintf(1,msg); end;
338
339 xxmargin=xmargin+db;
340 xmax=max(xxmargin);
341 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
11 % and measures the corresponding KS values. This is repeated
12 % for Nrep times. By sorting the KS vector of length Nrep
13 % in a descending order one gets a curve KS(id) over p=id/Nrep, which
14 % represents the p-value of the KS value KS(id) of the r_plfit estimate
15 % This means that for KS(id) there exists a chance p=id/Nrep that we
16 % might sample a KS value more extreme than KS(id). This allows us to
17 % determine the critical KS value KS(id) for which we may expect
18 % to reject only a fraction p=id/Nrep of samples of size Nsamples
19 % that actually have been drawn from a power-law with exponent alpha
20 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
21 % 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
    statistics values
26 % out.LAM=LAM; ... the Nrep x length(Nsamples) matrix of reconst.
    exponents
27 % out.KSSsorted=KSSsorted; ... the sorted matrix of KS statistics matrix
28 % out.LAMsorted=LAMsorted; ... the sorted matrix of exponents (ordered
    in the same way as KS)
29 % out.Pval=Pvalv; ... the vector p=(1:Nrep)/Nrep
30 %
31 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
32 % define distribution to sample from
33 p=(1:W).^(-alpha); p=p/sum(p);
34 L=length(Nsamples);
35 Nsm=max(Nsamples);
36 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
37 LAM=zeros(Nrep,L); % the expected exponent
38 KSS=zeros(Nrep,L); % the KS statistics variable
39 % produce Nrep data sets
40 for id=1:Nrep,
41     fprintf(1,['----- togo=', num2str(Nrep-id),
42               '\n']);
43     data=r_randi(p, 'p', 'Nsamples',Nsm);
44     % look at sample-sizes N
45     for id2=1:L;
46         N=Nsamples(id2);
47         [pdf1,dummy]=hist(data(1:N),1:W);
48         pdf1=pdf1/sum(pdf1);
49         % make the ML estimate of the exponent
50         x=r_plfit(data(1:N), 'noks');
51         % and compute the estimated power-law distribution
52         pdf2=(1:W).^x.exponent;
53         pdf2=pdf2/sum(pdf2);
54         % compute the KS value statistics
55         KSS(id, id2) = max(abs(cumsum(pdf1)-cumsum(pdf2)));
56         LAM(id, id2) = x.exponent;
57     end;
58 KSSsorted=zeros(size(KSS));
59 LAMsorted=zeros(size(LAM));
60 Pvalv=(1:Nrep)/Nrep;
61 KSScrit=zeros(Nrep,L);
62 % compute the p-value landscape
63 for id2=1:L,
64     [KSSv, I]=sort(KSS(:, id2), 'descend');
65     LAMv=LAM(I, id2);
66     KSSsorted(:, id2)=KSSv;
67     LAMsorted(:, id2)=LAMv;
68 end
69
70 % set the output
71 out.alpha=alpha;
72 out.W=W;
73 out.Nrep=Nrep;
74 out.KSS=KSS;
75 out.LAM=LAM;
76 out.KSSsorted=KSSsorted;
77 out.LAMsorted=LAMsorted;
78 out.Pval=Pvalv;
79 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
28 out.confidence=confidence;
29 out.samplesize=samplesize;
30 out.Nsample=in.Nsample;
31 out.Pval=in.Pval;
32 v=find(abs(in.Pval-confidence)==min(abs(in.Pval-confidence)));
33 cid=min(v);
34 if in.Pval(cid)<confidence,
35     if cid>1,
36         a=(in.Pval(cid)-confidence)/(in.Pval(cid)-in.Pval(cid-1));
37         b=(confidence-in.Pval(cid-1))/(in.Pval(cid)-in.Pval(cid-1));
38         out.KS=a*in.KSSsorted(cid,:)+b*in.KSSsorted(cid-1,:);
39     else
40         out.KS=in.KSSsorted(cid,:);
41     end;
42 else
43     if cid<length(in.Pval),
44         a=(in.Pval(cid+1)-confidence)/(in.Pval(cid+1)-in.Pval(cid));
45         b=(confidence-in.Pval(cid))/(in.Pval(cid+1)-in.Pval(cid));
46         out.KS=a*in.KSSsorted(cid+1,:)+b*in.KSSsorted(cid,:);
47     else
48         out.KS=in.KSSsorted(cid,:);
49     end;
50 end;
51 v=find(abs(in.Nsample-samplesize)==min(abs(in.Nsample-samplesize)));
52 sid=min(v);
53 if in.Nsample(sid)>samplesize,
54     if sid>1,
55         a=(in.Nsample(sid)-samplesize)/(in.Nsample(sid)-in.Nsample(sid-1));
56         b=(samplesize-in.Nsample(sid-1))/(in.Nsample(sid)-in.Nsample(sid-1));
57         out.KScrit=a*out.KS(sid)+b*out.KS(sid-1);
58     else
59         out.KScrit=out.KS(sid);
60     end;
61 else
62     if sid<length(in.Nsample),
63         a=(in.Nsample(sid+1)-samplesize)/(in.Nsample(sid+1)-in.Nsample(sid));
64         b=(samplesize-in.Nsample(sid))/(in.Nsample(sid+1)-in.Nsample(sid));
65         out.KScrit=a*out.KS(sid+1)+b*out.KS(sid);
66     else
67         out.KScrit=out.KS(sid);
68     end;
69 end;
70
71 out.KS=out.KS(out.Nsample);
72 out.KScrit=out.KScrit(out.Nsample);
73
74 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```



```

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
    cumulative
22 %     distribution function; X gets normalized by r_randi to
    X/X(length(X))
23 % n ... returns a number drawn randomly from the distribution
24 %     characterized by the cumulative distribution
25 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
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 cumulative 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;

```