Permutationstest bei komplexen Teststrategien

Albert Rosenberger
Universitätsmedizin Göttingen
Institut für Genetische Epidemiologie
Humboldtallee 32
37073 Göttingen
arosenb@gwdg.de

Zusammenfassung

Permutationstests sind eine vielseitig anwendbare Alternative zu herkömmlichen parametrischen Verfahren zum Testen von Hypothesen. Der größte Nachteil liegt im teils intensiven Re-Sampling, das zu einem hohen Bedarf an Rechenzeit, Rechenressourcen und Speicherplatz führen kann. PROC NPAR1WAY und PROC MULTTEST bieten bereits die Möglichkeit, Permutationsverfahren für etablierte statistische Tests durchzuführen. „Early Stopping" und das Implementieren komplexer Testverfahren ist jedoch nicht möglich. Ein SAS-Makro wird vorgestellt, mit dem beides umgesetzt werden kann. „Early Stopping" wird erzwungen, wenn a) die Chance auf ein signifikantes Ergebnis gering wird oder b) eine hochsignifikantes Ergebnis bereits erreicht wurde. Im Zweiten Fall wird die Robustheit des resultierenden p-Werts erhöht, in dem eine Pareto-Verteilung an die extremsten Verteilungen der simulierten Test-Statistik angepasst, und aus dieser Verteilung der p-Wert ableitet wird.

Schlüsselwörter: Permutationstest, Early stopping rules, Extrapolation

1 Einleitung

Permutationstests sind eine vielseitig anwendbare Alternative zu herkömmlichen parametrischen Verfahren zum Testen von Hypothesen [1], ebenso wie auf Rängen basierende Verfahren oder nicht-parametrische Bootstrap-Tests. Das zugrundeliegende Konzept ist einfach und gut verständlich. Bei Permutationstests wird die Verteilung der Teststatistik unter der Nullhypothese (Null-Verteilung) durch wiederholtes, zufälliges Anordnen der Einflussgrößen (Permutieren) iterativ erzeugt. Diese Testverfahren werden daher zu den rechenintensiven Re-Sampling-Verfahren, ebenso wie Bootstrap- oder Jackknife-Methoden, gezählt. All diese Verfahren haben den Vorteil, dass sie (nahezu uneingeschränkt) für Daten jeglichen Skalenniveaus anwendbar sind, unabhängig von Verteilungstypen, Abhängigkeits- oder Clustercharakteristik einzelner Variablen oder Beobachtungen [2].

Ebenso wie für nicht-parametrische Verfahren müssen keine oft unrealistischen Verteilungsannahmen getroffen werden [2]. Jedoch, trotz dieser Möglichkeiten und Einsetzbarkeit sollten Re-Sampling-Verfahren mit einem gewissen Grad an Vorsicht angewandt werden. Es sollte immer bedacht werden, dass – ebenso wie Bootstrapverfahren –

ausschließlich die zur Hand liegende Daten Eingang in die Testverfahren nehmen. "*Unless certain basic ideas are understood, it is all too easy to produce a solution to the wrong problem, or a bad solution to the right problem (conditional how appropriate to the problem which information is permuted how)*" [2].

Der größte Nachteil liegt im teils intensiven Re-Sampling, das zu einem hohen Bedarf an Rechenzeit, Rechenressourcen und Speicherplatz führen kann. Das ist vor allem bei sehr niedrigen Signifikanzniveaus der Fall, wie sie zum Beispiel im Rahmen von genomweiten Assoziationsstudien (z. B: $\alpha \sim 10^{-7}$) üblich sind. Ebenso sollte stets bedacht werden, dass die resultieren p-Werte auf Grund der lediglich finiten Anzahl an Permutationen nur approximativ und daher mit einer gewissen Ungenauigkeit versehen sind.
PROC NPAR1WAY und PROC MULTTEST [3] bieten bereits die Möglichkeit, Permutationstest als exakte Alternativen zu herkömmlichen parametrischen Testverfahren durchzuführen, wie etwa dem Cochran-Armitage Test auf lineare Trends oder den **t**-Test. Um darüber hinaus Permutationsverfahren anzuwenden, kann etwa eine ausreichende Anzahl permutierter „SRSWOR-Re-Samples" (simple random sampling without replacement) mit der Prozedur PROC SURVEYSELECT [3] effizient erzeugt und danach ausgewertet werden. Diese Analyse kann in jedem PROC-Step relative einfach durch die BY-Anweisung realisiert werden, um so die Null-Verteilung z. B. der Test-Statistik zu generieren. Auf diese Weise kann der Permutationsprozess jedoch nicht ständig überwacht werden. „Early stopping" von Applikationen, bei denen es abzusehen wäre, dass sie zu keinem signifikanten Ergebnis führen werden, ist nicht möglich.
Sind die zu analysierende Datensätze jedoch umfangreich oder sind die Verfahrensschritte des angewandten Tests komplex (erstrecken sich ggf. über mehrere PROC- und DATA-Steps), kann eine allgemeinere Strategie eine Herausforderung an deren Programmierung stellen.

In diesem Artikel wird ein SAS-Makro präsentiert, mit dem Permutationstests auch für komplexe Testverfahren durchgeführt werden können und das ein „early stopping" ermöglicht.
Zwei Maßnahmen wurden dabei getroffen, um den Aufwand, sprich die Anzahl an Permutationsschritten so gering wie möglich, aber so hoch wie nötig zu halten. Erstens, ein Algorithmus zum sequentiellen prüfen der „Chance auf eine signifikantes Ergebnis" wurde implementiert. Damit wird die Permutation gestoppt, falls sich ein $p-\text{Wert} \gg \propto$ abzeichnet [4]. Zweitens, falls die beobachtete Test-Statistik extrem weit in den Schwänzen der permutierten Null-Verteilung liegt ($p-\text{Wert} \ll \propto$), wird die Permutation ebenfalls gestoppt. An den „Schwanz" wird dann eine GPD-Verteilung (generalized pareto distribution) angepasst und der p-Wert extra- bzw. interpoliert [5].

2 Notation

Y	Zielgröße(n)
X	Einflussgröße(n)
Z	Störgröße(n)
M_0	Test-Statistik bei gegebener Zuordnung der Ziel-, Einfluss-, und Störgrößen
M_j	Test-Statistik bei j-ter Permutation der Ziel-, Einfluss-, und Störgrößen
m_{max}	maximale Anzahl an Permutationen
m*	Anzahl „überschreitender Permutationen“
p_{permut}	p-Werte des Permutationsverfahrens
α	Signifikanzniveau
RR_p	Re-Sampling-Risiko

3 Die Permutations-Strategie

In Anlehnung an den 5-schrittingen Algorithmus von Good [2] ergibt sich folgende Permutationsstrategie, bei der unnötige Permutationen vermieden werden:

1 Wahl einer geeigneten Teststrategie
   a. analysiere das Problem,
   b. lege die Null-, und Alternativ-Hypothese fest,
   c. bestimme den Schaden einer falschen Testentscheidung und lege das Signifikanzniveau α fest,
   d. wähle die maximal notwendige Anzahl Permutationen m_{max}. Diese orientiert sich entweder an der maximalen Laufzeit oder an der gewünschten Präzision eines p-Werts für $p_{permut} \sim \alpha$.
2 Wähle eine geeignete, suffiziente Test-Statistik *M* zum Unterscheiden von Null- und Alternative-Hypothese.
3 Berechne die Test-Statistik M_0 bei gegebener Zuordnung der Ziel-, Einfluss-, und Störgrößen.
4 Durchführen der Permutation
   a. k-maliges permutieren der Zielgröße (n) Y (mit $k<m_{max}$),

b. berechne die Test-Statistiken M_j (j=1 to $m = \sum k$) bei permutierter Zuordnung der Ziel-, Einfluss-, und Störgrößen.

5 Vergleiche M_0 mit der Null-Verteilung von M_j und entscheide:

a. die Chance eine $p_{permut} \leq \alpha$ zu erzielen ist gering: $p_{permut} \gg \propto$
→‘“early stopping“: bestimme p_{permut} regulär

b. die Chance eine $p_{permut} \leq \alpha$ zu erzielen ist weiterhin gegeben
→ Permutation fortsetzen bei Schritt 4

c. ein extrem kleiner p-Wert $p_{permut} \ll \propto$ wurde erzielt ($m^* \widehat{=}$ ist klein)
→ bestimme p_{permut} anhand einer GPD extremer M_j-Werte

d. m_{max} ist erreicht → bestimme p_{permut} *regulär.*

4 Reguläre und irreguläre p-Werte

4.1 Reguläre Bestimmung des p-Werts

Bei einem einseitigen Test wird die statistische Signifikanz, quantifiziert als p-Wert, bestimmt durch den Anteil an Permutationen die eine Test-Statistik M_j (*j=1 to m*) ergeben, die mindestens so groß (oder größer) ist als die Test-Statistik M_0 bei unpermutierter (originaler) Zuordnung der Ziel- und Einflussgrößen. Zu diesem Zweck wird $m^* = n(M_j \geq M_0)$, die Anzahl “überschreitender Permutationen” bestimmt. Der exakte p-Wert wird im Weiteren als $p_{permut} = P(M_j \geq M_0) = m^* + 1/m + 1$ berechnet. Dadurch wird ein $p_{permut}=0$ vermieden, der bei m*=0 auftreten kann, also wenn die beobachtete Test-Statistik M_0 größer ist als alle durch Permutation erzeugten Test-Statistiken *Mj* [6].

4.2 Early Stopping

Wurden beispielsweise in den ersten k=50 Permutationen immerhin m*=49-mal $M_j > M_0$ gefunden, so ist es unwahrscheinlich, das in folgenden Permutationen $p_{permut} \leq \alpha$ unterschritten wird. Ein Fortsetzen des Permutationsprozesses erscheint sinnlos.
Die Anzahl notwendiger Permutationen kann durch den “*Algorithm of sequential Monte Carlo testing”* [5] begrenzt werden. Dieser kontrolliert das Re-Sampling-Risiko eines falsch-signifikanten Ergebnisses, definiert als:

$$RR_p(\widehat{p}) = \begin{cases} p_p(p > \alpha) & if\ p \leq \alpha \\ p_p(p \leq \alpha) & if\ p > \alpha \end{cases}.$$

Ziel dieser Methode ist es, einen Bereich $[L_m, U_m]$ für m* (gegeben m bereits durchgeführter Permutationen) zu definieren, so dass das kleinste Re-Sampling-Ri-

siko $sup_{p \in [0,1]} RR_p \leq \epsilon$ ist. Damit kann man sich für „early stopping" entscheiden, falls m* außerhalb $[L_m, U_m]$ fällt und damit die Chance einen p-Wert um das Signifikanzniveau α zu erzielen klein wird, wobei das multiple Überprüfen – ähnlich wie beim sequentiellen Testen – berücksichtigt wird (siehe Abbildung 1). Im vorgestellten SAS-Makro ist „spending sequence" als $\varepsilon_n = \varepsilon \frac{n}{1000+n}$, mit ε=0.01 implementiert. Der Algorithmus ist im Detail bei Gandy [4] beschrieben. Das Permutationsverfahren stoppt somit bei $m<m_{max}$ bei klar insignifikanten aber auch bei hoch signifikanten Ergebnissen.

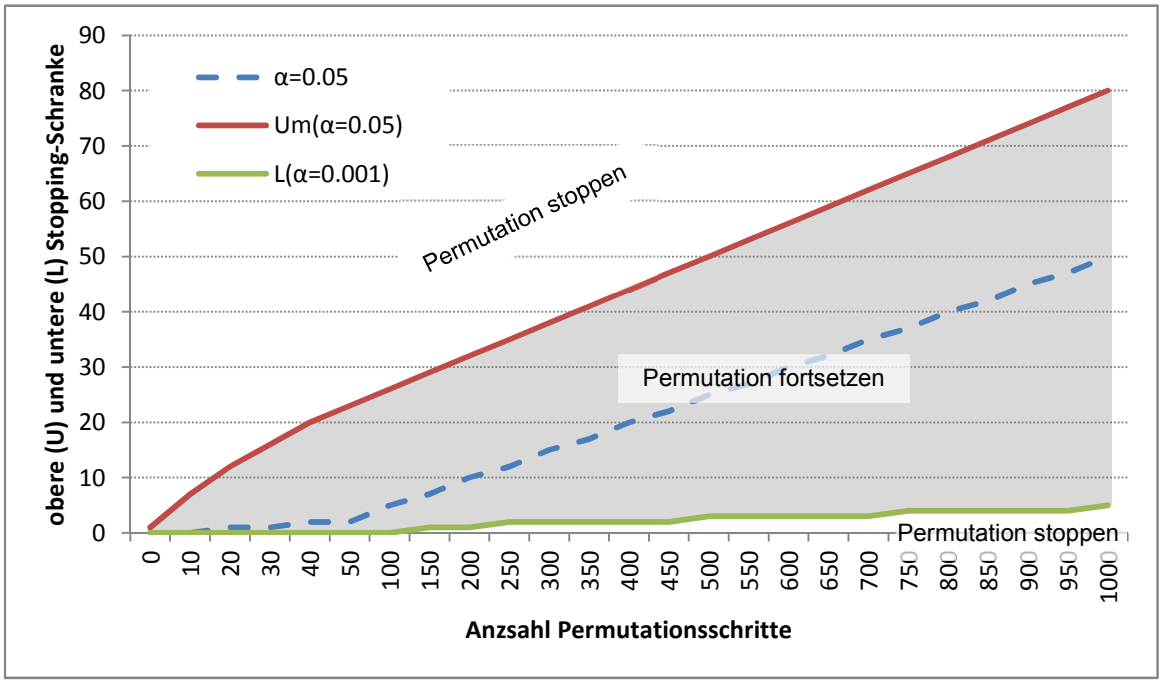

Abbildung 1: Obere und Untere Schranke eines Stopping-Bereichs

4.3 Inter-/Extrapolation extremer p-Werte

Wird bei der Überprüfung auf ein mögliches "early stopping" festgestellt, dass die Anzahl „überschreitender Permutationen" m* sehr klein ist (z. B.: m*≤10 bei m=1000 Permutationen), wird der Schwanz der Null-Verteilung aller M_j durch eine „generalized pareto distribution" (GPD) approximiert [5]. Für dieses Vorgehen konnte eine, gegenüber der regulären Bestimmung von p_{permut}, robustere Bestimmung gezeigt werden, insbesondere bei sehr kleinen Signifikanzniveaus.
Abweichend von der empfohlenen Approximation zu den 250 größten M_j-Werten wird unter den 100 bis 500 größten M_j-Werten (in Schritten zu je 50) die best-angepasste GPD bestimmt. Goodness of fit tests (GOF) werden durchgeführt, um die Reliabilität der Approximation zu prüfen. Im Fall, dass GPD nicht adäquat angepasst werden kann, wird ersatzweise eine Exponentialverteilung herangezogen (als Spezialfall der GPD).

5 Das Makro

Diese allgemeine Permutationsstrategie ist im SAS-Makro `%permut_ablauf` implementiert. Die vom Anwender ausgewählte komplexe Teststatistik muss dazu ebenfalls in einem Makro `%permutationstest(TD,TR,TM,gruppe,GS,repeat)` programmiert werden.
Folgende notwendigen Makrovariablen müssen dabei spezifiziert werden:

```
TD…       Testdatei (dataset)
TR…       Ergebnisdatei (dataset)
TM…       Test-Methode(n) (char. Variable)
gruppe…   Gruppen-Variable (untersuchte Einflussgröße)
GS…       Variable für stratifizierte Auswertung (via BY-
          Statement)
repeat…   Variable für wiederholte Auswertung (via BY-Statement)
```

Die Ergebnisdatei TR muss die Variablen TS (Test-Statistik) und TM (Test-Methode) enthalten. GS und Repeat können synonym und gleichzeitig verwendet werden. Zum Ablauf des Permutationsverfahrens sollten folgende 4 Schritte durchgeführt werden:

1 Generiere/Importiere die Testdaten TD
  Die Daten einer Beobachtung (eines Falls) sollten in einer Zeile enthalten sein. Die untersuchte Einflussgröße (z. B. Behandlungsgruppe) sollte in einer Spalte (Variable) erfasst sein.
2 Erstelle das Makro %permutationstest
3 Führe %permutationstest mit der ursprünglichen Gruppeneinteilung aus
  a. Ausgabedatei sollte mit TR_original benannt werden
  b. Benenne die Variable TS in TS_original um
4 Spezifiziere den Permutationsablauf und führe ihn aus

```
TD…            Testdaten
gruppe…        Gruppen-Variable (untersuchte Einflussgröße)
id…            ID pro Fall innerhalb der Gruppen
testside…      Test-Seitigkeit: "2", "1 (low) " oder "1 (high)"
TR_original…   Datei mit Originalergebnissen
TM…            Test-Methode(n) (char. Variable)
max_permut…    Anzahl maximaler Permutationen
max_step…      Anzahl maximaler Durchläufe/Inspektionen für Early
               Stopping (z.B. 30)
n_per_step…    Permutationen pro Durchlauf
               (Liste, z.B. 500 250 100) Die letzte Angabe wird
               bis zu max_step wiederholt, begrenzt durch
               max_permut.
histogram…     Ausgabe von Grafiken (ja, nein)
pace…          FAST oder SLOW (werden nur 2 Gruppen verglichen,
               so kann mit der Option FAST der
               Permutationsalgorithmus beschleunigt werden)
```

Das Makro läuft sowohl unter Windows als auch unter UNIX/LINUX. Zur Unterscheidung des Betriebssystems sollte die Anweisung `%let system=MS;` bzw. `%let system=UNIX;` verwendet werden. Ebenso muss eine Library SASout zum Zwischensichern der Ergebnisse spezifiziert werden: `libname SASout "C:\Temp";`

6 Ein Beispiel

Messungen eines x-beliebigen Experiments von 20 Fällen und 80 Kontrollen sollen verglichen werden, wobei eine genetische Prädisposition durch 25 Sets bestehend aus jeweils 50 Genen berücksichtigt werden soll. Dazu wird zunächst mittels PCA die genetische Information der ersten 5 Hauptkomponenten ermittelt. Danach sollen diese als Störgrößen in einem linearen Modell (GLM) berücksichtigt werden, wobei durch Selektion das best-passende Model (max. AIC-Kriterium) ausgewählt werden soll. Durch die PCA entstehen Abhängigkeiten zwischen den Beobachtungen. Durch die Selektion unterminiert die Berechnung der Freiheitsgrade im GLM. Die Verteilung der Teststatistik folgt somit nicht mehr der GLM-Theorie.

1 Generiere die Testdaten TD (siehe ANHANG)
2 Erstelle das Makro % permutationstest

```
%macro permutationstest(TD,TR,TM,gruppe,GS,repeat);
ods results off;
ods listing close;
   proc sort data=&TD.; by &GS. &gruppe.; run;
   proc princomp data=&TD. out=TDneu n=5 noprint;
        var d_1-d_50;  by &GS. &gruppe.;
   run;
   proc glmselect data=TDneu;
        ods select ParameterEstimates SelectedEffects;
        model wert=&gruppe. prin1-prin5 /
             include=1 SELECTION=backward SHOWPVALUES select=AIC;
        by &GS.;
        ods output
ParameterEstimates=&TR.(where=(effect="&gruppe.")
                  rename=(tValue=TS)
                  keep=&repeat. &GS. effect tValue Probt);
   run; quit;
ods results on;
ods listing;
   data &TR.; set &TR.(drop=effect); &TM.='complex ttest';
   proc datasets nolist; delete TDneu; quit;
%mend permutationstest;
```

3 Führe %permutationstest mit der ursprünglichen Gruppeneinteilung aus

```
%permutationstest(TD=TD,TR=TR_original,TM=TM,gruppe=CC,GS=GS_Nr);
   data TR_original;   set TR_original(rename=(TS=TS_original));
                            label TS_original='original TS';
   run;
```

4 Spezifiziere den Permutationsablauf und führe ihn aus

```
%permut_ablauf(   TD=TD,  gruppe=CC, GS=GS_Nr, id=id,
                  testside=1 (high),
                  TR_original=TR_original,
                  TM=TM,
                  n_per_step=200 100, max_step=9, max_permut=1000,
                  histogram=nein,     pace=FAST);
 proc print data=perm_result ;
   format p_wert percent percent8.3;
 run;
```

Die Ergebnisse sind in Tabelle 1 im Anhang dargestellt. Bei 17 der 25 Tests konnte die Permutation bereits nach 50 Durchläufen abgebrochen werden. Der Permutationsaufwand konnte damit auf $(17 * 50 + 8 * 1000)/(25 * 1000) = 35\%$ reduziert werden.
Vier (GS-Nr. 1-4) der 25 Tests zeigten ein klar signifikantes Ergebnis, bei dreien wurde der p-Wert über eine GPD inter-/extrapoliert. Ein weiterer Test (GS-Nr. 16) zeigte ein klar nicht-signifikantes Ergebnis. Bei 3 Tests (GS-Nr. 13, 19 und 25) kann von den, durch die Permutation ermittelten, p-Werten nicht eindeutig auf Signifikanz geschlossen werden, da die 95%-Konfidenzintervalle das Signifikanzniveau α=5% einschließen. Für diese wäre es zweckmäßig, die Anzahl maximaler Permutationen zu erhöhen.

Literatur

[1] Fisher RA. The design of experiments. Edinburgh, London,: Oliver and Boyde; 1935.

[2] Good P. Permutation Tests: a practical Guide to Resampling Methods for Testing Hypotheses. New York: Springer-Verlag; 1995.

[3] Publishing SAS. Base SAS 9.2 Procedures Guide. Sas Inst; 2009.

[4] Gandy A. Sequential Implementation of Monte Carlo Tests With Uniformly Bounded Resampling Risk. Journal of the American Statistical Association 2009;104(488):1504-11.

[5] Knijnenburg TA, Wessels LFA, Reinders MJT, et al. Fewer permutations, more accurate P-values. Bioinformatics 2009;25(12):i161-i8.

[6] Phipson B, Smyth GK. Permutation P-values should never be zero: calculating exact P-values when permutations are randomly drawn. StatApplGenet MolBiol 2010;9(1):Article39.

7 Anhang

```
********* 1) Generiere die Testdaten TD     **********;
   data TD(drop=seed j n);
     seed=12356;
     array d_ [2] d_1-d_50;
     do GS_Nr=1 to 25;
     do CC=0,1;
          if CC=0 then n=20; if CC=1 then n=80;
          do id=1 to n;
               if GS_Nr<5 then wert=RAND('UNIFORM')+CC/2/GS_Nr;
               else wert=RAND('UNIFORM');
               do j=1 to dim(d_); d_[j]=RAND('BERNOULLI',0.5)*2-1;
   end;
               output;
          end;
     end;
     end;
   run;
```

Tabelle 1: Erzielte p-Werte mit Konfidenzgrenzen.

GS-Nr.	Grund für „Early Stopping“	Anzahl Permutationen	p-Wert	95%-CI	
1	max. Durchläufe erreicht (clear)*	1000	**0.0010**	**<.0001**	**0.0074**
2	max. Durchläufe erreicht (clear)*	1000	**0.0020**	**0.0001**	**0.0092**
3	max. Durchläufe erreicht (clear)*	1000	**0.0010**	**<.0001**	**0.0074**
4	max. Durchläufe erreicht (clear)	1000	**0.0190**	**0.0097**	**0.0331**
5	upper boundary crossed	50	0.6471	0.4577	0.8083
6	upper boundary crossed	50	0.3922	0.2237	0.5813
7	upper boundary crossed	50	0.6471	0.4577	0.8083
8	upper boundary crossed	50	0.9412	0.8010	0.9933
9	upper boundary crossed	50	0.7255	0.5390	0.8686
10	upper boundary crossed	50	0.4314	0.2568	0.6191
11	upper boundary crossed	50	0.6863	0.4977	0.8391
12	upper boundary crossed	50	0.7451	0.5602	0.8828
13	max. Durchläufe erreicht (uncertain)	1000	**0.0559**	**0.0389**	0.0774
14	upper boundary crossed	50	0.6078	0.4187	0.7763
15	upper boundary crossed	50	0.6275	0.4381	0.7925
16	max. Durchläufe erreicht (clear)	1000	0.0699	0.0508	0.0933
17	upper boundary crossed	50	0.3333	0.1761	0.5225
18	upper boundary crossed	50	0.6275	0.4381	0.7925
19	max. Durchläufe erreicht (uncertain)	1000	**0.0569**	**0.0397**	0.0785
20	upper boundary crossed	50	0.8039	0.6261	0.9230
21	upper boundary crossed	50	0.3333	0.1761	0.5225
22	upper boundary crossed	300	0.1296	0.0843	0.1870
23	upper boundary crossed	50	0.8824	0.7210	0.9688
24	upper boundary crossed	50	0.8039	0.6261	0.9230
25	max. Durchläufe erreicht (uncertain)	1000	**0.0440**	**0.0290**	0.0634

95%.CI…95% Konfidenzintervall des p-Werts.
* … diese p-Werte wurde mittel GPD inter/extrapoliert