Permutationstests mit Stata

Felix Bittmann

kontakt@felix-bittmann.de

V.1.0 / Oktober 2020

Besonders nach der Durchführung eines Experiments ist der Vergleich zweier Gruppen eine Standardaufgabe der angewandten Statistik. Dazu wird meistens ein Zweistichproben-t-Test durchgeführt und getestet, ob sich das arithmetische Mittel beider Gruppen signifikant unterscheidet. Auf diese Weise kann in einem Experimentalsetting der Effekt eines Treatments nachgewiesen werden. Allerdings sind Ergebnisse eines regulären t-Tests oftmals zweifelhaft, besonders, wenn schiefe Verteilungen vorliegen. Berechnete p-Werte können damit irreführend sein. Als robuste Alternative bieten sich Permutationstests an, welche weniger Annahmen machen und präzisere Ergebnisse bieten.

Keywords: Permutation, Permutationstest, t-Test, Gruppenvergleich, Experiment, Stata

Dieses Dokument kann wie folgt zitiert werden:

Bittmann, Felix (2020): Permutationstests mit Stata. Abgerufen von: http://felix-bittmann.de/downloads/artikel/permutationstest_stata.pdf am [DATUM]

Inhaltsverzeichnis

1 Schwächen des t-Tests

Die Nachteile und Schwächen des üblichen Zweistichproben-t-Test sind seit längerer Zeit bekannt. Besonders bei schiefen Verteilungen können die berechneten p-Werte irreführend sein (Sutton 1993). Folgende zwei Annahmen sind häufig verletzt: a.) das untersuchte Merkmal ist in der Population der beiden Gruppen normalverteilt. Dies mag zutreffen oder auch nicht, viele Merkmale sind offenbar *nicht* normalverteilt (man denke beispielsweise an Einkommen). b.) Die Varianzen der beiden Gruppen sind in der Population annähernd gleich (Varianzhomogenität). Auch diese Annahme ist oftmals verletzt, was die Testergebnisse beeinträchtigen kann. Damit liegen bereits zwei zentrale Annahmen vor, die verletzt werden können und damit als Schwächen des t-Tests zu werten sind.

Ein Blick in die Geschichte dieses Tests ist ebenfalls sehr aufschlussreich. Ursprüngliche wurde der t-Test als *Approximation* des Permutationstests entwickelt, d.h. er wurde eingeführt, um den notwendigen Rechenaufwand zu reduzieren. Dass damit Nachteile einhergehen, liegt auf der Hand. Angesichts der enormen Rechenkapazitäten, die dem Durchschnittsnutzer mittlerweile auf Standardhardware zur Verfügung stehen, gibt es wenig Gründe, den t-Test beizubehalten. Sollten demnach der reguläre t-Test und ein Permutationstest zu unterschiedlichen Schlussfolgerungen kommen, so ist der Permutationstest zweifelsohne vorzuziehen (Hesterberg 2015).

2 Logik des Permutationstests

Der Permutationstest macht diese Annahmen nicht und ist auch für Variablen geeignet, die nicht intervallskaliert sind (eine ordinale Skala ist ausreichend, wenn etwa Schulnoten untersucht werden). Zudem ist der Permutationstest in zweierlei Hinsicht flexibel: es können beliebige Statistiken zwischen zwei Gruppen vergleichen werden, also nicht nur das arithmetische Mittel, sondern beispielsweise auch der Median, aber auch alle anderen Punktschätzer. Weiterhin kann die Präzision des Tests durch die Anzahl der zu testenden Permutationen in der Monte Carlo Version beliebig eingestellt werden, sodass die Rechenzeit angepasst werden kann. Dabei gilt, dass die Präzision mit der Anzahl der zufälligen Permutationen ansteigt (Good 2013).

Die Logik des Tests ist simpel und basiert auf elementarer Kombinatorik. Um dies zu illustrieren wird ein einfaches Beispiel gewählt mit den Gruppen G1 und G2. Beide Gruppen enthalten jeweils drei Messpunkte. Diese sind wie folgt:

$$G1 = [35, 55, 78] \qquad x_1 = 56$$

$$G2 = [68, 79, 81] \qquad x_2 = 76$$

Die Gruppenmittelwerte unterscheiden sich offenbar, die Differenz der Mittelwerte ist $diff = x_1 - x_2 = -20$. Es stellt sich nun die Frage, ob dieser Unterschied statistisch signifikant ist. Wir nehmen an, dass diese Daten aus einem Experiment stammen, das Treatment demnach von den VersuchsleiterInnen appliziert wurde und eine Randomisierung stattgefunden hat. Demnach sollten alle Gruppenunterschiede alleine auf die Wirkung des Treatments zurückzuführen sein. Wir stellen die Frage: Wie hoch ist die Wahrscheinlichkeit, dass bei einer rein zufälligen Gruppenzuordnung

ein identisches oder gar noch extremeres Ergebnis zustande gekommen wäre? Damit ist gemeint, dass eine Differenz von -20 oder ein noch extremeres Ergebnis vorliegt (d.h. kleiner als -20). Zur Veranschaulichung bilden wir zunächst eine Gesamtgruppe, in der alle Datenpunkte enthalten sind.

$$G = [35, 55, 78, 68, 79, 81]$$

Nun stellen wir die kleinere Gruppengröße fest. Da im Beispiel die Gruppengrößen identisch sind ist dies gleichgültig und der Wert ist 3. Hier wird deutlich, dass der Test auch dann angewendet werden kann, wenn sich die Gruppengrößen unterscheiden. Nun formen wir alle denkbaren Permutationen, d.h. alle Möglichkeiten der Gruppenaufteilung. Wichtig ist dabei, dass wir alle Elemente erhalten und keine doppelt zählen.[1]

G1*	**G2***	**diff**
35, 55, 78	68, 79, 81	-20*
35, 55, 68	78, 79, 81	-26.7*
35, 55, 79	78, 68, 81	-19.3
35, 55, 81	78, 68, 79	-18
35, 78, 68	55, 79, 81	-11.3
35, 78, 79	55, 68, 81	-4
35, 78, 81	55, 68, 79	-2.7
35, 68, 79	55, 78, 81	-10.7
35, 68, 81	55, 78, 79	-9.3
35, 79, 81	55, 78, 68	-2
55, 78, 68	35, 79, 81	2
55, 78, 79	35, 68, 81	9.3
55, 78, 81	35, 68, 79	10.7
55, 68, 79	35, 78, 81	2.7
55, 68, 81	35, 78, 79	4
55, 79, 81	35, 78, 68	11.3
78, 68, 79	35, 55, 81	18
78, 68, 81	35, 55, 79	19.3
78, 79, 81	35, 55, 68	26.7
68, 79, 81	35, 55, 78	20

Deutlich wird, dass die Reihenfolge innerhalb einer Gruppe ohne Bedeutung ist, deshalb sind diese für eine bessere Übersicht nach Größe sortiert. Wir sehen jeweils auch die Differenz der Mittelwerte. Dabei wird erkennbar, dass nur zwei Werte gleich oder extremer sind (markiert mit einem Sternchen). Das eine ist das faktisch erhaltene Ergebnis, also genau -20. Die Gesamtzahl der Permutationen ist 20. Der p-Wert berechnet sich nun als der Anteil der gleichen oder noch extremeren Ergebnisse an der Gesamtzahl, demnach 2 / 20 = 0.10. Es wird anhand des Beispiels deutlich, dass das faktische Ergebnisse immer im Zähler steht, ein p-Wert von exakt 0 ist somit nicht möglich. Es sei darauf hingewiesen, dass noch andere Berechnungsmethoden existieren (Phipson & Smyth 2010).

Es wird zwischen dem exhaustiven Permutationstest und der Monte Carlo Implementierung unterschieden. Hier gezeigt wurde ein exhaustiver Tests, d.h. alle denkbaren Permutationen sind berücksichtigt. Dies ist allerdings nur bei kleinen Samples durchführbar, da die Anzahl exponentiell

1 Für eine Berechnung aller Permutationen kann Python genutzt werden mit dem Modul *itertools*.

Ansteigt und daher ab einer bestimmten Gruppengröße nicht mehr alle berücksichtigt werden können. In diesen Fällen wird die zweite Version gewählt, in der zufällige Samples gezogen werden. Sofern die Anzahl dieser Zufallssamples groß genug ist wird der exakte Wert der exhaustiven Version approximiert.

3 Durchführung in Stata

Die Durchführung in Stata wird über den Befehl *permute* möglich, der die Monte Carlo Version implementiert. Die Handhabung ist einfach. Jedoch muss mitunter ein kleines Hilfsprogramm geschrieben werden, um die Differenz der Gruppen zu berechnen. Wir gehen davon aus, dass wir am arithmetischen Mittel interessiert sind. Wir geben zunächst unsere Daten ein und berechnen die empirische Differenz (der vollständige Code ist im Appendix zum schnellen Kopieren dokumentiert). Für mehr Informationen zur allgemeinen Stata Syntax siehe Bittmann (2019).

```
. clear all

. version 16.1

. input y gruppe

            y     gruppe
  1.        35 0
  2.        55 0
  3.        78 0
  4.        68 1
  5.        79 1
  6.        81 1
  7. end

. bysort gruppe: sum y
```

```
-> gruppe = 0

    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
           y |          3          56    21.51743         35         78

-> gruppe = 1

    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
           y |          3          76            7         68         81
```

Nun können wir das Hilfsprogramm definieren und testen.

```
. cap program drop diff

. program define diff, rclass
  1.         quiet sum y if gruppe == 0, detail
  2.         local m1 = r(mean)
  3.         quiet sum y if gruppe == 1, detail
  4.         local m2 = r(mean)
  5.         return scalar diff = `m1' - `m2'
  6. end

. diff

. return list

scalars:
              r(diff) =  -20
```

Das empirische Ergebnis von -20 wird korrekt produziert. Wären wir am Median interessiert, so würden wir *r(mean)* durch *r(p50)* austauschen, da *summarize* in diesem Skalar den Median abspeichert. Es folgt nun die eigentliche Durchführung mit *permute*.

```
. permute gruppe r(diff), reps(5000) seed(123) nodots: diff

Warning:  Because diff is not an estimation command or does not set e(sample), permute has no way to determine
          which observations are used in calculating the statistics and so assumes that all observations are
          used.  This means that no observations will be excluded from the resampling because of missing
          values or other reasons.

          If the assumption is not true, press Break, save the data, and drop the observations that are to be
          excluded.  Be sure that the dataset in memory contains only the relevant data.

Monte Carlo permutation results          Number of observations  =          6
                                         Number of permutations  =      5,000

      command:  diff
        _pm_1:  r(diff)
  permute var:  gruppe
```

						Monte Carlo error	
T	T(obs)	Test	c	n	p	SE(p)	[95% CI(p)]
_pm_1	-20	lower	514	5000	.1028	.0043	.0945 .1116
		upper	4753	5000	.9506	.0031	.9442 .9564
		two-sided			.2056	.0057	.1944 .2168

```
Note: For lower one-sided test, c = #{T <= T(obs)} and p = p_lower = c/n.
Note: For upper one-sided test, c = #{T >= T(obs)} and p = p_upper = c/n.
Note: For two-sided test, p = 2*min(p_lower, p_upper); SE and CI approximate.
```

Der Befehl ist wie folgt:

permute gruppe r(diff), reps(5000) seed(123) nodots: diff

Dabei ist *permute* der Befehl für den Permutationstests. Es folgt die zu randomisierende Gruppenzugehörigkeit, in diesem Fall die Variable *gruppe*. Danach wird der Skalar gesetzt, in dem die Differenz gespeichert wird. Wir haben diesen oben im Programm *diff* genannt, also muss *r(diff)* spezifiziert werden. Nach dem Komma folgen die Optionen des Befehls. Wir legen fest, dass 5,000 zufällige Permutationen gezogen werden sollen. Über *seed* wird der Zufallsgenerator in Stata auf

einen bestimmten Startwert gebracht, was bedeutet, dass wir reproduzierbar das exakt gleiche Ergebnis erhalten, wenn wir den Befehl später erneut ausführen. Dies ist für alle wissenschaftlichen Anwendungen und Reproduzierbarkeit essentiell. Danach geben wir an, dass wir keine Punkte angezeigt bekommen wollen mit *nodots*, sodass unser Display nicht unnötig gefüllt wird. Nach dem Doppelpunkt erfolgt das Programm bzw. der Befehl, der ausgeführt werden soll. Da unser Programm *diff* heißt, geben wir diesen Namen an.

Nun zum Output. Es folgt eine Warnung, die den Nutzer darauf hinweist, dass nicht geprüft wird, ob fehlende Werte vorliegen. Wäre dies der Fall, so wären diese ebenfalls randomisiert und das Ergebnis vermutlich falsch. Deshalb müssen wir sicherstellen, dass keine fehlenden Werte vorliegen, was wir haben. Die interessanten Ergebnisse finden sich in der Spalte p, also die p-Werte. Wie wir sehen wird unser „perfekter“ Wert von 0,10 approximiert (.1028). Der zweiseitige Wert ist immer das doppelte des einseitigen Wertes. Wir sehen, dass 514 von 5,000 Permutationen einen gleich großen oder noch extremeren Wert haben als der faktische Wert, der unter *T(obs)* angeführt wird. Zudem bekommen wir eine Schätzung über den Monte Carlo Fehler. Dieser rührt daher, dass wir zufällige Stichproben ziehen und daher der Zufall einen Einfluss hat. Ist dieser Wert zu groß, so müssen wir mehr Replikationen wählen, also etwa 10,000. Die Rechenzeit für das aktuelle Beispiel ist aufgrund der Gruppengröße extrem klein.

Es stellen sich im Anschluss einige Folgefragen:

- Grundsätzlich sollte die Anzahl der Replikationen so groß wie möglich gewählt werden. Je nachdem wie groß der Datensatz ist und wie lange es dauert, die Statistik zu berechnen, kann man hier Modifikationen vornehmen. Ist die Rechenzeit kurz, so können auch große Zahlen wie 50,000 oder gar 500,000 Replikationen sinnvoll sein. Mehr ist immer besser. Für anfängliche Tests können auch kleinere Werte gewählt werden. Letztlich ist zu bedenken, dass es nach Monaten der Arbeit an einem Experiment wohl kaum ein Problem darstellt, wenn die abschließende Berechnung des p-Wertes einige Stunden in Anspruch nimmt.
- Ob ein einseitiger oder zweiseitiger Wert genutzt werden soll, ist idealerweise *vor* Durchführung des Experiments zu klären. Teste ich beispielsweise ein neues Medikament, so habe ich sicher eine Annahmen über die Wirkung („das Medikament sollte den Blutdruck *reduzieren*“). Bei gerichteten Hypothesen ist daher der einseitige Test zu wählen. Ist hingegen meine Erwartung ohne Richtung („...nach Behandlung werden sich die Gruppenmittelwerte unterscheiden“), so ist der zweiseitige Wert zu benutzen. Ehrlicherweise sollte man immer den zweiseitigen Test benutzen, wenn man die Ergebnisse bereits gesehen hat, also weiß, in welche Richtung der Effekt ausschlägt. Ein zweites Beispiel findet sich im Appendix, in dem untersucht wird, ob sich das Medianeinkommen je nach Heiratsstatus unterscheidet. Da wir keine Annahme über die Richtung des Effekts haben nutzen wir den zweiseitigen Wert. Es zeigt sich, dass das Ergebnis nicht signifikant ist ($p = 0.2880$).

4 Abhängige Stichproben

Bisher wurde der Permutationstest als Alternative zum Zweistichproben-t-Test herangezogen. Doch was ist, wenn die Stichproben abhängig sind, also Korrelationen zu erwarten sind? Ein häufiges Beispiel sind Pre-Post-Vergleiche. Der Wert eines Probanden wird einmal vor und einmal nach Applizierung des Treatments gemessen. Offenbar besteht hier eine Abhängigkeitsstruktur und wir müssen annehmen, dass Pre- und Postmessung miteinander korrelieren. In diesen Fällen ist der reguläre t-Test und auch der reguläre Permutationstest nicht valide. Eine Lösung ist der Permutationstest für abhängige Stichproben. Das Vorgehen ist dabei noch einfacher. Es wird für jede Person bzw. jeden Fall das Differenzmaß berechnet (Postwert - Prewert). Nun wird das faktische Ergebnis für diese Variable berechnet. Anschließend werden wiederholt *Vorzeichen* randomisiert zugewiesen. Dabei wird also simuliert, wie sich die Ergebnisse ändern würden, wenn Pre- und Postmessung vertauscht wären. Der Rest des Tests ist identisch. Dieser Test ist in Stata nicht direkt verfügbar, kann aber über ein Zusatzpaket installiert werden. Das Paket ist *permtest1* und wurde von Johannes Kaiser veröffentlicht. Zunächst muss man das Paket manuell suchen und installieren. Dazu den Befehl *findit permtest1* eingeben und dann *st0134* auswählen und installieren. Die Durchführung des Tests ist danach simpel.

```
. sysuse bpwide, clear
(fictional blood-pressure data)

. set seed 123

. sum bp_before bp_after
```

Variable	Obs	Mean	Std. Dev.	Min	Max
bp_before	120	156.45	11.38985	138	185
bp_after	120	151.3583	14.17762	125	185

```
. permtest1 bp_before = bp_after
Fisher-Pitman permutation test for paired replicates
```

difference vector	bp_before-bp_after
observations	120
- positive	73
- negative	43
- zero	4
critical value	611

```
mode of operation:  Montecarlo simulation (200000 runs)

Progress:  |....................................|

Test of hypothesis Ho: bp_before>=bp_after : p = .999445
Test of hypothesis Ho: bp_before<=bp_after : p = .00057
Test of hypothesis Ho: bp_before==bp_after : p = .00114
```

Bei kleiner Gruppengröße wird automatisch der exakte Test mit allen denkbaren Permutationen durchgeführt. In diesem Beispiel mit 120 Fällen wird ein Monte Carlo Test mit 200,000 Permutationen durchgeführt. Der p-Wert des zweiseitigen Tests liegt bei 0,00114. Für mehr Details *help permtest1* eingeben.

5 Literaturverzeichnis

- Bittmann, F. (2019). *Stata: A Really Short Introduction*. Walter de Gruyter GmbH & Co KG.
- Good, P. (2013). *Permutation tests: a practical guide to resampling methods for testing hypotheses*. Springer Science & Business Media.
- Hesterberg, T. C. (2015). What teachers should know about the bootstrap: Resampling in the undergraduate statistics curriculum. *The American Statistician*, *69*(4), 371-386. Siehe auch: http://arxiv.org/abs/1411.5279
- Phipson, B., & Smyth, G. K. (2010). Permutation P-values should never be zero: calculating exact P-values when permutations are randomly drawn. *Statistical applications in genetics and molecular biology*, *9*(1).
- Sutton, C. D. (1993). Computer-intensive methods for tests about the mean of an asymmetrical distribution. *Journal of the American Statistical Association*, *88*(423), 802-810.

6 Appendix

Stata-Code für alle Beispiele.

```
clear all
version 16.1
input y gruppe
      35 0
      55 0
      78 0
      68 1
      79 1
      81 1
end
bysort gruppe: sum y


cap program drop diff
program define diff, rclass
      quiet sum y if gruppe == 0, detail
      local m1 = r(mean)
      quiet sum y if gruppe == 1, detail
      local m2 = r(mean)
      return scalar diff = `m1' - `m2'
end

diff
return list

permute gruppe r(diff), reps(5000) seed(123) nodots: diff

*Zweites Beispiel*

sysuse nlsw88, clear
drop if missing(union, married)
cap program drop diff_median
program define diff_median, rclass
      quiet sum wage if married == 0, detail
      local m1 = r(p50)
      quiet sum wage if married == 1, detail
      local m2 = r(p50)
      return scalar diff = `m1' - `m2'
end
diff_median
return list

permute married r(diff), reps(7500) seed(123) dots(100): diff_median
```

```
*Permutationstest fuer abhaengige Stichproben
*Ado permtest1 muss installiert sein
sysuse bpwide, clear
set seed 123
sum bp_before bp_after
permtest1 bp_before = bp_after
```