Elisabeth Scholz-Wardzala

Automatisierung und Anwendung des Run-Tests

Diplomarbeit

FACHHOCHSCHULE GIESSEN-FRIEDBERG

Fachbereich Mathematik, Naturwissenschaften und Datenverarbeitung
Studiengang Mathematik
Studienschwerpunkt Mathematik und Wirtschaft

Referent:	Prof. Dr. Manfred Börgens
Korreferent:	Prof. Dieter Greiner

August 2001

Inhaltsverzeichnis

1 Vorwort

In dieser Diplomarbeit wird der Run-Test (oder Iterationstest) beschrieben, untersucht und mit Hilfe des Computer-Algebra-Programms automatisiert.

Zum einen wurden die wichtigsten Formeln des Run-Tests in Mathematica programmiert, wobei das Hauptprogramm zur exakten Auswertung des Run-Tests erstellt wurde (siehe Kapitel 5.7 und 7.1.9).
Zum anderen wurden verschiedene Programme zur Anwendung des Run-Tests ausgearbeitet (siehe Kapitel 5.4 - 5.7 und 7.1).

Ein Kernpunkt dieser Arbeit ist die exakte Auswertung der dem Run-Test zugrunde liegenden Wahrscheinlichkeitsverteilung. Es wird gezeigt, dass die übliche Näherung mit der Normalverteilung zu falschen Ergebnissen führen kann.

Wozu dient ein solcher Test?

Es gibt im täglichen Leben viele Probleme, die mit Hilfe des Run-Tests geprüft werden können. Ein Beispiel dafür wäre die Regelmäßigkeit bei den Präsidentschaftswahlen in den USA. Die Bürger wählen fast abwechselnd Republikaner und Demokraten, vielleicht weil sie nie zufrieden sind. Ist aber die Reihenfolge der Parteien wirklich nicht zufällig?

Oder wie zufällig sind die Schwankungen der Benzinpreise an den Tankstellen (siehe Kapitel 6.2)?

Um das Problem noch besser erläutern zu können, kann man an ein Kartenspiel denken. Die Karten werden gemischt. Beim ersten Mal erhält man zuerst nur schwarze und dann nur rote Karten, beim zweiten Mal immer abwechselnd rote Karten, schwarze Karten. Intuitiv ist es jedem klar, dass hier eine Manipulation vorliegt. Beim ersten Mal gibt es zu wenige, beim zweiten zu viele „Runs“ (das heißt Serien gleicher Karten), deswegen scheint die Reihenfolge der Karten nicht zufällig zu sein. Dies wird im Kapitel 6.1 untersucht.

Aber was heißt eigentlich zu viele und zu wenige Runs? Wo liegen hier die Grenzen, d.h. statistisch gesehen, die kritischen Werte des Tests? Oder mathematisch ausgedrückt: Mit welcher Irrtumswahrscheinlichkeit α kann man die Nullhypothese (die Reihe der Beobachtungen ist zufällig) ablehnen?

Im Rahmen dieser Arbeit werden Programme zum einseitigen und zweiseitigen Test erstellt, so dass die gestellten Fragen beantwortet werden können.

Der Run-Test kann uns beispielsweise helfen, die Zufälligkeit von automatisch generierten Zufallszahlen zu prüfen.
Es gibt viele Methoden der Generierung von Zufallszahlen auf einem Rechner und der Testung des Grades der Abweichung der aufgrund dieser Methoden erzielten Zahlen von der Zufälligkeit.

Zum Testen von Zufallszahlen werden vorwiegend Häufigkeitstests durchgeführt. Es wird zum Beispiel geprüft, ob es unter den Zufallszahlen gleich viele gerade und

ungerade Zahlen gibt. Mit dem Run-Test kann man zusätzlich *die Reihenfolge* des Auftretens der geraden und ungeraden Zahlen prüfen. Falls zuerst nur gerade und dann nur ungerade Zahlen kommen (das heißt ganz wenige Runs), oder abwechselnd gerade und ungerade Zahlen (ganz viele Runs), dann ist es klar, dass dieser Zufallsgenerator nicht gut arbeiten kann. Man kann außerdem die Reihenfolge anderer Merkmale überprüfen. Auf die Einzelheiten kommen wir in den nächsten Abschnitten zurück. Das Problem der Zufallszahlen wird ausführlich im Kapitel 6.4 behandelt.

2 Einleitung

2.1 Begriff des Runs

Ein „Run“ (Iteration) ist eine ununterbrochene Folge von einem oder mehreren identischen Symbolen.

Dieser Diplomarbeit liegt ein Spezialfall des Runs mit nur zwei Arten von Symbolen (Beobachtungen) zugrunde.

Von der Iteration mehrerer Arten von Beobachtungen kann man in [17] nachlesen.

Hierzu werden folgende Bezeichnungen benutzt:

Bezeichnung	Beschreibung
$\mathbf{r}_{ij}$	Anzahl von Runs des Objekts vom Typ *i* mit der Länge *j* (i=1,2,..)
$\mathbf{r}_{i}$	Anzahl von Runs des Objekts vom Typ *i* unabhängig von der Länge *j*
$\mathbf{n}_{i}$	Anzahl der Objekte vom Typ *i*
n	Gesamtanzahl der Objekte von beiden Typen, also in einer Folge von zwei verschiedenen Objekten ist, $n = n_1 + n_2$

Tabelle 2.1: Bezeichnungen zur Run-Theorie

Zur Bezeichnung der zwei Arten von Beobachtungen wird der Index 1 und 2 verwendet.

In einer Folge mit n Objekten von zwei Arten von Beobachtungen ist:

$$2 \leq r \leq n \quad \text{wobei} \quad r \in N.$$

In einer Folge mit n Objekten gibt es mindestens 2 und höchstens n Runs.

Daraus folgt, insgesamt gibt es $n-1$ Runs.

2.2 Beispiele zum besseren Verständnis des Runs

Beispiel 1

Die Anzahl der roten und schwarzen Karten ist jedesmal gleich, nur die Anzahl der Runs ändert sich.

Fall 1

Beim Kartenmischen entstand beim ersten Mal folgende Reihenfolge der roten und schwarzen Karten:

S S S R S R R R S S R S R S R R R S S R S S S R R S S S S R R R

R - rote Karten, Objekte von Typ 1
S - schwarze Karten, Objekte von Typ 2

$n_1 = 16,\ n_2 = 16,\ n_1 + n_2 = n = 32$
$r_1 = 8,\quad r_2 = 8,\quad r = 16$

Fall 2

Beim zweiten Kartenmischen sah die Folge folgendermaßen aus:

S S S S S S S S S S S S S S S R R R R R R R R R R R R R R R

$n_1 = 16,\ n_2 = 16,\ n_1 + n_2 = n = 32$
$r_1 = 1,\quad r_2 = 1,\quad r = 2$

Fall 3

Beim dritten Mal hatte die Kartenfolge folgendes Muster:

S R S R S R S R S R S R S R S R S R S R S R S R S R S R S R S R

$n_1 = 16,\ n_2 = 16,\quad n_1 + n_2 = n = 32$
$r_1 = 16,\ r_2 = 16,\quad r = 32$

Bei diesem Kartenmischen kamen immer abwechselnd rote Karten und schwarze Karten. Wir haben hier die maximale Anzahl von Runs bekommen, r = n.

Intuitiv ist jedem klar, dass nur im ersten Fall keine Manipulation vorliegt. Im zweiten Fall gibt es zu wenige, im dritten zu viele Runs, deswegen scheint die Reihenfolge der Karten nicht zufällig zu sein.

Dies werden wir anschließend mit dem zweiseitigen Run-Test untersuchen (siehe Kapitel 6.1.1).

Beispiel 2

Achtzehn Schüler einer Grundschule, davon $n_1 = 8$ Jungs und $n_2 = 10$ Mädchen, stellen sich in eine Schlange vor dem Getränkeautomaten an. Die Schlange hat folgendes Muster:

J J J M M M M J J M M M M M J J J M $\qquad$ n = 18, $\qquad$ r = 6

Hier liegen also 6 Runs vor. Ein Soziologe wünscht zu testen, ob die Gruppierung der Kinder zufällig bezüglich des Geschlechts ist, oder die Gruppierung geschlechtshomogen ist.

Auf diese Fragen wird uns später der einseitige Run-Test antworten (siehe Kapitel 6.1.2).

3 Grundlagen der Statistik

Die mathematischen Grundlagen für dieses Kapitel findet man in [1] und [2].

„Statistik ist eine Zusammenfassung von Methoden, die uns erlauben, vernünftige optimale Entscheidungen im Falle von Ungewissheit zu treffen“ ([2], Seite 26).
Die beschreibende (deskriptive) Statistik befasst sich mit der Untersuchung und Beschreibung möglichst der ganzen Grundgesamtheit. Die moderne induktive oder analytische (beurteilende) Statistik untersucht demgegenüber nur einen Teil, der für die Grundgesamtheit, deren Eigenschaften uns interessieren, charakteristisch oder repräsentativ sein soll.

Es wird von den Beobachtungen in einem Teil auf die Grundgesamtheit geschlossen, das heißt, man geht induktiv vor. Ausschlaggebend ist hierbei, dass der zu prüfende Teil der Grundgesamtheit - die Stichprobe - zufällig ausgewählt wird. Wir bezeichnen eine Stichprobenentnahme als zufällig, wenn jede mögliche Kombination von Stichprobenelementen der Grundgesamtheit dieselbe Chance der Entnahme besitzt. Zufallsstichproben sind wichtig, da nur sie Rückschlüsse auf die Grundgesamtheit zulassen. Totalerhebungen sind häufig kaum oder nur mit großem Kosten- und Zeitaufwand möglich!

„Folgende vier Stufen wissenschaftlicher Methodik können unterschieden werden:

1. Es werden Beobachtungen gemacht.
2. Abstraktionen wesentlicher Elemente als Basis einer Hypothese oder Theorie.
3. Entwicklung der Hypothese oder Theorie mit der Voraussage neuer Erkenntnisse und/oder neuer Ereignisse.
4. Neue Tatsachen werden zur Überprüfung der aus der Theorie heraus gemachten Voraussagen gesammelt: Beobachtungen II “ ([2], Seite 26).

Damit beginnt der gesamte Kreislauf noch einmal. Wird die Hypothese bestätigt, dann werden die Prüfungsbedingungen durch präzisere Fassung und Erweiterung der Voraussagen so lange verschärft, bis endlich irgendeine Abweichung gefunden wird, die eine Verbesserung der Theorie erforderlich macht. Ergeben sich Widersprüche zur Hypothese, so wird eine neue Hypothese formuliert, die mit der größeren Anzahl von beobachteten Daten übereinstimmt. Endgültige Wahrheit kennt die tatsachenbezogene Wissenschaft sowieso nicht. „Die Vergeblichkeit aller Versuche, eine bestimmte Hypothese zu widerlegen, wird unser Vertrauen in sie vergrößern, jedoch ein endgültiger Beweis, dass sie stets gilt, lässt sich nicht erbringen. Hypothesen können nur geprüft, nie aber bewiesen werden! Empirische Prüfungen sind Widerlegungsversuche.

In dem geschilderten Kreisprozess kann die Statistik auf allen Stufen eingreifen:

1. Bei der Auswahl der Beobachtungen (Stichprobentheorie)
2. Bei der Klassifizierung, Darstellung und Zusammenfassung der Beobachtungen (beschreibende Statistik)
3. Bei der Schätzung von Parametern (Schätztheorie)
4. Bei der Formulierung und Überprüfung der Hypothesen (Prüfverfahren)“ ([2],Seite 27).

Auf der beschreibenden Statistik aufbauend, spielt die beurteilende, induktive oder analytische Statistik die wesentliche Rolle. *„Sie ermöglicht den Schluss von der Stichprobe auf die zugehörige Grundgesamtheit (zum Beispiel die Schätzung des Wahlresultates anhand bekannter Einzelergebnisse ausgewählter Wahlkreise), auf allgemeine Gesetzmäßigkeiten, die über den Beobachtungsbereich hinaus gültig sind“ ([2], Seite 27).* In allen empirischen Wissenschaften erlaubt sie durch Gegenüberstellung empirischer Befunde mit Ergebnissen, die man aus wahrscheinlichkeitstheoretischen Modellen - Idealisierungen spezieller experimenteller Situationen - geleitet, die Beurteilung empirischer Daten und die Überprüfung wissenschaftlicher Theorien; wobei allerdings nur Wahrscheinlichkeitsaussagen möglich sind, die dann dem Praktiker essenzielle Informationen als Grundlage für seine Entscheidung bieten.

„Statistische Einheiten, Merkmale, Gesamtheiten

Statistische Einheiten:	Objekte, an denen interessierende Größen erfasst werden
Grundgesamtheit:	Menge aller für die Fragestellung relevanten statistischen Einheiten
Teilgesamtheit:	Teilmenge der Grundgesamtheit
Stichprobe:	tatsächlich untersuchte Teilmenge der Grundgesamtheit
Merkmal:	interessierende Größe, Variable
Merkmalsausprägung:	konkreter Wert des Merkmals für eine bestimmte statistische Einheit“ ([15], Seite 15).

3.1 Elemente der Wahrscheinlichkeitsrechnung

3.1.1 Statistische Wahrscheinlichkeit

Die Grundlage des Wahrscheinlichkeitsbegriffes ist das bekannte Verhältnis

Anzahl der günstigen Fälle / Anzahl der möglichen Fälle

- die Definition der Wahrscheinlichkeit von Jakob Bernoulli (1654-1705) und de Laplace (1794-1827). Hierbei wird vorausgesetzt, dass alle möglichen Fälle gleichwahrscheinlich sind. Jede Wahrscheinlichkeit ist damit eine Zahl zwischen Null und Eins:

Formel 3.1.1 $0 \leq P \leq 1.$

Ein unmögliches Ereignis hat die Wahrscheinlichkeit Null, ein sicheres Ereignis die Wahrscheinlichkeit Eins. Im täglichen Leben wird diese Wahrscheinlichkeit mit 100 multipliziert in Prozenten ausgedrückt ($0\% \leq P \leq 100\%$).

Die meisten bei praktischen Problemen vorkommenden Zufallsvariablen lassen sich in zwei Klassen einteilen, nämlich in die sogenannten diskreten und stetigen Variablen.

Definition der Zufallsvariable

Eine Zufallsvariable X ist eine Abbildung, die jedem Element aus der Ergebnismenge eine reelle Zahl x zuordnet.

Definition der Wahrscheinlichkeitsverteilung

„Die Wahrscheinlichkeitsverteilung oder kurz Verteilung von X ist die Zuordnung von Wahrscheinlichkeiten $P(X \in I)$ oder $P(X \in B)$ für Intervalle oder zulässige Bereiche“ ([15], Seite 322).

3.1.2 Diskrete Verteilung

Definition der diskreten Verteilung

Eine Zufallsvariable X und ihre Verteilung heißen *diskret,* wenn folgende Bedingungen erfüllt sind:

- die Variable X nimmt nur endlich viele oder abzählbar unendlich viele reelle Werte mit positiver Wahrscheinlichkeit an

- in jedem endlichen Intervall der reellen Zahlengeraden liegen nur endlich viele der genannten Werte. Für jedes Intervall $a < X \leq b$, das keinen solchen Wert erhält, ist die zugehörige Wahrscheinlichkeit $P(a < X \leq b)$ gleich null.

Es gilt::

Formel 3.1.2 $P(X>c) = 1-P(X\leq c)$,

Formel 3.1.3 $P(a<X\leq b) = \sum_{j:a<x_j\leq b} p_j = 1$ für diskrete Zufallsvariable.

Es ist $P(X = x_1) = p_1$ usw., dann gilt für die diskrete Funktion

Formel 3.1.4 $f(x) = P(X=x) = \begin{cases} p_j & \text{für } x = x_j \\ 0 & \text{für alle übrigen } x \end{cases}$, wobei j=1,2,... .

Diese heißt ***die Wahrscheinlichkeitsfunktion*** der betreffenden Zufallsvariablen X. Sie ist Zusammenstellung der Wahrscheinlichkeiten für die einzelnen Realisationen der diskreten Zufallsvariable.

Man kann diese Funktionen mit Balkendiagrammen und Treppenfunktionen graphisch darstellen.

Die Verteilungsfunktionen geben Wahrscheinlichkeiten für Realisationen einer Zufallsvariable im Bereich von $-\infty$ bis zu einer gewissen oberen Grenze x an:

Formel 3.1.5 $F(x) = P(X\leq x)$.

Die Verteilungsfunktion einer diskreten Zufallsvariablen ist demnach eine Treppenfunktion, die Sprünge in jedem x_j besitzt, wobei die Sprunghöhe $f(x_j)=p_j$ beträgt. Der stufenförmige Verlauf der Verteilungsfunktion ist typisch für eine diskrete Zufallsvariable.

3.1.3 Stetige Verteilung

Definition einer stetigen Verteilung:

Eine Zufallsvariable X und deren Verteilung heißen *stetig,* wenn die zugehörige Verteilungsfunktion in Integralform dargestellt werden kann.

Der Integrand f heißt ***die Wahrscheinlichkeitsdichte*** oder Dichte der gegebenen Verteilung.

Formel 3.1.6 $F'(x) = f(x)$, $f(x)\geq 0$ für alle $x \in R$.

Die Verteilungsfunktion einer stetigen Zufallsvariable besitzt folgende Gestalt:

Formel 3.1.7 $F(x) = \int_{-\infty}^{x} f(t)dt$, für jedes $x \in R$, f ist die Wahrscheinlichkeitsdichte.

Formel 3.1.8 $\int_{-\infty}^{\infty} f(x)\,dx = 1.$

Außerdem gilt:

Formel 3.1.9 $P(a < X \leq b) = F(b) - F(a) = \int_{a}^{b} f(t)\,dt.$

Bei einer graphischen Darstellung verläuft die Dichtefunktion stets oberhalb oder auf der reellen Zahlengerade (x-Achse), wobei die zwischen x-Achse und Dichtefunktion liegende Fläche den Wert 1 besitzt.

3.1.4 Erwartungswert

Der Erwartungswert einer Verteilung wird mit μ bezeichnet. Er ist im Falle einer diskreten Verteilung durch

Formel 3.1.10 $\mu = \sum_{j} x_j f(x_j)$, falls die Summe konvergiert

definiert und im Falle einer stetigen Verteilung durch

Formel 3.1.11 $\mu = \int_{-\infty}^{\infty} x f(x)\,dx$, falls die Integrale konvergieren .

3.1.6 Varianz

Die Varianz einer Verteilung wird mit σ^2 bezeichnet. Sie ist im diskreten Fall durch

Formel 3.1.12 $\sigma^2 = \sum_{j} (x_j - \mu)^2 f(x_j)$, falls die Summe konvergiert,

und im stetigen Fall durch

Formel 3.1.13 $\sigma^2 = \int_{-\infty}^{\infty} (x - \mu)^2 f(x)dx$**,** falls die Integrale konvergieren,

definiert.

Die positive Quadratwurzel der Varianz heißt die Standardabweichung σ .

3.2 Binomialverteilung

3.2.1 Einführung

Die Bernoulli- oder Binomialverteilung ist eine der diskreten Verteilungen, die für die Statistik besonders wichtig sind. Die Binomialverteilung erhält man, wenn man sich für die Anzahl des Eintreffens eines bestimmten Ereignisses A bei der wiederholten Ausführung eines Zufallsexperimentes interessiert, bei dem A bei jeder Ausführung dieselbe Wahrscheinlichkeit *p* besitzt und sich die Ergebnisse der verschiedenen Ausführungen gegenseitig nicht beeinflussen.
Wird das genannte Experiment sehr oft ausgeführt und ist *p* klein, so lässt sich die Binomialverteilung durch die in vieler Hinsicht bequemere Poisson-Verteilung annähern.
Die Binomialverteilung ist auch beim Ziehen mit Zurücklegen von Bedeutung. Die entsprechende Rolle spielt beim Ziehen ohne Zurücklegen die etwas kompliziertere hypergeometrische Verteilung.

3.2.2 Beschreibung

„Hat ein Ereignis A bei einem Zufallsexperiment die Wahrscheinlichkeit

p,

so ist die Wahrscheinlichkeit, dass A nicht eintrifft, gleich

q = 1 - p.

Wir betrachten nun die Zufallsvariable

X = Anzahl des Eintreffens von A.

Wird das Experiment nur einmal ausgeführt, so kann X natürlich nur die Werte 0 oder 1 annehmen je nachdem, ob A nicht eintrifft oder eintrifft.
Die zugehörigen Wahrscheinlichkeiten sind

$P(X=0) = q$ und $P(X=1) = p$.

Also hat die zu X gehörige Wahrscheinlichkeitsfunktion f(x) die Werte

$p(0) = q, \quad f(1) = p$

und f(x)=0 für alle übrigen x. Wie man leicht nachprüft, lassen sich die beiden Formeln zu einer einzigen Formel zusammenfassen:

Formel 3.2.1 $f(x) = p^x q^{n-x}$ $(x = 0,1, ..., n)$“ ([1], Seite 107-108).

Wird ein Zufallsexperiment mehrmals ausgeführt, etwa n-mal hintereinander, so kann X die Werte 0,1,...,n annehmen. Wir fragen nach den zugehörigen Wahrscheinlichkeiten unter der Voraussetzung, dass die Ergebnisse der einzelnen

Ausführungen voneinander unabhängig sind, d.h. sich gegenseitig nicht beeinflussen.

Das Ereignis

X = x („A trifft bei den n Ausführungen genau x-mal ein“)

ist realisiert, wenn wir (in der Reihenfolge des Auftretens) die Einzelereignisse

A A ... A B B ... B (B= $\bar{A}$, d.h. A trifft nicht ein)
x mal n-x mal

erhalten. Diese Einzelereignisse sind unabhängig. Also hat unsere Form der Realisierung des Ereignisses X = x die Wahrscheinlichkeit

$$\underset{x\text{ mal}}{p\ p\ \ldots\ p}\ \underset{n-x\text{ mal}}{q\ q\ \ldots\ q} = p^x q^{n-x}.$$

Jede andere Form der Realisierung des Ereignisses X = x ergibt sich aus der betrachteten Form durch Änderung der Reihenfolge der Einzelereignisse.

Da sich diese Formen gegenseitig ausschließen folgt aus dem Satz:

$$P(A) = 1 - P(\bar{A}),$$

dass die Wahrscheinlichkeit des Ereignisses X = x gleich der Summe der Wahrscheinlichkeiten aller möglichen Formen ist, also gleich $p^x q^{1-x}$ mal der Anzahl dieser Formen. Diese Anzahl ergibt sich so:
Wir nummerieren die n Ausführungen von 1 bis n durch, x dieser n Zahlen greifen wir heraus. Das sind die Nummern der Ausführungen, bei denen A eintrifft.
Auf die Reihenfolge der herausgegriffenen Zahlen kommt es nicht an. Also haben wir es mit Kombinationen x-ter Ordnung von n verschiedenen Elementen (den n Nummern der Ausführungen) ohne Berücksichtigung der Anordnung zu tun. Gemäß dem Satz für die Kombination ohne Wiederholung und ohne Berücksichtigung der Anordnung ist deren Anzahl gleich

Formel 3.2.2 $$\binom{n}{x} = \frac{n!}{x!(n-x)!}.$$

Damit wird die Wahrscheinlichkeit P(X = x) des Ereignisses X = x gleich

Formel 3.2.3 $$f(x) = \binom{n}{x} p^x q^{n-x} \qquad (x=0,1,\ldots,n),$$

$$(f(x) = 0 \qquad \text{(für alle übrigen x-Werte)}).$$

Dies ist also die Wahrscheinlichkeit, dass ein Ereignis A bei n unabhängigen Ausführungen eines Experimentes genau x-mal eintrifft, wenn A bei der Einzelausführung die Wahrscheinlichkeit p besitzt und q = 1 - p ist.
Die durch die oben genannte Wahrscheinlichkeitsfunktion bestimmte Verteilung heißt die Bernoulli- oder Binomialverteilung. Sie ist die wichtigste diskrete Verteilung. Das Eintreffen des genannten Ereignisses A wird gewöhnlich als Erfolg bezeichnet und das Nichteintreffen als Misserfolg. Die Zahl p heißt die *Erfolgswahrscheinlichkeit beim Einzelversuch.* Ein solches Experiment, bei dem nur zwei verschiedene sich gegenseitig ausschließende Ereignisse eintreffen können, heißt ein Bernoulli-Experiment, nach Jakob Bernoulli, der sich zuerst mit solchen Experimenten befasst hat.

Aus der Definition der Binomialkoeffizienten folgt unmittelbar

Formel 3.2.4 $$\binom{n}{x+1} = \frac{n-x}{x+1}\binom{n}{x}.$$

Damit erhalten wir die nützliche Rekursionsformel

Formel 3.2.5 $$f(x+1)= \left(\frac{n-x}{x+1}\right)\frac{p}{q}\, f(x) \quad (x = 0,1,...,n-1).$$

Aus 3.2.3 ergibt sich die Verteilungsfunktion

Formel 3.2.6 $$F(x) = \sum_{k\le x}\binom{n}{k} p^k q^{n-k} \text{ für } x \ge 0,$$

wobei über alle nichtnegativen ganzzahligen k zu summieren ist, die höchstens gleich *x* sind.

Die Binomialverteilung hat den *Erwartungswert* $\mu = np$, und die *Varianz* $\sigma^2 = npq$.

In einer Kiste sind N Kugeln, darunter M schwarze. Die Wahrscheinlichkeit, beim zufälligen Herausgreifen einer Kugel eine schwarze zu erhalten ist gleich p = M/N.

Also ist *die Wahrscheinlichkeit bei n Zügen mit Zurücklegen* genau x schwarze Kugel zu erhalten, gemäß Formel 3.2.3 gleich

Formel 3.2.7 $$f(x)=\binom{n}{x}\left(\frac{M}{N}\right)^x\left(1-\frac{M}{N}\right)^{n-x}.$$

3.3 Hypergeometrische Verteilung

3.3.1 Einführung

Der Name *hypergeometrisch* rührt daher, dass sich die zugehörige momenterzeugende Funktion durch die hypergeometrische Funktion ausdrücken lässt. Diese Verteilung spielt beim Ziehen ohne Zurücklegen dieselbe Rolle wie die Binomialverteilung beim Ziehen mit Zurücklegen.

3.3.2 Beschreibung

Die Verteilung mit der Wahrscheinlichkeitsfunktion

Formel 3.3.1
$$f(x) = \frac{\binom{M}{x}\binom{N-M}{n-x}}{\binom{N}{n}} \qquad \text{für } x = 0,1,...,n$$

heißt die **hypergeometrische Verteilung**.

Für alle übrigen Werte von x ist $f(x) = 0$.

Erwartungswert und Varianz

Die hypergeometrische Verteilung hat den *Erwartungswert*

Formel 3.3.2
$$\mu = n\frac{M}{N}$$

und die *Varianz*

Formel 3.3.3
$$\sigma^2 = \frac{nM(N-M)(N-n)}{N^2(N-1)}.$$

3.3.3 Vergleich der hypergeometrischen Verteilung und der Binomialverteilung

Für die Wahrscheinlichkeitsfunktion der hypergeometrischen Verteilung gilt folgendes:

Formel 3.3.4 $$\binom{n}{x}\left(p-\frac{x}{N}\right)^{x}\left(q-\frac{n-x}{N}\right)^{n-x} < f(x) < \binom{n}{x}p^{x}q^{n-x}\left(1-\frac{n}{N}\right)^{-n}$$

Hierbei ist $p = \frac{M}{N}$ und $q = 1-p$.

Halten wir n und x fest und lassen N und M derart gegen unendlich streben, dass p = M/N einem Wert zwischen 0 und 1 zustrebt, so sehen wir, dass der erste und der letzte Ausdruck in 3.3.4 gegen die Wahrscheinlichkeitsfunktion 3.2.7 der Binomialverteilung streben. Dies bedeutet praktisch:

Für große N,M und N-M und im Vergleich dazu kleine n kann die hypergeometrische Verteilung durch die Binomialverteilung mit dem durch p = M/N gegebenen p brauchbar angenähert werden.

3.4 Normalverteilung

3.4.1 Einführung

Die GAUSS- Verteilung oder Normalverteilung wurde von Gauss im Zusammenhang mit der Theorie der Messfehler eingeführt. *Sie ist die wichtigste stetige Verteilung*.

Hierfür gibt es verschiedene Gründe:

- Viele Zufallsvariablen, die bei Experimenten und Beobachtungen in der Praxis auftreten, sind normalverteilt.
- Gewisse nicht normalverteilte Variablen lassen sich auf verhältnismäßig einfache Weise derart transformieren, dass die sich ergebende Variable normalverteilt ist.
- Bei statistischen Prüfverfahren und deren Begründung kommen oft Größen vor, deren Verteilungen normal sind oder wenigstens bei gewissen Grenzübergängen einer Normalverteilung zustreben.
- Um diese Verteilung zu würdigen, ist auf jedem 10 DM-Schein neben dem Bild von Carl Friedrich Gauß, dem Entdecker der Normalverteilung, die Dichtefunktion mit Formel abgebildet.
- *Die zentrale Bedeutung der Normalverteilung besteht darin, dass eine Summe von vielen unabhängigen, beliebig verteilten Zufallsvariablen gleicher Größenordnung angenähert normalverteilt ist, und zwar um so besser angenähert, je größer ihre Anzahl ist (Zentraler Grenzwertsatz).*

Die Verteilungsfunktion der Normalverteilung ist ein Integral, das sich nicht mehr elementar auswerten lässt. Numerische Werte kann man aber der Tafel entnehmen.

3.4.2 Beschreibung

Die **Wahrscheinlichkeitsdichte** der Normalverteilung ist durch Formel 3.4.1 gegeben:

Formel 3.4.1
$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \quad \text{für} \quad (-\infty < x < +\infty,\ \sigma > 0).$$

Eine Zufallsvariable mit dieser Verteilung heißt kurz eine **normalverteilte Zufallsvariable**.

Die Kurve von f(x), die sogenannte **Glockenkurve**, ist symmetrisch bezüglich μ, sie fällt nach beiden Seiten von μ glockenförmig ab, dabei nähert sie sich der x-Achse

asymptotisch, d.h. sie berührt die x-Achse erst im Unendlichen. Der Abstand der beiden Wendepunkte zum Erwartungswert μ wird als Standardabweichung σ bezeichnet.

Die Normalverteilung ist vollständig durch die Parameter μ und σ charakterisiert. Der Erwartungswert μ bestimmt die Lage der Verteilung im Hinblick auf die x-Achse, die Standardabweichung σ die Form der Kurve. Je größer σ ist, um so flacher ist der Kurvenverlauf, um so breiter ist die Kurve und um so niedriger liegt das Maximum, das an der Stelle des Erwartungswertes $E(x) = \mu$ liegt.

Abbildung 3.1 zeigt f(x) für $\mu = 3$, $\sigma = 1$. Für $\mu > 3$ bzw. $\mu < 3$ erhält man Kurven derselben Form, die lediglich um den Betrag $|\mu|$ nach rechts bzw. links verschoben sind.

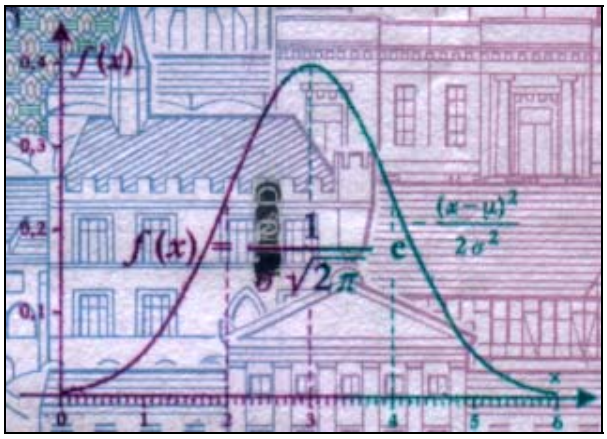

Abbildung 3.1: Dichte der Normalverteilung [3,1] (Darstellung aus dem 10 DM-Schein)

Verteilungsfunktion der Normalverteilung (kumulierte Dichtefunktion)

Formel 3.4.4

$$N_{\mu,\sigma^2}(x) = F(x) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{1}{2}\left(\frac{v-\mu}{\sigma}\right)^2} dv$$

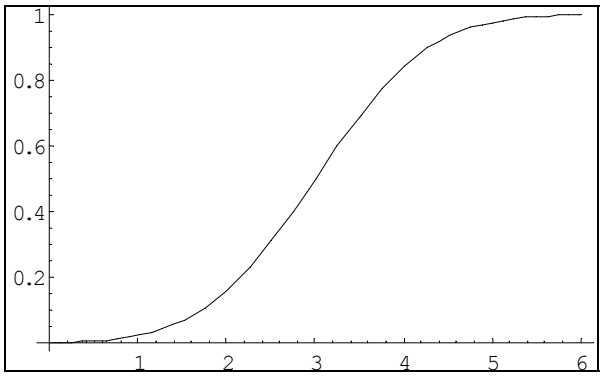

Abbildung 3.2: Verteilungsfunktion der Normalverteilung [3,1]

Bei der standardisierten Normalverteilung ist $\mu = 0$ und $\sigma^2 = 1$.

$\Phi(z)$ ist die Verteilungsfunktion dieser Standardnormalverteilung.
Eine grundlegende Beziehung finden wir in der

Formel 3.4.6 $F(x) = \phi\left(\frac{x-\mu}{\sigma}\right)$, $z = \frac{x-\mu}{\sigma}$

Daraus folgt weiterhin

$$P(a < X \leq b) = F(b) - F(a) = \phi\left(\frac{b-\mu}{\sigma}\right) - \phi\left(\frac{a-\mu}{\sigma}\right).$$

Damit kann man Wahrscheinlichkeiten normalverteilter Zufallsvariablen unter Benutzung der Tafel für die Normalverteilung berechnen.

Bei der Normalverteilung sind praktisch alle Werte zwischen den **„3 σ - Grenzen“** $\mu - 3\sigma$ und $\mu + 3\sigma$ zu erwarten.

Formel 3.4.7

$P(\mu - \sigma < X \leq \mu + \sigma) \approx 68\ \%$
$P(\mu - 2\sigma < X \leq \mu + 2\sigma) \approx 95{,}5\ \%$ (siehe die Abbildung 3.3)
$P(\mu - 3\sigma < X \leq \mu + 3\sigma) \approx 99{,}7\ \%$

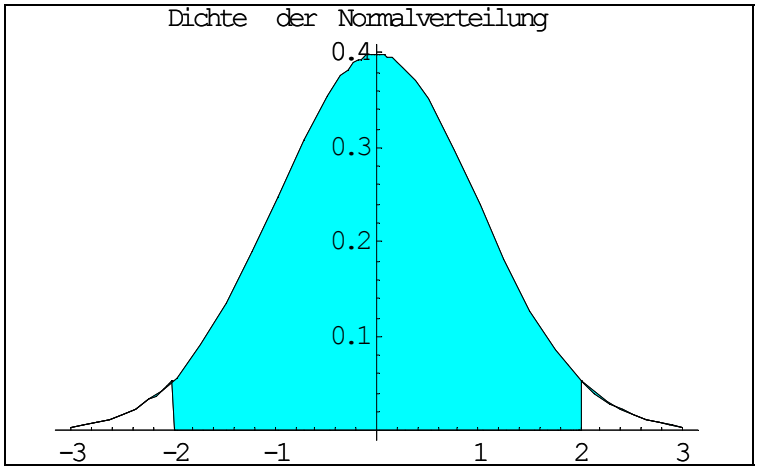

Abbildung 3.3: Dichte der Normalverteilung [0,1], markierte Fläche 95,5 %.

Eine wichtige Formel für die statistische Praxis ist:

Formel 3.4.9

$P(\mu - 1{,}96\sigma < X \leq \mu + 1{,}96\sigma) \approx 95\ \%$
$P(\mu - 2{,}58\sigma < X \leq \mu + 2{,}58\sigma) \approx 99\ \%$
$P(\mu - 3{,}29\sigma < X \leq \mu + 3{,}29\sigma) \approx 99{,}9\ \%$.

3.5 Hypothesen, Fehler beim Testen, Irrtumswahrscheinlichkeit

3.5.1 Hypothesen

Unter einer Hypothese versteht man in der Statistik eine Annahme über die Verteilung einer Zufallsvariablen, zum Beispiel die Annahme, dass die betreffende Verteilung einen bestimmten Erwartungswert hat, usw. Und ein Test einer Hypothese ist ein Prüfverfahren, das man anwendet, wenn man wissen will, ob man die Hypothese „annehmen“ kann oder „verwerfen“ soll, das heißt ob man der Hypothese Vertrauen schenken und sie für richtig halten kann oder nicht.
Mit anderen Worten, die Aufgabe eines statistischen Tests ist es, Hypothesen über die Verteilung einer Zufallsvariablen X anhand einer Stichprobe $x_1, \ldots, x_n$ zu überprüfen.

Zur Überprüfung der Hypothesen dient eine *Teststatistik* $T=T(X_1, \ldots, X_n)$. Je nach Realisation von T entscheiden wir uns für oder gegen die vorliegenden Hypothesen.

Das Testproblem wird in Form einer *Nullhypothese* H_0 und einer Gegenhypothese (*Alternativhypothese*) H_1 formuliert. Nach bestimmten Kriterien wird eine Menge C gewählt, die Entscheidung zugunsten einer der beiden Hypothesen vorschreibt. Die Nullhypothese wird in Abhängigkeit von der Alternativhypothese formuliert. „Die Nullhypothese ist eine Negativhypothese, mit der behauptet wird, dass die zur Alternativhypothese komplementäre Aussage richtig sei“ ([18], Seite 109).

Entscheidung für H_0, falls T sich nicht in C realisiert, Entscheidung für H_1, falls T sich in C realisiert.

C wird *kritisches Gebiet* oder *kritischer Bereich* genannt. Meistens ist C eine Teilmenge der reellen Zahlen.

Woher stammen die Hypothesen? Typische Fälle sind:

1. Die Hypothese entspricht einer Güteforderung, die man erfüllen soll (Sollwert).
2. Man kennt die jeweils interessierenden Werte der Grundgesamtheit aus langer Erfahrung.
3. Die Hypothese ergibt sich aus einer Theorie, die man verifizieren möchte.
4. Die Hypothese ist eine reine Vermutung, angeregt durch Wünsche oder gelegentliche Wahrnehmungen.

Beim Testen von Hypothesen schließt man von Stichproben auf Grundgesamtheiten. Wir wissen, dass es dabei keine vollkommen sicheren Schlüsse gibt. Also muss es auch beim Testen von Hypothesen Fehlerrisiken geben.

3.5.2 Fehler beim Testen

Der Anwender muss vor der Durchführung eines Tests eine kleine Irrtumswahrscheinlichkeit α (das sogenannte Signifikanzniveau) festlegen, also die Wahrscheinlichkeit, die er für die Fehlentscheidung zulassen will, dass H_0 zu Unrecht abgelehnt wird. Je kleiner er α wählt, desto unwahrscheinlicher wird zwar diese Fehlentscheidung, desto geringer wird jedoch auch die Chance, bei falscher H_0 zu einer Ablehnung zu kommen. Gebräuchlich sind für α vor allem Werte wie 0,1; 0,05; 0,01.

Die Entscheidung für H_0 oder H_1 kann falsch sein. Man unterscheidet zwischen:

Fehler 1. Art: Entscheidung für H_1, d.h. H_0 ablehnen (wenn $T \in C$), obwohl H_0 zutrifft.
Fehler 2. Art: Entscheidung für H_0, d.h. H_0 annehmen (wenn $T \notin C$), obwohl H_1 zutrifft.

3.5.3 Irrtumswahrscheinlichkeit im zweiseitigen und einseitigen Test

Erklärung am Beispiel der Normalverteilung:

Es liege eine $N(\mu,\sigma^2)$-verteilte Grundgesamtheit G mit bekannter σ^2 vor.
$X_1, \ldots, X_n$ seien die Variablen einer einfachen Stichprobe aus G. Über den unbekannten Erwartungswert μ von G bestehe die Hypothese H_0, dass μ gleich einem gegebenen Wert μ_0 ist.

Man betrachte folgende Testprobleme:

(1): $H_0: \mu = \mu_0$ gegen $H_1: \mu < \mu_0$
(2): $H_0: \mu = \mu_0$ gegen $H_1: \mu > \mu_0$
(3): $H_0: \mu = \mu_0$ gegen $H_1: \mu \neq \mu_0$.

Für $\mu = \mu_0$ ist $Z = \frac{\overline{X} - \mu_0}{\sigma}\sqrt{n}$.

Basierend auf dieser Prüfgröße Z fällt die Entscheidung für H_1 im Testproblem
(1), falls $z < -z_{1-\alpha}$
(2), falls $z > z_{1-\alpha}$
(3), falls $|z| > z_{1-\frac{\alpha}{2}}$, wobei $(1-\frac{\alpha}{2})$ bzw. $(1-\alpha)$-Fraktilen der N(0;1)-Verteilung sind.

Die folgenden Abbildungen demonstrieren diese drei Fälle. Die markierten Bereiche veranschaulichen, wann H_0 abgelehnt wird.

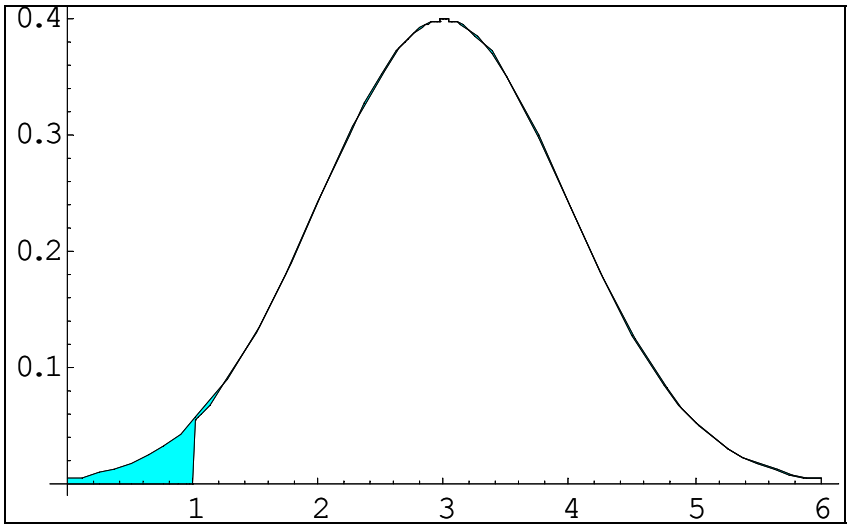

Abbildung 3.4: Dichtefunktion, Ablehnungsbereich im linksseitigen Test

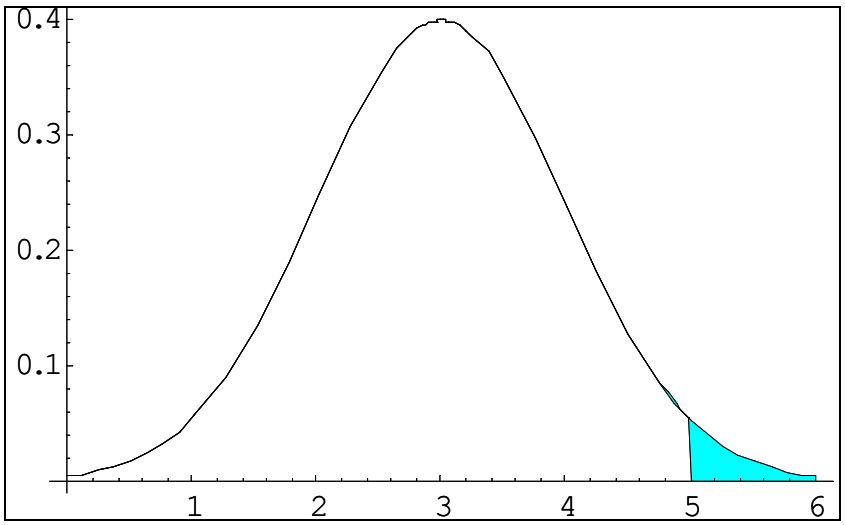

Abbildung 3.5: Dichtefunktion, Ablehnungsbereich im rechtsseitigen Test

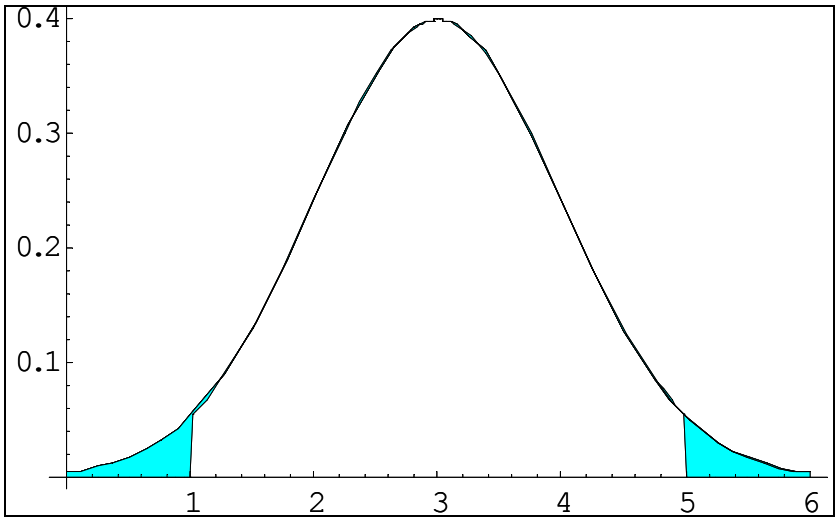

Abbildung 3.6: Dichtefunktion, Ablehnungsbereich im zweiseitigen Test

3.6 Parametrische und nichtparametrische Methoden

„Zwischen den Begriffen *nichtparametrisch* und *parametrisch* wird in den meisten Publikationen nicht streng unterschieden; oft werden diese Begriffe sogar gleichgesetzt und gegeneinander austauschbar verwendet. Grundsätzlich kann folgende Unterscheidung vorgenommen werden: Ein „verteilungsfreies“ Verfahren in der Schätz- und Testtheorie basiert auf einer Statistik, deren Verteilung nicht von der speziellen Gestalt der Verteilungsfunktion der Grundgesamtheit abhängt, aus der die Stichprobe gezogen wurde“ ([16], Seite 13).
Der Ausdruck *nichtparametrisch* bezieht sich auf Verfahren, die keine Aussagen über einzelne Parameter der Grundgesamtheitsverteilung machen.

Vorteile nichtparametrischer Methoden:

- Nichtparametrische Verfahren erfordern keine spezielle Verteilungsannahme für die Grundgesamtheit und kein umfangreiches Messniveau der Daten, wie es bei parametrischen Verfahren der Fall ist. Dies ist der Hauptvorteil verteilungsfreier Verfahren.

- Diese Methoden sind häufig effizienter als parametrische, wenn eine andere Verteilung als diejenige postuliert wird, unter die parametrischen Tests optimal sind. Selbst bei Annahme dieser Verteilung ist der Effizienzverlust nichtparametrischer Verfahren meist gering.

- Die zur Anwendung erforderlichen (schwachen) Annahmen sind in der Regel erfüllt.

- Nichtparametrische Techniken sind leicht anzuwenden und erfordern nur einen geringen Rechenaufwand und keine „spitzfindige“ Mathematik. Dadurch sind sie auch leicht zu erlernen.

- Die Teststatistik ist vorwiegend diskret. Die Herleitung ihrer Verteilung gelingt oft mit Hilfe einfacher kombinatorischer Überlegungen bzw. durch „Auszählen“.

- Die asymptotischen Verteilungen der Teststatistiken werden in der Regel durch die Normalverteilung und die χ^2-Verteilung erfasst.

Nachteile nichtparametrischer Methoden:

- Falls alle Annahmen des parametrischen statistischen Modells erfüllt sind, dann ist die Effizienz des nichtparametrischen Verfahrens geringer im Vergleich zu dem unter diesen Annahmen optimalen parametrischen Test. „Das ist jedoch ganz natürlich, denn ein Werkzeug, das für viele Zwecke gebraucht wird, ist gewöhnlich nicht so wirkungsvoll für einen einzigen Zweck, wie das gerade für diesen Zweck entwickelte Werkzeug“ ([16], Seite 15).

- Da nichtparametrische Teststatistiken zumeist diskret sind, kann ein vorgegebenes Testniveau α in der Regel ohne Randomisierung nicht voll ausgeschöpft werden.

4 Beschreibung des Run-Tests

Die mathematischen Grundlagen für dieses Kapitel findet man in [6],[8] und [16].

4.1 Was ist ein Run-Test

4.1.1 Einführung

Der Run-Test ist ein nichtparametrischer Test (siehe Kapitel 3.6), dem die Anzahl der „Runs“ (Iterationen) in einer Stichprobe zugrunde liegt. Die Run-Tests können als Zufälligkeitstests oder als Tests zum Vergleich unabhängiger Stichproben benutzt werden.

Für gegeben Wert von n weist ein kleines r auf Klumpungen ähnlicher Beobachtungen hin, ein großes r auf einen regelmäßigen Wechsel.

Die Run-Theorie sagt:

Bei zufälliger Reihenfolge ist anzunehmen, dass sich die beiden Ausprägungen des Merkmals weder ganz regelmäßig abwechseln (viele Runs), noch, dass zuerst mehr die eine dann mehr die andere Ausprägung auftritt (wenige Runs).

Die Iterationstheorie (Run-Theorie) gibt die Wahrscheinlichkeitsverteilung und die Momente der zufälligen Anzahl der Iterationen in einer mathematischen Stichprobe vom Umfang n und Testverfahren zur Überprüfung der Hypothese an, dass ein vorliegendes n-Tupel von Zahlen eine Realisierung einer mathematischen Stichprobe aus einer vorgegebenen Grundgesamtheit ist.

*Im Unterschied zu anderen Testverfahren wird bei den Tests der Iterationstheorie in erster Linie **die Reihenfolge** der Beobachtungen der vorgegebenen Folge (des n-Tupels) berücksichtigt. Bei Run-Tests sind die Objekte nicht nur Zahlen, sondern andere Merkmalswerte (siehe Einleitung).*

4.1.2 Daten

Für die n Beobachtungen r_1 , ... , r_n ist jedes Messniveau zugelassen.

4.1.3 Annahmen

„Dieser Diplomarbeit liegt ein Spezialfall des Runs mit nur zwei Arten von Symbolen (Beobachtungen) zugrunde“ (Einleitung, Seite 7).

Sei X eine zweipunktverteilte Zufallsgröße mit zwei Merkmalswerten und (r_1 , ... , r_n) eine konkrete Stichprobe aus der zu X gehörenden Grundgesamtheit.

Es wird nur zwischen Beobachtungen vom Typ 1 und 2 unterschieden.

Falls die Daten mehrere Arten von Beobachtungen beinhalten, muss man die Beobachtungen in zwei Kategorien klassifizieren (siehe Kapitel 4.5).

4.1.4 Testproblem

H_0 : Die Reihenfolge der Beobachtungen ist zufällig.

„Die Alternativhypothesen für einen einseitigen oder zweiseitigen Run-Test, die in irgendeiner Weise einen systematischen Einfluss (bedingt zum Beispiel durch eine bestimmte Form der Abhängigkeit der Stichprobenvariablen) auf die Reihenfolge der Beobachtungen beinhalten, sind nicht so spezifiziert, wie das sonst bei Testproblemen der Fall ist“ ([16], Seite 118). Lautet die Alternative im eigentlichen Sinne „Nichtzufälligkeit“, so ist ein zweiseitiger Test zu wählen, bei Annahme eines Trends dagegen ein einseitiger Test.

Im Falle eines zweiseitigen Run-Tests lautet die Alternativhypothese:

H_1 : Die Reihenfolge der Merkmalswerte ist gegenüber dem, was bei Zufälligkeit zu erwarten wäre, verletzt.

Im Falle eines einseitigen Run-Tests kommen folgende Fälle als mögliche Alternativhypothesen in Frage:

a) gleiche Merkmalswerte treten zu häufig hintereinander auf („Klumpungseffekt“), das heißt es gibt zu wenige Runs (einseitiger Run-Test links),

b) die Merkmalswerte wechseln zu häufig („Regelmäßiger Wechsel“), das heißt zu viele Runs (einseitiger Run-Test rechts).

4.2 Teststatistik

Als Teststatistik betrachten wir die Anzahl *U* der Runs in der Reihenfolge der *n* Beobachtungen. Um die Verteilung von *U* unter H_0 herzuleiten, bezeichnen n_1 die Anzahl der Beobachtungen vom Typ 1 und n_2 die vom Typ 2.

4.2.1 Wahrscheinlichkeitsverteilung von *U*

Sei *U* Gesamtanzahl der Runs (beliebiger Länge),

$U = r_1 + r_2 = r$,
$n = n_1 + n_2$, so gilt für die Verteilung:

Formel 4.2.1

$$P(U) = \frac{2\binom{n_1 - 1}{u - 1}\binom{n_2 - 1}{u - 1}}{\binom{n}{n_1}}, \qquad \text{wobei } u = \frac{r}{2}$$

Falls *U* (die Gesamtanzahl von Runs) eine gerade Zahl ist.

Formel 4.2.2

$$P(U) = \frac{\binom{n_1 - 1}{u - 1}\binom{n_2 - 1}{u} + \binom{n_1 - 1}{u}\binom{n_2 - 1}{u - 1}}{\binom{n}{n_1}}, \qquad \text{wobei } u = \frac{r - 1}{2}$$

Falls *U* (die Gesamtanzahl von Runs) eine ungerade Zahl ist.

Auf der nächsten Seite werden die Formeln hergeleitet und in den nächsten Abschnitten werden wir mit Hilfe des Mathematica-Programms diese Formeln auf ihre Richtigkeit überprüfen.

4.2.2 Erwartungswert der Run-Verteilung

Formel 4.2.3

$$\mu_r = \frac{2n_1n_2}{n_1 + n_2} + 1 = \frac{2n_1n_2}{n} + 1$$

4.2.3 Varianz der Run-Verteilung

Formel 4.2.4

$$\sigma_r^2 = \frac{2n_1n_2(2n_1n_2 - n_1 - n_2)}{(n_1 + n_2)^2(n_1 + n_2 - 1)} = \frac{2n_1n_2(2n_1n_2 - n)}{n^2(n - 1)}$$

4.2.4 Graphische Darstellung der Run-Verteilung

In dieser Diplomarbeit wird die Run-Verteilung mit Balkendiagrammen und Treppenfunktionen dargestellt.

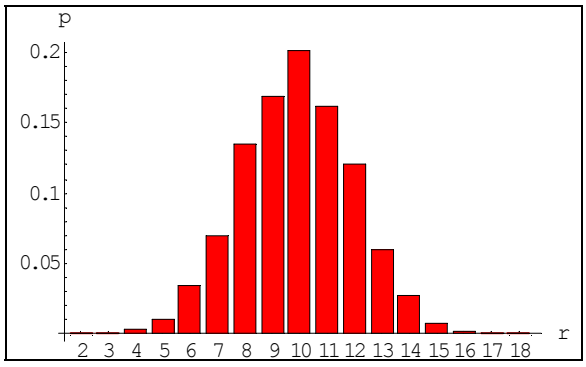

Abbildung 4.1: Balkendiagramm für n=18

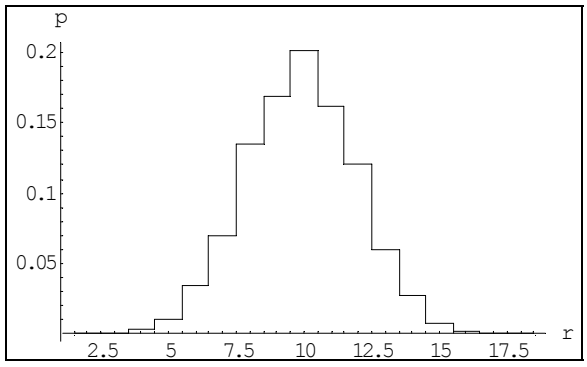

Abbildung 4.2: Treppenfunktion für n=18

Man sieht deutlich, dass die Treppenfunktion bei der Run-Verteilung nicht symmetrisch bezüglich μ ist, wie das bei der Normalverteilung der Fall ist.

Die Form der Run-Wahrscheinlichkeitsfunktion hängt von n, aber auch von n1 und n2 ab. Nicht nur die Gesamtanzahl der Objekte n=n1+n2 hat einen Einfluss auf das Aussehen der Funktion. Die Abbildung 4.3 zeigt, dass die Treppenfunktion der Run-Verteilung mit n1=n2=16 andere Form hat als die mit n1=20 und n2=12, obwohl n (die Gesamtanzahl der Objekte) bei beiden gleich ist.

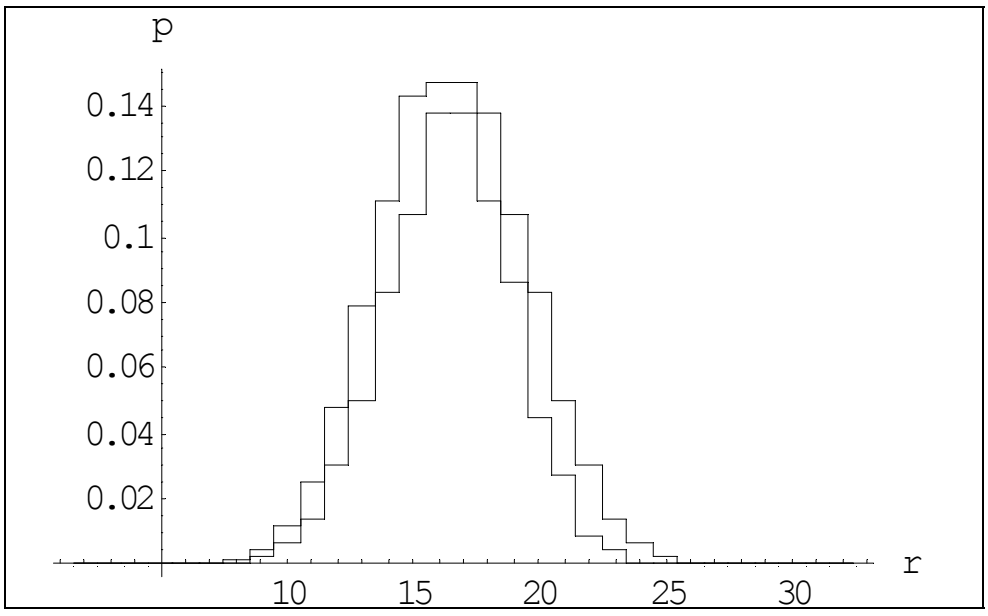

Abbildung 4.3: Treppenfunktionen zur Run-Wahrscheinlichkeitsverteilung für n=32. Einmal für n1=n2=16, einmal für n1=20, n2=12.

Die nächsten Abbildungen zeigen die Wahrscheinlichkeitsfunktionen und die zugehörigen Verteilungsfunktionen für kleine (n=100), mittlere (n=500) und große Anzahl (n=1000) von Beobachtungen. In allen drei Fällen ist n1= n2.

Fall 1, n=100

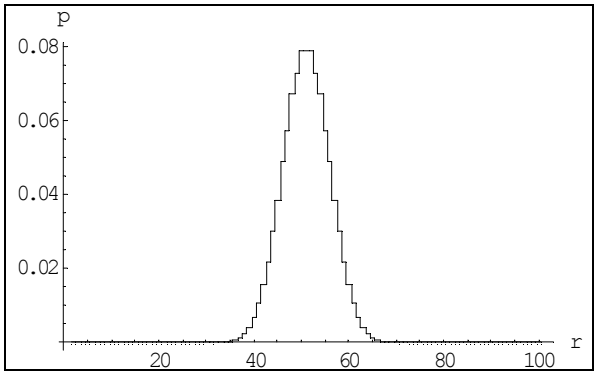

Abbildung 4.4 A: Wahrscheinlichkeitsfunktion für n=100, n1=n2=50, μ = 51, σ =4,975.

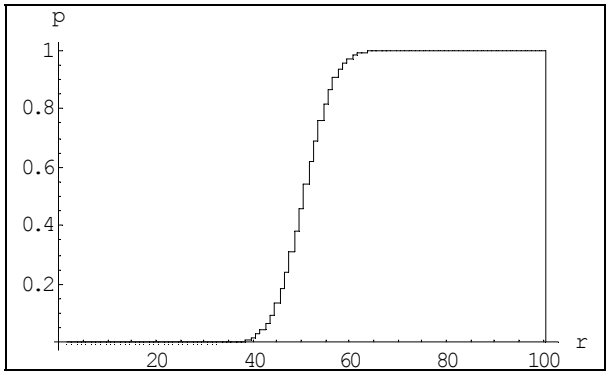

Abbildung 4.4 B: Zugehörige Verteilungsfunktion

Fall 2, n=500

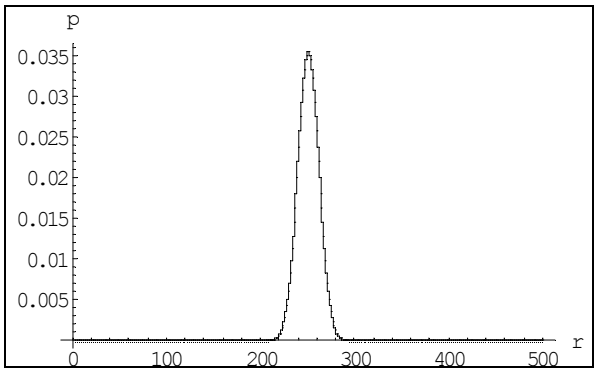

Abbildung 4.5 A: Wahrscheinlichkeitsfunktion für n=500, n1=n2=250, μ = 251, σ =11,169.

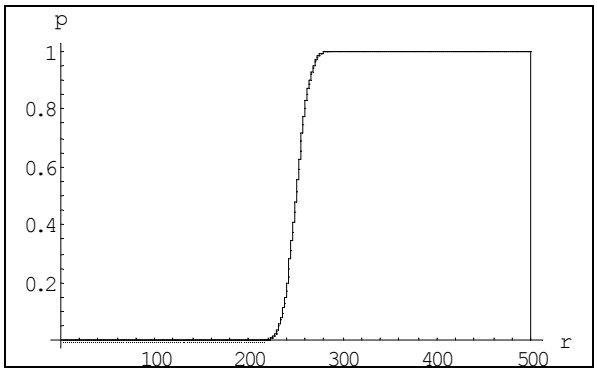

Abbildung 4.5 B: Zugehörige Verteilungsfunktion

Fall 3, n=1000

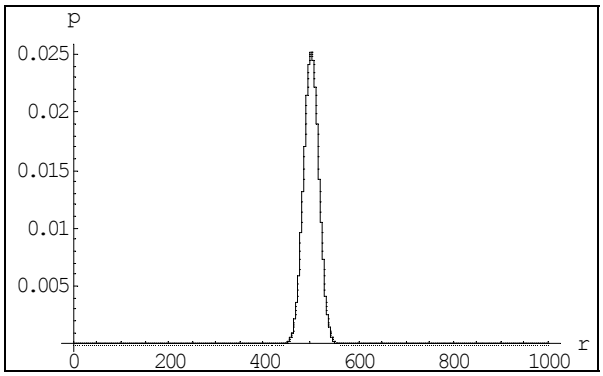

Abbildung 4.6 A: Wahrscheinlichkeitsfunktion für n=1000, n1=n2=500, μ = 501, σ =15,8.

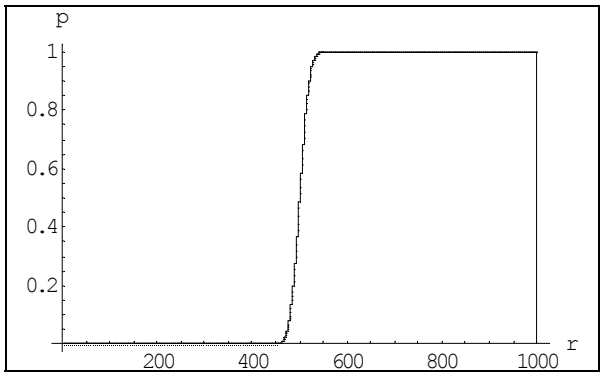

Abbildung 4.6 B: Zugehörige Verteilungsfunktion

Vergleich der drei Wahrscheinlichkeitsfunktionen

In der Abbildung 4.7 werden alle drei Funktionen zusammengestellt.

Betrachtet man die untere Abbildung, so sieht man, dass je größer n ist, um so flacher ist der Kurvenverlauf der Wahrscheinlichkeitsfunktion und um so niedriger liegt das Maximum.

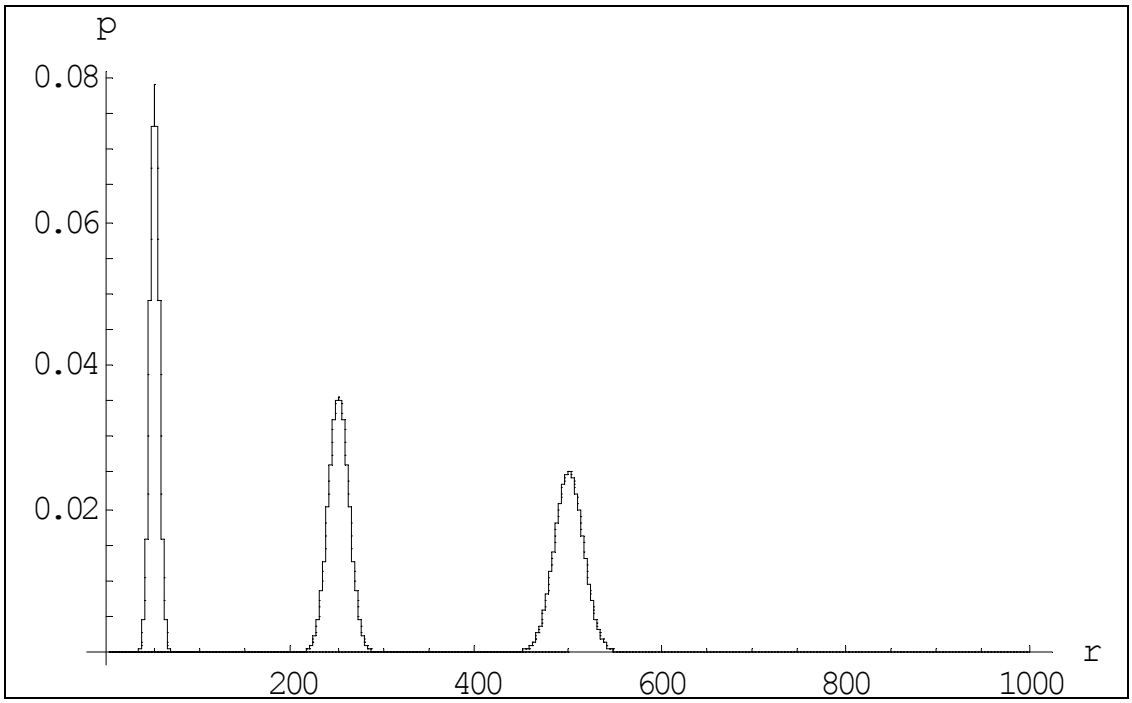

Abbildung 4.7: Zusammenstellung der Treppenfunktionen zur Run-Verteilung für n=100, n=500, n=1000 (von links betrachtet)

4.3 Herleitung der Formeln für die Run-Wahrscheinlichkeitsverteilung

In einer Folge mit zwei Arten von Beobachtungen ist:

<u>Fall 1</u> $r_2 \neq r_1$

a) $r_1 = r_2 + 1$
b) $r_2 = r_1 + 1$

<u>Fall 2</u> $r_1 = r_2$.

Im ersten Fall muss die Folge mit dem Run der Objekte vom Typ 1 (oder vom Typ 2) anfangen und enden, im zweiten Fall fängt die Reihe mit dem Run der Objekte vom Typ1 an, endet aber mit dem Run der Objekte vom Typ 2, oder umgekehrt.

Die Anzahl der unterschiedlichen Permutationen ist für den Run der Objekte vom Typ 1 gleich

$$\frac{r_1!}{r_{11}!\,r_{12}!\ldots r_{1n_1}!}$$

und entsprechend für den Run der Objekte vom Typ 2 gleich

$$\frac{r_2!}{r_{21}!\,r_{22}!\ldots r_{2n_2}!}.$$

Falls H_0 wahr ist, kann argumentiert werden, dass die Anordnungsmöglichkeiten (Permutationen) für n_1 Werte der Objekte vom Typ 1 und n_2 Werte der Objekte vom Typ 2 gleich wahrscheinlich sind. Es ist klar, dass es genau $\binom{n_1+n_2}{n_1}$ Anordnungen gibt.

Um P(U) zu finden, ist es notwendig, alle Anordnungen mit genau U Runs zu zählen. Angenommen, U ist gerade, sagen wir U=2u, dann muss es u Runs der Objekte vom Typ 1 und u Runs der Objekte vom Typ 2 geben. Um u Anordnungen der Objekte vom Typ 1 zu bekommen, müssen die n_1 Objekte vom Typ 1 in u-Gruppen geteilt werden. Wir können diese u-Gruppen oder Runs bilden, indem wir den Teiler $u-1$ in die n_1-1Räume (Abstände zwischen den n_1 Werten der Objekte vom Typ 1) einsetzen, aber nicht mehr als einen Teiler pro Zwischenraum. Wir können diese $u-1$-Teiler in die n_1 -Räume auf $\binom{n_1-1}{u-1}$ Arten platzieren. Entsprechend können wir die u Runs für die Objekte vom Typ 2 konstruieren, auf $\binom{n_2-1}{u-1}$ Arten.

Jede bestimmte Anordnung der u Runs der Objekte vom Typ 1 kann mit irgendeiner Anordnung der u Runs der Objekte vom Typ 2 kombiniert werden. Außerdem kann der erste Run in der kombinierten Anordnung entweder der Run der Objekte vom Typ1 oder 2 sein.

Folglich, gibt es eine Gesamtanzahl von $\binom{n_1-1}{u-1}\binom{n_2-1}{u-1}$ Anordnungen, die genau U=2u Runs haben, $n = n_1 + n_2$.

Es folgt für gerade Anzahl von Runs:

$$P(U = u) = P(U = 2u) = \frac{2\binom{n_1 - 1}{u - 1}\binom{n_2 - 1}{u - 1}}{\binom{n}{n_1}} \text{ (siehe Formel 4.2.1).}$$

Entsprechend folgt für eine ungerade Anzahl von r:

$$P(U = u) = P(U = 2u) = \frac{\binom{n_1 - 1}{u - 1}\binom{n_2 - 1}{u} + \binom{n_1 - 1}{u}\binom{n_2 - 1}{u - 1}}{\binom{n}{n_1}} \quad \text{(siehe Formel 4.2.2).}$$

Angenommen, wir interessieren uns nur für die Anzahl der Runs der Objekte (Elemente) vom Typ 1, gleichgültig, ob $r_2 = r_1 - 1$, $r_2 = r_1$ oder $r_2 = r_1 + 1$ ist. Wir wählen die Runs r_1 von Elementen des Typs 1 aus, indem wir $r_1 - 1$ von den $n_1 - 1$ Zwischenräumen auswählen. Jetzt sind die Zwischenräume vor und nach, sowie zwischen den Elementen des Typs 2 für die Besetzung der Elemente des Typs 1 vorhanden, da die Anzahl der Runs von Elementen des Typs 2 nicht festgelegt ist. Deswegen gibt es $n_2 + 1$ Räume, die für das Platzieren der r_1-Runs benutzt werden. Sie können auf $\binom{n_2 + 1}{r_1}$ Arten platziert werden. Der Rest der Herleitung ist analog zu dem, was vorher beschrieben wurde.

Es folgt, die Wahrscheinlichkeit dafür, dass es genau r_1 Runs der Elemente des Typs 1 gibt, ist:

$$P(r_1) = \frac{\binom{n_1 - 1}{r_1 - 1}\binom{n_2 + 1}{r_1}}{\binom{n}{n_1}} = \frac{\binom{n_1 - 1}{r_1 - 1}\binom{n_2 + 1}{r_1}}{\binom{n_1 + n_2}{n_1}}.$$

Man kann leicht erkennen, dass es sich hier um eine hypergeometrische Verteilung handelt. Die Faktoren der oberen Ebene im Nenner sind gleich der Summe der Faktoren der oberen Ebene im Zähler. Entsprechend das gleiche gilt für die Faktoren der unteren Ebene. Betrachten wir nun die Formel für die Wahrscheinlichkeitsfunktion der hypergeometrischen Verteilung:

$$f(x) = \frac{\binom{M}{x}\binom{N - M}{n - x}}{\binom{N}{n}} \quad \text{(siehe Formel 3.3.1)}$$

Es ist tatsächlich die Wahrscheinlichkeit, dass von den n_2+1 Zellen, die von Elementen des Typs 2 belegt sind, n_2+1-r_1 leer bleiben. Das heißt, sie werden die r_1 -Runs nicht beinhalten.

Schlussfolgerung

Unsere Verteilung lässt sich durch die hypergeometrische Verteilung annähern, diese approximiert durch die Binomialverteilung. Für sehr große Werte von n lässt sich die Binomialverteilung durch die Normalverteilung annähern.

4.4 Testprozeduren

Bei einer vorgegebenen Irrtumswahrscheinlichkeit α würde die Testprozedur folgendermaßen aussehen:

Ist keine Richtung der Abweichung der Zufälligkeit ausgezeichnet (zweiseitiger Run-Test), so wird die Nullhypothese abgelehnt, wenn die beobachtete Anzahl von Runs rbeob < $r_{\frac{\alpha}{2}}$ oder rbeob > $r_{1-\frac{\alpha}{2}}$ ist. Die vorher festgesetzte Richtung der Abweichung von der Zufälligkeit (einseitiger Run-Test) kann hinweisen auf:

a) zu wenig Runs (einseitiger Test links), das heißt H_0 wird abgelehnt, wenn rbeob < r_α ist,

b) zu viele Runs (einseitiger Test rechts), das heißt H_0 wird abgelehnt, wenn rbeob > $r_{1-\alpha}$ ist.

4.4.1 Die exakte Auswertung der dem Run-Test zugrunde liegenden Wahrscheinlichkeitsverteilung

Bei dem einseitigen und zweiseitigen Run-Test wird *die exakte Irrtumswahrscheinlichkeit* α berechnet, mit welcher die Nullhypothese zugunsten der gestellten Alternativhypothese abgelehnt wird.

Eine bestimmte Anzahl von Runs, anders gesagt: die Anzahl der beobachteten Runs, für welche die Irrtumswahrscheinlichkeit bestimmt werden soll, wird mit *rbeob* bezeichnet.

Zweiseitiger Run-Test

Im Falle eines zweiseitigen Tests gibt es folgende Möglichkeiten der Berechnung von *alpha*:

Um zu entscheiden, ob wir zuerst links oder rechts aufsummieren, legen wir zuerst als Richtwert den Erwartungswert μ (Formel 4.2.3) fest.

<<< Fälle für rbeob < μ >>>

Fall 1 A

Ist der Wert von *rbeob* kleiner als der Erwartungswert μ, dann summieren wir die Wahrscheinlichkeiten links auf. Ist die berechnete Summe $\sum_{r=2}^{rbeob} P(U=r) < 0{,}5$, dann ist die Berechnung beendet und es gilt:

Formel 4.4.1 $$\alpha = 2 \sum_{r=2}^{rbeob} P(U=r).$$

Ist dagegen $\sum_{r=2}^{rbeob} P(U=r) \geq 0{,}5$, dann rechnen wir die Werte von P(U=r) rechts zusammen und es kann zwei weitere Fälle geben.

Ist $\sum_{r=rbeob}^{n} P(U=r) < 0{,}5$ haben wir den Fall 2 A. Es gilt:

Formel 4.4.2 $$\alpha = 2 \sum_{r=rbeob}^{n} P(U=r).$$

Für einen schnellen Ablauf wird im Programm die folgende Formel benutzt, die äquivalent zur Formel 4.4.2 ist.

Formel 4.4.2(P) $\alpha = 2\,(1 - \sum_{r=2}^{rbeob} P(U=r) + P(U=rbeob))$.

Ist $\sum_{r=rbeob}^{n} P(U=r) \geq 0{,}5$, dann haben wir den Fall 3 A. Es gilt: **Formel 4.4.3** $\alpha = 1$.

Den Fall 3A wollen wir graphisch veranschaulichen.

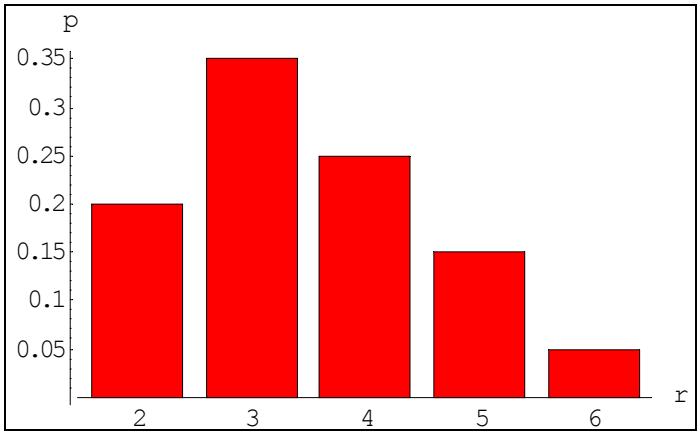

Abbildung 4.8: Balkendiagramm - Wahrscheinlichkeitsverteilung für n=6

Es liegt hier folgende Wahrscheinlichkeitsverteilung vor:

für r=2, P(U=2)=0,2
für r=3, P(U=3)=0,35
für r=4, P(U=4)=0,25
für r=5, P(U=5)=0,15
für r=6, P(U=6)=0,05.

Es soll α für *rbeob*=3 bestimmt werden.

Der Wert von *rbeob*=3 ist kleiner als der Erwartungswert (=4), trotzdem ist die Summe

$$\sum_{r=2}^{rbeob} P(U=r) = \sum_{r=2}^{3} P(U=r) = 0{,}55$$

größer als 0,5.

Wir berechnen die Wahrscheinlichkeiten nun auf der anderen Seite

$$\sum_{r=rbeob}^{n} P(U=r) = \sum_{r=3}^{6} P(U=r) = (0{,}35+0{,}25+0{,}15+0{,}05) = 0{,}8 > 0{,}5$$

und stellen fest, dass auch diese Summe größer als 0,5 ist.

Daraus folgt: $\alpha = 1$ (siehe Formel 4.4.3).

<<< Fälle für rbeob $\geq \mu$ >>>

Fall 1 B

Ist der Wert von *rbeob* größer gleich dem Erwartungswert, dann summieren wir die Wahrscheinlichkeiten rechts auf. Ist die Summe $\sum_{r=rbeob}^{n} P(U=r) < 0{,}5$, dann ist die Berechnung beendet und es gilt:

$$\alpha = 2 \sum_{r=rbeob}^{n} P(U=r) \quad \text{(nach der Formel 4.4.2).}$$

Ist dagegen $\sum_{r=rbeob}^{n} P(U=r) \geq 0{,}5$, dann rechnen wir die Werte von P(U=r) links zusammen und es kann zwei weitere Fälle geben.

Ist $\sum_{r=2}^{rbeob} P(U=r) < 0{,}5$ haben wir den Fall 2 B und es gilt:

$$\alpha = 2 \sum_{r=2}^{rbeob} P(U=r) \quad \text{(nach der Formel 4.4.1)}$$

oder für den schnellen Programmablauf:

Formel 4.4.1(P) $\alpha = 2 \left(1 - \sum_{r=rbeob}^{n} P(U=r) + P(U=rbeob)\right)$

Ist dagegen $\sum_{r=2}^{rbeob} P(U=r) \geq 0{,}5$ haben wir den Fall 3 B und es gilt:

$\alpha = 1$ (siehe Formel 4.4.3).

Einseitiger Run-Test

Wie die Abbildung 4.9 zeigt, werden bei linksseitigen Run-Tests die Balken links aufsummiert (z.B. für rbeob=8). In der Zeichnung sind das die durchnummerierten Balken.

Die Abbildung 4.10 veranschaulicht den rechtsseitigen Test (für rbeob=12).

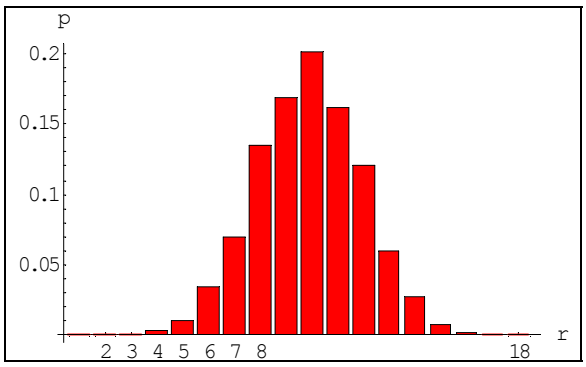

Abbildung 4.9: Wahrscheinlichkeitsverteilung, n=18, rbeob=8 (linksseitiger Test)

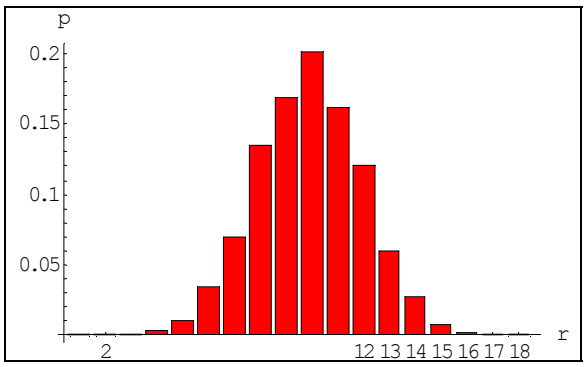

Abbildung 4.10: Wahrscheinlichkeitsverteilung, n=18, rbeob=12 (rechtsseitiger Test)

Im Falle eines einseitigen Tests gelten folgende Formeln:

Die Irrtumswahrscheinlichkeit α ist gleich

Formel 4.4.4 $$\alpha = \sum_{r=2}^{rbeob} P(U = r)$$ für **den linksseitigen Test**

und

Formel 4.4.5 $$\alpha = \sum_{r=rbeob}^{n} P(U = r)$$ für **den rechtsseitigen Test**.

Diesen exakten Test (zweiseitigen und einseitigen) ermöglicht das Hauptprogramm dieser Diplomarbeit (siehe Programmbeschreibung im Kapitel 5.7 und Notebook im Kapitel 7.1.9).

4.4.2 Approximation der Run-Verteilung durch die Normalverteilung

Für sehr große Werte von *n* können die Quantile der standardisierten Normalverteilung genommen werden, da die Run-Verteilung in diesem Fall gut durch die Normalverteilung (mit dem Erwartungswert 4.2.3 und der Varianz 4.2.4) approximiert wird. Der kritische Wert *z* ist damit:

Formel 4.4.7

$$z = \frac{r - \mu_r}{\sigma_r} = \frac{r - \left(\frac{2n_1n_2}{n_1 + n_2} + 1\right)}{\sqrt{\frac{2n_1n_2(2n_1n_2 - n_1 - n_2)}{(n_1 + n_2)^2(n_1 + n_2 - 1)}}}$$

Damit fällt die Entscheidung für die Nullhypothese nach dem folgenden Kriterium:

$$\left|\frac{r - \mu_r}{\sigma_r}\right| < \left|z_{\frac{\alpha}{2}}\right| \Rightarrow H_0$$

$$\left|\frac{r - \mu_r}{\sigma_r}\right| \geq \left|z_{\frac{\alpha}{2}}\right| \Rightarrow H_1 \text{ mit Risiko } \alpha$$

wobei $z_{\frac{\alpha}{2}}$ das Quantil der Normalverteilung ist.

In dieser Diplomarbeit wird gezeigt, wie gut die Annäherung mit der Normalverteilung für verschiedene Werte von *n* ist. Zahlreiche Graphiken für kleine und große *n* zeigen in den nächsten Kapiteln die Approximation der Run-Verteilung durch die Normalverteilung.

Die zwei unteren Abbildungen zeigen allgemein, dass für große *n* die Näherung durch die Normalverteilung viel besser ist als für ganz kleine.
(rote Linie - Kurve der Normalverteilung, schwarze Linie - Kurve der Run-Verteilung)

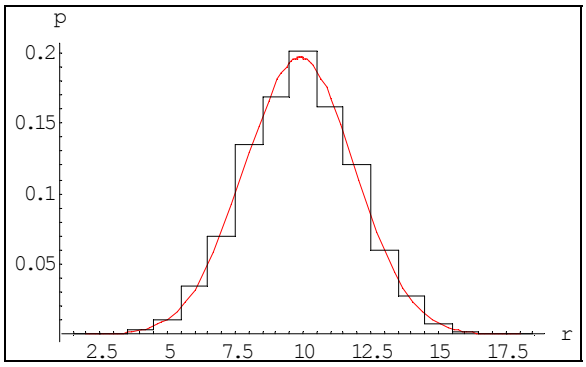

Abbildung 4.11:Approximation der Run-Verteilung durch die Normalverteilung für n=18

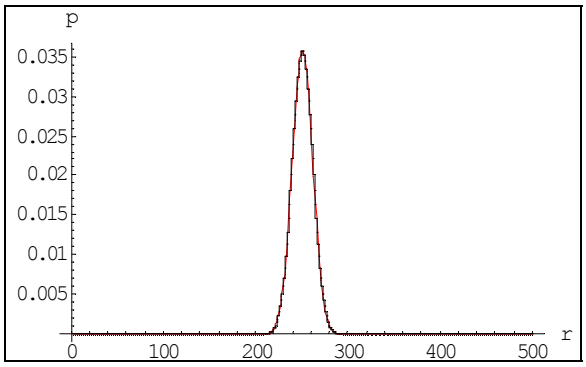

Abbildung 4.12:Approximation der Run-Verteilung durch die Normalverteilung für n=500

Vergleicht man nun die beiden Abbildungen, so erkennt man leicht, dass die Kurve der Run-Verteilung in der rechten Abbildung (für große *n*) näher an der Kurve der Normalverteilung entlang läuft als bei der linken Abbildung (für kleine *n*).

4.5 Testmethoden

In diesem Kapitel werden zum einen die Klassifizierungsmethoden für den Zufälligkeitstest vorgestellt. Zum anderen wird der Test zum Vergleich unabhängiger Stichproben beschrieben.

4.5.1 Klassifizierungsmethoden für den Zufälligkeitstest

Da die meisten Daten nicht nur zwei Arten von Beobachtungen haben, ist eine Klassifizierung notwendig. Manchmal kann man die Testdaten gleich in zwei Kategorien unterteilen, zum Beispiel die Spielkarten in rote und schwarze Karten, die Kinder in Mädchen und Jungs usw. (siehe Beispiele aus der Einleitung). Meistens ist aber eine kompliziertere Methode erforderlich.

Einige von ihnen, die für diese Diplomarbeit erforderlich sind, werden in diesem Kapitel vorgestellt.

Die beiden Methoden sind allerdings nur bei quantitativen Daten anzuwenden.

Die folgenden Klassifizierungsmethoden werden durch die Mathematica-Hilfsprogramme unterstützt.

„Runs up and down“-Methode

In der Reihe wird jede Beobachtung mit der unmittelbar folgenden Beobachtung verglichen.
Ist die nächste Beobachtung größer, beginnt ein „run up“ (Ansteigung), ist sie kleiner, ein „run down“ (Absteigung).

Zu wenige Ansteigungen und Absteigungen deuten auf einen Trend hin.

Da manche nebeneinander stehende Zahlen in unseren Exempel gleich sind, müssen wir den „runs up and down“ - Test auf zwei Arten durchführen.

Im ersten Test entsteht die Folge folgendermaßen:

Nächste Zahl $\leq$ Vorherige Zahl $\rightarrow$ 0
Nächste Zahl > Vorherige Zahl $\rightarrow$ 1

Im zweiten Test bekommt das nachfolgende Element, das mit dem vorigen gleich ist, eine Eins und nicht eine Null wie im ersten Test.

Nächste Zahl < Vorherige Zahl $\rightarrow$ 0
Nächste Zahl $\geq$ Vorherige Zahl $\rightarrow$ 1

Diese Vorgehensweise wird beim Testen der Schwankungen der Treibstoffpreise, Währungsschwankungen, sowie beim Testen der Zufallszahlen eingesetzt (siehe Kapitel 6.2, 6.3 und 6.4).
Das Klassifizieren von Beobachtungen nach dieser Methode erleichtert das Programm *runs_up_and_down_Test* (siehe Kapitel 5.6.1 und 7.1.7).

Methode der geraden und ungeraden Zahlen

Nach dieser Methode entsteht die Folge der Einser und Nullen folgenderweise:

Eine gerade Zahl bekommt $\rightarrow$ 0
Eine ungerade Zahl bekommt $\rightarrow$ 1

So entstandene Zahlenfolgen werden auf Zufälligkeit mit dem zweiseitigen Run-Test getestet. Ein Beispiel dazu findet man im Kapitel 6.4 bei den Zufallszahlen.

Das Klassifizieren von Beobachtungen nach dieser Methode macht das Programm *gerade_ungerade_Test* möglich (siehe Kapitel 5.6.2 und 7.1.8).

4.5.2 Zweistichproben-Problem für unabhängige Stichproben

Wir haben den Run-Test bereits als einen Test auf Zufälligkeit kennengelernt. Er lässt sich auch für das Zweistichproben-Problem formulieren.

Das Zweistichproben-Problem kann wie folgt beschrieben werden: Es seien zwei unabhängige Stichprobenvariablen $x_1, \ldots, x_m$ und $y_1, \ldots, y_n$ aus einer Grundgesamtheit mit unbekannten (stetigen) Verteilungsfunktionen F und G. Der Test soll zeigen, dass die beiden Stichproben der gleichen Grundgesamtheit entstammen. Die Hypothesen beziehen sich auf den Vergleich der beiden zugrundeliegenden Verteilungen des Merkmals.

Die parametrischen Tests (z.B. der t-Test) untersuchen, ob die Verteilungsfunktionen der beiden Stichproben sich durch die Erwartungswerte oder Varianzen unterscheiden.

Der nichtparametrische Run-Test prüft, ob die aus beiden kombinierte und geordnete Stichprobe ein bestimmtes Muster hat. Ist die Reihenfolge der beiden Beobachtungen (z.B. Körpergrößen der Mädchen und Jungen) zufällig, dann entstammen die beiden Stichproben der gleichen Grundgesamtheit.

Prinzipiell werden hier zwei Merkmale X und Y unterschieden, deren Verteilungen verglichen werden. Um die Verteilungen miteinander vergleichen zu können, müssen die Merkmale X und Y sinnvoll vergleichbare Größen darstellen. Meist stellen X und Y dasselbe Merkmal dar, allerdings gemessen unter verschiedenen Bedingungen.

Ein Beispiel:

Im Rahmen einer medizinischen Untersuchung an Schulanfängern sollen in einer Großstadt die Körpergrößen der 6jährigen Mädchen und Jungen miteinander verglichen werden.
Die Frage ist: Liegt bezüglich der Körpergröße bei den Mädchen und Jungen dieselbe Verteilung vor?

Daten

Die Beobachtungen $x_1, \ldots, x_m$ und $y_1, \ldots, y_n$, wobei die Stichprobenumfänge unterschiedlich sein können. Die Stichprobenvariablen sind unabhängig.

Testproblem

H_0 : F(z) =G(z) für alle $z \in R$

H_1 : $F(z) \neq G(z)$ für mindestens ein $z \in R$

Teststatistik

Wir bilden aus den beiden Stichproben die kombinierte, aufsteigend (oder absteigend) sortierte Stichprobe und kennzeichnen mit 0 bzw. 1, ob es sich jeweils um einen x_i - Wert oder einen y_j - Wert handelt.
Als Teststatistik wählen wir die Anzahl *rbeob* der Runs in der kombinierten, geordneten Stichprobe.

ZWEISEITIGER TEST

Unter der Nullhypothese ist zu erwarten, dass die Einser und Nullen „gut gemischt“ sind, das heißt nicht zu viele Runs und nicht zu wenige Runs. Mit anderen Worten, die Reihenfolge der Beobachtungen aus beiden Stichproben ist zufällig.

EINSEITIGER TEST

Bei bestimmten Zweistichproben-Problemen ist auch ein einseitiger Test angebracht (siehe Kapitel 6.5).
In diesem Fall lauten die Alternativhypothesen (nach 4.1.4):

a) gleiche Merkmalswerte treten zu häufig hintereinander auf, das heißt, es gibt zu wenige Runs, „Klumpungseffekt“ - für den linksseitigen Run-Test,

b) die Merkmalswerte wechseln zu häufig, das heißt zu viele Runs, „Regelmäßiger Wechsel“ - für den rechtsseitigen Run-Test.

5 Automatisierung des Run-Tests

Die Software zu dieser Diplomarbeit wurde in Mathematica Version 3.0 erstellt. Im folgenden ein Überblick über die erstellten Programme:

Hauptformeln der Run-Verteilung

Prüfen der Formeln

Erwartungswert, Varianz und Standardabweichung

Graphische Darstellung der Run-Verteilung:

- Balkendiagramme für kleine n
- Treppenfunktionen für große n
- Approximation der Run-Verteilung durch die Normalverteilung, kleine n
- Approximation der Run-Verteilung durch die Normalverteilung, große n

Hilfsprogramme:

- Hilfsprogramm zur „runs up and down“ Methode
- Hilfsprogramm zur Methode der geraden und ungeraden Zahlen
- Hilfsprogramm zum Zählen von *rbeob, n1* und *n2*

Run-Test

Außerdem wurden noch andere Programme für Graphik und Hilfsberechnungen zu dieser Diplomarbeit geschrieben. Da sie nur für die Erstellung dieser Arbeit wichtig waren, werden sie nicht mehr erwähnt.

5.1 Anleitung zur Benutzung von Mathematica

5.1.1 Was ist Mathematica

Mathematica ist ein unbegreiflich leistungsstarkes Hilfsmittel zur Durchführung mathematischer Berechnungen in den Kategorien Graphik, Numerik, Statistik, Analysis und Algebra. Es ist ein Programm, um Mathematik mit dem Computer zu machen. Mathematica arbeitet mit Zahlen ebenso wie mit Symbolen, es löst symbolische Gleichungen, beherrscht numerische Mathematik, hat umfangreiche Graphikfähigkeiten, lässt sich programmieren, verwaltet Wissen und beherrscht so grundlegende mathematische Funktionen, dass man seine eigenen Mathematik formulieren und ausführen lassen kann. Mathematica vereinigt damit Fähigkeiten, die sich in vielen anderen Programmen finden, unter einer konsistenten Oberfläche. Die

Benutzeroberfläche ist allen Funktionen gleich, und es gibt keinen Übergang von einem Mathematikbereich zum anderen.

Mathematica ersetzt eine ganze Palette von Programmen, Programmpaketen und Programmiersprachen. Es ist einfach, Randbedingungen zu variieren und man kann sofort ihren Einfluss auf die Lösung sehen.

Mathematica ist zwar technisch ein leistungsfähiges Programm, aber seine Bedienung ist gewöhnungsbedürftig. Es geht hier weniger um formale Dinge (wie eckige Klammern oder die Groß- und Kleinschreibung) sondern um ein Gespür, mit welchem Kommando ein Ausdruck am ehesten vereinfacht werden kann, in welcher Form Mathematica ein Problem am ehesten lösen kann, wie man auch in längeren Berechnungen den Überblick bewahren kann.

Mit der Verfügbarkeit von Mathematica auf populären Rechnern ist es durchaus möglich, dass Mathematica der wissenschaftliche Taschenrechner der Zukunft wird - ein Werkzeug, dessen Verwendung auf höheren Schulen, auf Universitäten, in der Industrie so selbstverständlich wird, wie heute der konventionelle Taschenrechner. Außerdem kann man mit Mathematica wie in C programmieren. Diese Fähigkeiten wurden beim Schreiben der Programme in dieser Diplomarbeit voll ausgenutzt.

Voraussetzungen

Um die Programme verwenden zu können, benötigt man einen IBM-kompatiblen PC, Windows und Mathematica Version 3.
Für einen schnellen und problemlosen Ablauf sollte der Rechner mit 16 MByte Arbeitsspeicher ausgestattet werden. Auf der Festplatte müssen mindestens 12 MByte freier Speicherplatz zur Verfügung stehen.

5.1.2 Wichtigste Regeln für den Umgang mit Mathematica

1. Namen für eigene Symbole

Alle Mathematica-Symbole und -Kommandos beginnen mit Großbuchstaben, deswegen wird dem Benutzer empfohlen, für eigene Symbole Kleinbuchstaben zu verwenden. Die Namen dürfen grundsätzlich alle Buchstaben beinhalten. Der Unterstrich _ darf nicht in Variablen verwendet werden, da dieses Zeichen in Mathematica reserviert ist. Der Unterstrich wird korrekt in Parametern bei der Definition von neuen Funktionen und Kommandos verwendet. Hinter dem Unterstrich kann der erforderliche Datentyp oder eine andere Bedingung für den Parameter angegeben werden.

Ein Beispiel aus dem Hauptprogramm Run-Test:

```
formel1[n1_, n2_, u_] := 2 (Binomial[n1 - 1, u - 1] Binomial[n2 - 1, u - 1]) / Binomial[n1 + n2, n1];
```

2. Multiplikation

Die Multiplikation muss in Mathematica nicht durch „ * „ ausgedrückt werden. Es reicht ein Leerzeichen.

3. Operator „ ; „

Eingaben, die mit „ ; „ abgeschlossen werden, werden normal ausgewertet, allerdings erfolgt keine Ausgabe am Bildschirm.

Außerdem kann man mit dem Operator „ ; „ mehrere Kommandos auf einmal eingeben. Zum Beispiel:

```
x = 3; y = 5.5; z = 19;
```

4. Unterschiede zwischen ( ), [ ],[[ ]] und {}

In Mathematica gibt es verschiedene Arten von Klammern:

- ( ) werden verwendet, um Kommandos zu gruppieren (siehe das erste Beispiel) und die Auswertungsreihenfolge zu bestimmen (siehe das zweite Beispiel).

Beispiel 1:

```
ausgabeeinslinks[] := (Print["Einseitiger Test links"];
          Print["Mit einer Irrtumswahrscheinlichkeit alpha =",
                  PaddedForm[100 alpha, {NS + 3, NS}], " %", "wird Ho abgelehnt zugunsten der Alternativhypothese."];);
```

Beispiel 2:

```
m = (2 n1 n2 / (n1 + n2)) + 1;
```

- [ ] enthalten die Parameter bei allen Mathematica-Kommandos und eigenen Funktionen; mehrere Parameter werden durch Kommata getrennt

- { } dienen zur Eingabe und Ausgabe von Listen, Matrizen und Vektoren, z.B.:

```
raucher = { 132, 146, 153, 130, 146, 126 };
```

Die geschweiften Klammern können mehrfach verschachtelt werden, man findet sie in vielen Optionen, z.B. in Plot-Optionen :

```
AxesLabel -> { r, p }
```

- [[ ]] doppelte eckige Klammern dienen zum Indizieren

```
Do[ If [ zuzpm [ [ i ] ] != zuzpm [ [ i + 1 ] ], rbeob = rbeob + 1 ], { i, 1, m – 1 } ]
```

6. Setzen der Operatoren =, := und ==

Unterschiede zwischen den drei Operatoren:

„symbol = ausdruck — Zuweisungsoperator: Weist dem Symbol auf der linken Seite sofort den vorher ausgewerteten Term auf der rechten Seite zu.

symbol := ausdruck — Verzögerter Zuweisungsoperator: Weist dem Symbol auf der linken Seite den Term auf der rechten Seite zu. Eine Auswertung des Terms erfolgt aber erst bei der Verwendung des Symbols. Aus diesem Grund wird das Ergebnis der Definition auch nicht angezeigt.

ausdruck1 == ausdruck2 — Stellt fest, ob die beiden Terme links und rechts gleich sind. Dabei werden allerdings keine mathematischen Umformungen vorgenommen. == wird vor allem zur Definition von Gleichungen verwendet“ [12, Seite 82]. Im Hauptprogramm dieser Diplomarbeit wird dieser Operator bei den If-Abfragen verwendet (siehe unten).

```
If[testwahl == 1, (berechnungzwei[]; testzwei[]; ausgabezwei[];),

                If[testwahl == 2, (berechnungeinslinks[]; testeinslinks[]; ausgabeeinslinks[];),

                                        (berechnungeinsrechts[]; testeinsrechts[]; ausgabeeinsrechts[];)

                ]; (*ende von else von If testwahl=2*)
        ];
(*ende von else von If testwahl=1*)
```

7. Arbeiten mit „Packages“

Mathematica stellt zahllose Kommandos standardmäßig zur Verfügung. Diese Kommandos können sofort verwendet werden. Daneben existieren in sogenannten „Packages“ viele weitere Spezialkommandos für verschiedene Teilbereiche der Mathematik. Hier einige Beispiele aus den Notebooks dieser Diplomarbeit.

Mit diesem Kommando wird das Package für alle Statistik-Funktionen geladen:

```
Needs["Statistics`Master`"];
```

Nach diesem Befehl:

```
Needs["Graphics`Graphics`"];
```

stehen verschiedene Graphik-Kommandos zur Verfügung, z.B. für BarChart.

5.2 Hauptformeln der Run-Verteilung

Programmname

Hauptformeln (siehe Notebook, Kapitel 7.1.1)

Programmbeschreibung

Mit dem Mathematica-Programm können wir bequem die Formeln 4.2.1 und 4.2.2 einprogrammieren, so dass mit der einfachen Eingabe von *n1, n2* und *rbeob* die entsprechende Wahrscheinlichkeit P(U) berechnet wird.

Die Formel 4.2.1 (siehe Kapitel 4.2.1) wird in Mathematica folgendermaßen umgesetzt:

```
formel1[n1_, n2_, u_] := 2 (Binomial[n1 - 1, u - 1] Binomial[n2 - 1, u - 1]) / Binomial[n1 + n2, n1];
```

und die Formel 4.2.2 auf diese Weise:

```
formel2[n1_, n2_, u_] := ((Binomial[n1 - 1, u - 1] Binomial[n2 - 1, u]) +
                          (Binomial[n1 - 1, u] Binomial[n2 - 1, u - 1])) / Binomial[n1 + n2, n1];
```

Im Programm werden die beiden Formeln zu einer universalen Formel zusammengefasst.

Die Variable u ist in der Formel 4.2.1 (für gerade Anzahl von r) gleich r/2, und in der Formel 4.2.2 (für ungerade Anzahl von r) gleich (r-1)/2. Deswegen muss bei der Übergabe des Parameters u in die allgemeine Formel unterschieden werden.

```
u := If[r / 2 - Floor[r / 2] == 0, r / 2, (r - 1) / 2];
```

Die Option *Floor[x]* berechnet die abgerundete ganze Zahl von *x*. Das heißt: Ist r gerade Zahl, so muss $\frac{r}{2} - \text{Floor}[\frac{r}{2}]$ gleich Null sein. Ist r ungerade Zahl, so ist $\frac{r}{2} - \text{Floor}[\frac{r}{2}]$ gleich 0,5. Auf diese Weise kann man den richtigen Wert für u in die Formel übergeben.

5.3 Prüfen der Formeln

Wir wissen, dass r eine positive ganze Zahl zwischen 2 und n ist. Die Axiome der Wahrscheinlichkeit sagen uns unter anderem, dass

$$P(a < X \le b) = \sum_{j:a<x_j\le b} p_j = 1$$ ist (nach Formel 3.1.3 für diskrete Zufallsvariable).

Die Summe der einzelnen Wahrscheinlichkeiten für alle möglichen Werte von r muss also eine Eins ergeben, wenn die Formeln richtig sind.

Die Behauptung lautet : $\sum_{r=2}^{n} p_r = 1$**, wobei** p_r **= P(U) ist.**

Wir programmieren die Summe und stellen fest, dass die Behauptung wahr ist. Tatsächlich die Summe ergibt eine Eins (siehe die zwei letzten Zeilen des Outputs).

```
2  6.48119×10^-16  6.48119×10^-16
3  1.71752×10^-14  1.78233×10^-14
4  4.51091×10^-13  4.68914×10^-13
.
.
.
43  0.0000213486  0.999993
44  5.63041×10^-6  0.999998
45  1.4076×10^-6  1.
46  2.79194×10^-7  1.
47  5.46249×10^-8  1.
48  7.38887×10^-9  1.
49  1.07754×10^-9  1.
50  7.69674×10^-11  1.
51  7.69674×10^-12  1.
52  0  1.
53  0  1.
54  0  1.
55  0  1.

Die Summe der Wahrscheinlichkeiten fuer r von 2 bis n ist gleich:1.
```

Daraus folgt, die Formeln 4.2.1 und 4.2.2 sind richtig.

5.4 Erwartungswert , Varianz und Standardabweichung

Programmname

evs (siehe Notebook, Kapitel 7.1.2)

Programmbeschreibung

Wir können auch den Erwartungswert, die Varianz und die Standardabweichung für die Run-Verteilung einprogrammieren und bei Bedarf ausrechnen lassen.

```
(*Erwartungswert*)
erwartungswert[n1_, n2_] := (2 n1 n2 / (n1 + n2)) + 1;

(*Varianz*)
varianz[n1_, n2_] := (2 n1 n2 (2 n1 n2 - n1 - n2)) / ((n1 + n2) (n1 + n2) (n1 + n2 - 1));

(*Standardabweichung*)
standab = √varianz[n1, n2];
```

5.5 Graphische Darstellung der Run-Verteilung

Die wichtigsten Optionen für Plot, die in den Graphikprogrammen verwendet wurden:

Optionsname	**Funktion**
AxesLabel	Beschriftungen auf den Achsen, {xlabel,ylabel} spezifiziert Beschriftung für beide Achsen
AxesOrigin	der Punkt, an dem sich die Achsen kreuzen, wird bestimmt
Axes	ob Achsen zu zeichnen sind (mit ->True werden gezeichnet)
PlotStyle->RGBColor	bestimmt die Farbe vom Plot
PlotRange	der Koordinatenbereich, der im Plot erscheinen soll, All zeichnet alle Punkte
Ticks	bestimmt welche Skalenstriche zu zeichnen sind, wenn es Achsen gibt

Tabelle 5.1: Verwendete Optionen für Plot

In dieser Diplomarbeit wird die Run-Verteilung mit Balkendiagrammen und Treppenfunktionen dargestellt.

Die ersten sind nur für kleine Anzahl von Beobachtungen gut, denn ab etwa *n*=100 werden die Balkendiagramme zu unübersichtlich. Für große Werte von *n* sind die Treppenfunktionen gut geeignet, die für jedes *n* empfehlenswert sind.

5.5.1 Balkendiagramme für kleine *n*

Für kleine Werte von *n* kann man für die Run-Verteilung ein sehr schönes Balkendiagramm mit dem Kommando *BarChart* bekommen.

A. Balkendiagramm für die Wahrscheinlichkeitsfunktion

Programmname

balken_klein_n (siehe Notebook, Kapitel 7.1.3)

Programmbeschreibung

Für *BarChart* muss mit *Needs[„Graphics`Graphics`“]* das Graphik-Packages geladen werden.
Die Option *BarChart* erstellt ein Balkendiagramm für die paarweise gegebenen Daten. Wir brauchen also Zahlenpaare, die aus zwei Listen bestehen. Die erste Liste (mit dem Namen „wertex“) enthält die x-Werte: 2, . . . ,n. Die zweite Liste mit den y-Werten, den Wahrscheinlichkeiten für alle Werte von r (für $2 \leq r \leq n$), heißt „ywerte“.
Das Kommando *Transpose* transponiert aus den beiden Listen eine zweidimensionale Liste, die den Namen „daten“ bekommt. Für *BarChart* müssen die Zahlenpaare die x- und y-Werte in umgekehrter Reihenfolge als bei der Funktion bekommen. Jetzt kann das Balkendiagramm gezeichnet werden.

B. Balkendiagramm für die Verteilungsfunktion

Programmname

balkenverteil_klein_n (befindet sich auf dem beigefügten CD-ROM Rohling)

Programmbeschreibung

In diesem Programm müssen nur die y-Werte für das Balkendiagramm geändert werden, alles andere wird wie bei A programmiert. Die Liste der y-Werte enthält die entsprechenden Wahrscheinlichkeitssummen:

```
summe = 0;
ywerte = Table[summe = summe + formel, {r, 2, n}];
```

5.5.2 Treppenfunktionen für große *n*

Für alle Werte von *n* kann man in Mathematica-Programm eine Treppenfunktion erstellen. Dies ist besonders dann anzuwenden, wenn die „BarChart“-Graphik nicht mehr funktioniert, da die Anzahl der Objekte zu groß ist.

A. Treppenfunktion für die Wahrscheinlichkeitsfunktion

Programmname

treppe_gross_n (siehe Notebook, Kapitel 7.1.4)

Programmbeschreibung

Die Treppenfunktion wird mit dem Kommando *Line* erstellt, das eine Linie durch die gegebenen Punkte zeichnet.

Zu diesem Zweck werden zwei Listen mit x-Werten und y-Werten erzeugt.
Die erste Liste mit dem Namen „listex“ muss für die Treppenfunktion folgendermaßen aussehen: {1,1,2,2,3,3,4,4,...,n,n}.

Die zweite Liste mit dem Namen “wertey“ hat folgendes Muster:
{0, p_2,p_2,p_3,p_3,...,p_n,p_n,0}, wobei p_r die entsprechende Wahrscheinlichkeit P(r) bezeichnet. Damit der x-Wert genau unter der Mitte der einzelnen Stufe der Treppenfunktion liegt, wird zu jedem Wert der Liste „listex“ 0,5 addiert.
Mit der Option *Table* wird eine Liste mit den Zahlenpaaren erstellt. Die Linie durch die Punkte mit den Koordinaten (listex[[i]]+0.5, wertey[[i]], für i=1,..,2n) bekommt das Symbol „s“. Mit *Show[Graphics[s], Optionen für Plot]* wird die

Wahrscheinlichkeitsfunktion gezeichnet, wobei *Graphics* eine zweidimensionale graphische Abbildung repräsentiert.

B. Treppenfunktion für die Verteilungsfunktion

Programmname

verteil_gross_n (befindet sich auf dem beigefügten CD-ROM Rohling)

Programmbeschreibung

Für die Verteilungsfunktion müssen nur die y-Koordinaten geändert werden, alles andere wird wie bei A programmiert. Die Liste der y-Werte enthält nun die entsprechenden Wahrscheinlichkeitssummen.

5.5.3 Approximation der Run-Verteilung durch die Normalverteilung

Programmname

nv_rv_klein_n (siehe Notebook, Kapitel 7.1.5) und
nv_rv_gross_n (siehe Notebook, Kapitel 7.1.6)

Programmbeschreibung

In diesem Programm muss die Dichte der Normalverteilung mit dem Erwartungswert (Formel 4.2.3) und der Varianz (Formel 4.2.4) graphisch dargestellt werden.
Im ersten Teil des Programms steht die Erzeugung der Wahrscheinlichkeitsfunktionen der Run-Verteilung entsprechend für kleine oder große *n* (siehe Notebooks in 7.1.3 und 7.1.4).
Wir beginnen das Programmieren der Graphik zur Dichtefunktion mit *Needs[„Statistik`Master`“].* Mathematica kennt dann alle Statistik-Kommandos und lädt je nach Bedarf die entsprechenden Packages, sobald ein Kommando aus diesem Package verwendet wird.
Nachdem der Erwartungswert und die Varianz berechnet wurden, wird mit *NormalDistribution* [erwartungswert, $\sqrt{\text{varianz}}$] die gewünschte Normalverteilung berechnet. Dieses Zwischenergebnis bekommt den Namen „ndist“.
Folgende Option: *Plot* [*PDF* [ndist, x], {x, 2, r}] ermöglicht die graphische Darstellung der zugehörigen Dichtefunktion.

Da Plot der Normalverteilung automatisch bei 1 beginnt, wurde in die Liste der Zahlenpaare für das Balkendiagramm absichtlich ein zusätzliches Zahlenpaar {0,1} eingefügt. Sonst wäre die Kurve der Normalverteilung verschoben.

Mit dem Kommando *Show* können mehrere (in diesem Fall zwei) Graphik-Objekte kombiniert gezeigt werden.
Dieses Teilprogramm sieht wie folgt aus:

```
Needs["Statistics`Master`"]; (*laedt Packages für alle Statistik-Funktionen*)
mittelwert[n1_, n2_] := (2 n1 n2 / (n1 + n2)) + 1;
varianz[n1_, n2_] := (2 n1 n2 (2 n1 n2 - n1 - n2)) / ((n1 + n2) (n1 + n2) (n1 + n2 - 1));
ndist = NormalDistribution[mittelwert[n1, n2], √varianz[n1, n2]];
grenze = n;
nvrun = Plot[PDF[ndist, x] , {x, 2, grenze}, PlotRange -> All];
Show[balken, nvrun]
```

Hinweis für den Benutzer

Bei der großen Menge der eingefügten Notebooks zur graphischen Darstellung der Run-Verteilung wurde auf die Beschreibung der Programme zur Approximation der Verteilungsfunktionen durch die Normalverteilung bewusst verzichtet.
Man kann diese Programme ganz schnell selbst erzeugen, indem man die Notebooks zu den Verteilungsfunktionen (*balkenverteil_klein_n* und *verteil_gross_n* auf dem CD-ROM Rohling) mit dem oben beschriebenen Teilprogramm ergänzt. Man muss dabei nur eine Option ändern. Statt PDF wird CDF geschrieben. Auf diese Weise wird nicht die Dichtefunktion sondern die kumulierte Dichtefunktion berechnet.

5.6 Hilfsprogramme

In diesem Kapitel werden einige Programme vorgestellt, die trotz ihrer einfachen Form eine große Hilfe bei der oft sehr aufwendigen Vorbereitung der Daten zur Durchführung des Run-Tests sind.

Folgende Hilfsprogramme werden behandelt:

- Hilfsprogramm zur „runs up and down“-Methode
- Hilfsprogramm zur Methode der geraden und ungeraden Zahlen
- Hilfsprogramm zum Zählen von *n1, n2* und *rbeob* (ist Bestandteil der beiden ersten Programme)

5.6.1 Hilfsprogramm zur „runs up and down“-Methode

Programmname

runs_up_and_down_Test (siehe Notebook, Kapitel 7.1.7)

Dieses Programm wandelt jede Zahl aus der gegebenen Liste mit dem Namen „zuz“ (z.B. Liste der Zufallszahlen) in eine Null oder eine Eins nach der „runs up and down“-Methode (siehe Kapitel 4.5.1) um. Die erzeugten Zahlen werden gleich in eine neue Liste mit dem Namen „zuzpm“ geschrieben. Mit diesem Notebook kann man zwei Varianten von diesem Test durchführen. Nach jeder Testvariante werden die Variablen *n1, n2* und *rbeob* gezählt.

5.6.2 Hilfsprogramm zur Methode der geraden und ungeraden Zahlen

Programmname

gerade_ungerade_Test (siehe Notebook, Kapitel 7.1.8)

Programmbeschreibung

In diesem Notebook werden die Elemente einer Zahlenliste mit dem Namen „zuz“ in eine Null (bei geraden Zahlen) oder eine Eins (bei ungeraden Zahlen) geändert (siehe Kapitel 4.5.1). Die erzeugten Elemente stehen dann in einer neuen Liste mit dem Namen „guzuz“. Dann werden die beiden Merkmale und ihre Runs gezählt. Die erhaltenen Werte von *n1, n2* und *rbeob* können nun in das Programm zum Run-Test eingegeben werden.

5.6.3 Hilfsprogramm zum Zählen der Runs *rbeob* und der Anzahl *n1* und *n2*

Dieses Programm ist ein Teil der Programme *runs_up_and_down_Test* und *gerade_ungerade_Test.*

```
n = Length[zuz];

n1 = Count[zuzug, 1]; (*Die Einser werden gezählt*)
n2 = n - n1;          (*Die Nullen werden gezählt*)

(*Die Anzahl der beobachteten Runs wird gezählt.*)

rbeob = 1;
Do[If [zuzug[[i]] != zuzug[[i + 1]], rbeob = rbeob + 1], {i, 1, n - 1}];
```

5.7 Run-Test im Programm

5.7.1 Erläuterung zum Test

Das Programm kann die exakte Irrtumswahrscheinlichkeit α der Run-Verteilung für eine unbegrenzte Anzahl von Objekten ausrechnen. Für einen problemlosen Ablauf des Programms kann der Benutzer allerdings einen Zeitlimit setzen. Der komplette Programmlisting zum Run-Test steht im Anhang im Kapitel 7.1.9.

Damit man das Programm ohne Probleme mehrmals hintereinander laufen lassen kann, wird die Option *Clear* verwendet. Mit ihrer Hilfe werden alle alten Werte der Variablen gelöscht, die Variablenname selbst bleiben allerdings weiterhin erhalten.

In unserem Programm sieht das wie folgt aus:

```
Clear[n1, n2, rbeob, alpha, testwahl, zeitlimit];
(*löscht die Zuordnung von Werten zu den genannten Symbolen*)
```

Hinweise für den Benutzer

Wartezeit

Nachdem das Programm gestartet wurde, steht, während die Berechnung von alpha läuft, im Rahmen des aktiven Fensters „Running“. Man soll keinen neuen Start auslösen, bevor die Berechnung fertig ist und die Ausgabe von alpha, bzw. eine Fehlermeldung auf dem Bildschirm erscheint.

Der Start

Zum Start des Programms müssen die Tasten [Shift]+[Return] gleichzeitig gedrückt werden, das Programm wird dann geladen und gestartet.

5.7.2 Programmaufbau

Das Programm wurde wie folgt aufgebaut:

S T A R T
Eingabe: Parameter mit Sicherheitsabfragen zeitlimit, n1, n2, rbeob, NS testwahl *(Kapitel 5.7.3.1)*
Berechnungen für alle Testarten: Formeln 4.2.1 und 4.2.2 *(Kapitel 5.7.3.2)*
Berechnung je nach Testart als Unterprogramm: Testart 1 - berechnungzwei Testart 2 - berechnungeinslinks Testart 3 - berechnungeinsrechts *(Kapitel 5.7.3.3)*
Run-Test für jede Testart als Unterprogramm: Testart 1 - testzwei Testart 2 - testeinslinks Testart 3 - testeinsrechts *(Kapitel 5.7.3.4)*
Allgemeine Ausgabe der eingegebenen Parameter als Unterprogramm ausgabe *(Kapitel 5.7.3.5)*
Ausgabe von alpha für Testart 1 - Unterprogramm: ausgabezwei Testart 2 - Unterprogramm: ausgabeeinslinks Testart 3 - Unterprogramm: ausgabeeinsrechts *(Kapitel 5.7.3.6)*
Aufruf der Unterprogramme in folgender Reihenfolge: Berechnung, Test und Ausgabe *(Kapitel 5.7.3.7)*

Tabelle 5.2: Ablauf vom Run-Test-Programm

5.7.3 Erklärung aller Programmteile

In diesem Kapitel werden die einzelnen Programmteile ausführlich beschrieben (siehe Tabelle 5.2).

In der Tabelle 5.3 werden alle Variablen erklärt, die für die Berechnung, sowie den problemlosen Ablauf des Programms notwendig sind. Die Eingabevariablen werden separat in der Tabelle 5.4 aufgeführt.

Variable	Erläuterung
alpha	*Irrtumswahrscheinlichkeit* α *, der einzige Wert für die Ausgabe*
n	Gesamtanzahl der Objekte n = n1 + n2
r	Anzahl aller möglichen Runs
u	r/2 falls r gerade, (r-1)/2 falls r ungerade (siehe Hauptformeln)
summe	Summe der Wahrscheinlichkeiten P(U)
m	Erwartungswert der Run-Verteilung (siehe Formel 4.2.3)

Tabelle 5.3: Verwendete Variablen im Run-Test-Programm

5.7.3.1 Eingabeteil, Variableneinstellung mit Sicherheitsabfragen

Der Eingabeteil ist eine Oberfläche zur Dateneingabe. Es muss hier nur auf die angegebenen Variablen zugegriffen werden, damit das Programm weiterhin funktioniert. In der folgenden Tabelle werden alle Variablen erklärt. Zu jeder Variable ist angegeben, unter welcher Bedingung sie eingesetzt werden muss, welche Funktion sie hat und welche Werte erlaubt sind.

Variable	Bedingung	Funktion	erlaubte Werte
zeitlimit	immer	der vorgegebene Zeitlimit in Sekunden für den Lauf des Programms	positive ganze Zahlen
n1	immer	Anzahl der Objekte vom Typ 1	positive ganze Zahlen
n2	immer	Anzahl der Objekte vom Typ 2	positive ganze Zahlen
rbeob	immer	beobachtete Anzahl von Runs	positive ganze Zahlen zwischen 2 und n
NS	immer	Anzahl der Nachkommastellen in der Ausgabe	positive ganze Zahlen
testwahl	immer	gibt an, welcher Test gemacht werden soll	1 - Zweiseitiger Test 2 - Einseitiger Test links 3 - Einseitiger Test rechts

Tabelle 5.4: Variablen für die Eingabe im Run-Test-Programm

Sicherheitsabfragen für die Eingabe von *rbeob* und *testwahl*

Die eingegebenen Daten werden durch zwei Sicherheitsabfragen geprüft und es wird zu einer eventuellen Neueingabe aufgerufen.

Zuerst wird abgecheckt, ob der eingegebene Wert vom *rbeob* sich im zugelassenen Intervall befindet, da die Anzahl von Runs eine Zahl zwischen 2 und n ist (siehe Einleitung). Danach wird geprüft, ob die eingegebene Zahl für die Testwahl eine von drei zugelassenen Werten 1, 2 oder 3 ist.

Variablen *n1, n2* und *rbeob*

Sie sind die wichtigsten Variablen für den Run-Test, denn mit diesen drei Parametern werden die Berechnungen für die Irrtumswahrscheinlichkeit durchgeführt. Diese Werte können aus den gegebenen Daten mit verschiedenen Testmethoden gewonnen werden (siehe Kapitel 4.5).

Zeitlimit

Bei sehr großen Werten von *n1* und *n2* läuft das Programm lange. Der Programmbenutzer darf für den Lauf des Programms einen Zeitlimit setzen. Die Variable *zeitlimit* bestimmt, wie lange die Berechnung der Wahrscheinlichkeitssummen dauern soll. Da die Erstellung der Liste mit den einzelnen Wahrscheinlichkeitssummen die zeitaufwendigste Berechnung von allen ist, und der restliche Programmablauf danach sehr schnell geht, wird mit dem *Zeitlimit* eigentlich die Dauer des Programmlaufes bestimmt.

Für die Zeitbeschränkung wurde die Option *TimeConstrained* gewählt.
TimeConstrained[ausdruck, t, fehlausdruck] liefert Fehlausdruck (hier Null) zurück, wenn die Zeitbeschränkung t (hier zeitlimit) nicht eingehalten wird.
Im Programm bezieht sich dieses Kommando auf das Schreiben der Liste *sumliste*:

```
sumliste = TimeConstrained[Table[summe = summe + formel, {r, 2, rbeob}], zeitlimit, Null]
```

Bekommt die Liste *sumliste* den Fehlausdruck Null,

```
If[ sumliste == Null, zuvielzeit[]];
```

weil die vorgegebene Zeitbeschränkung überschritten wird, wird das Unterprogramm *zuvielzeit* aufgerufen:

```
zuvielzeit[] := ( Print[ "Die vorgegebene Zeit von ", zeitlimit,
  " Sekunden wurde überschritten. Bitte waehlen Sie kleinere Werte fuer n1 und n2 !"];
   Abort[];);
```

Auf dem Bildschirm kommt die Fehlermeldung und das Programm wird mit der Funktion *Abort* unterbrochen.

Eingabe der Nachkommastellen für die Ausgabe von *alpha*

α wird mit der gewünschten Anzahl der Nachkommastellen ausgegeben. Dies ermöglicht die Option *PaddedForm,* die Zahl *alpha* mit Platz für *NS+3* Ziffern und genau *NS* Ziffern rechts vom Dezimalpunkt druckt.

```
PaddedForm[100 alpha, {NS + 3, NS}]
```

Wahl der Testart

Mit der Eingabe einer Zahl 1, 2 oder 3 für die Variable *testwahl* kann man die Art des Run-Tests bestimmen.

1 - zweiseitiger Run-Test
2 - einseitiger Run-Test links
3 - einseitiger Run-Test rechts

5.7.3.2 Allgemeine Berechnungen für alle Testarten

In diesem Kapitel werden die Grundformeln für die Berechnung von *alpha* definiert. Hier werden die verzögerten Zuweisungsoperatoren verwendet (siehe Kapitel 5.1.2). Die ausführlichen Erklärungen zu dieser Berechnung findet man im Kapitel 5.2.

5.7.3.3 Berechnungen speziell für jede Testart

Unterprogramm für den zweiseitigen Test

```
berechnungzwei[] := (
  m = (2 n1 n2 / (n1 + n2)) + 1; (*Erwartungswert Formel 4.2.3 wird berechnet*)

  If[rbeob < m,
   sumliste = TimeConstrained[Table[summe = summe + formel, {r, 2, rbeob}], zeitlimit, Null],
   sumliste = TimeConstrained[Table[summe = summe + formel, {r, rbeob, n}], zeitlimit, Null]
    ];

  z = Table[formel, {r, rbeob, rbeob}];(*Hier wird die Wahrscheinlichkeit für r=rbeob
                                        berechnet, für den Fall dass alpha > 0.5*)
                    );
```

Zuerst wird der Erwartungswert nach der Formel 4.2.3 berechnet und mit dem Symbol *m* bezeichnet.
Falls der eingegebene Wert von *rbeob* kleiner als *m* ist, werden folgende Wahrscheinlichkeitssummen $\sum_{r=2}^{rbeob} P(U = r)$ für jedes r=2, . . . , rbeob ermittelt.

Ist *rbeob* größer gleich dem Erwartungswert, dann werden die Werte von P(U) auf der anderen Seite zusammengezählt: $\sum_{r=rbeob}^{n} P(U = r)$ für jedes r= rbeob, . . . , n.

Diese Wahrscheinlichkeitssummen, die im Programm mit dem Symbol *summe* bezeichnet sind, werden gleich in eine Liste mit dem Namen *sumliste* geschrieben. Die Liste für rbeob<m besteht aus (rbeob-1) Elementen und für rbeob $\geq$ m aus den (n-rbeob+1) Elementen.

Für eventuell weitere Berechnungen (siehe Erklärung in 5.7.3.4) wird eine Liste *z* mit einem Element P(U=rbeob) erzeugt.

Unterprogramm für den linksseitigen Test

```
(*Berechnungen für den linksseitigen Run-Test*)

   (*Hier werden nur die vorderen Balken
  (für r von 2 bis rbeob) aufsummiert und in die Liste "sumliste" geschrieben*)

   berechnungeinslinks[] := (
   sumliste = TimeConstrained[Table[summe = summe+ formel, {r, 2, rbeob}], zeitlimit, Null];);
```

Hier wird die Liste *sumliste* mit den Elementen $\sum_{r=2}^{rbeob} P(U = r)$ für jedes r=2, . . . , rbeob definiert.

Unterprogramm für den rechtsseitigen Test

```
(*Berechnungen für den rechtsseitigen Run-Test*)

   (*Hier werden nur die hinteren Balken
  (für r von rbeob bis n) aufsummiert und in die Liste "sumliste" geschrieben*)

   berechnungeinsrechts[] := (
   sumliste = TimeConstrained[Table[summe = summe+ formel, {r, rbeob, n}], zeitlimit, Null];);
```

In diesem Fall haben die einzelnen Elemente von *sumliste* folgende Gestalt:

$\sum_{r=rbeob}^{n} P(U = r)$ für jedes r = rbeob, . . . , n.

5.7.3.4 Run-Test speziell für jede Testart

Zweiseitiger Run-Test

Das Unterprogramm *testzwei* führt den zweiseitigen Run-Test durch. Das sieht wie folgt aus:

```
(*DER ZWEISEITIGE RUN-TEST*)

  (*Nach den Formeln 4.4 .1 und 4.4 .2*)

  testzwei[] := ( If[rbeob < m,
  alpha = (sumliste[[rbeob - 1]]), (*alpha = das letzte Element von sumliste*)
  alpha = (sumliste[[n - rbeob + 1]])];

  (*Weil man nicht genau weiß,
   bei welchem r die Summe der Balken über 0.5 ist -> Sicherheitsabfrage*)
  (*nach der Formel 4.4 .1 (P) und 4.4 .2 (P)*)
   If[alpha >= 0.5, alpha = 1 - alpha + z[[1]]];

  (**Falls alpha immer noch größer gleich 0.5 ist, dann ist alpha = 0,5
                                                    (siehe Formel 4.4 .3)*)
  If[alpha >= 0.5, alpha = 0.5];);
```

Zuerst wird alpha ($=\alpha/2$) gemäß der Formel 4.4.1 (für rbeob<m) bzw. gemäß der Formel 4.4.2 (für rbeob $\geq$ m) ermittelt. Zu diesem Zweck reicht es das letzte Element aus der Liste *sumliste* herauszugreifen und gleich dem Ausdruck *alpha* zuzuweisen. An dieser Stelle verbirgt sich hinter *alpha* das vorläufige Ergebnis der zu bestimmenden Irrtumswahrscheinlichkeit.

Dann kommt die If-Sicherheitsbfrage für den Fall 2 A bzw. 2 B (siehe Kapitel 4.4.1). Ist die Bedingung alpha $\geq$ 0.5 wahr, wird alpha nach der Formel 4.4.2(P) oder 4.4.1(P) berechnet, je nachdem, ob *rbeob* kleiner oder größer gleich dem Erwartungswert *m* ist.

Mit dem Ausdruck: alpha = 1 – alpha + z[[1]] werden beide Formeln (Formeln 4.4.1(P) und 4.4.2(P)) abgefangen, wobei mit z[[1]] auf das erste (einzige) Element der Liste „z“ zugegriffen wird (siehe Kapitel 5.7.3.3).

Mit der zweiten If-Abfrage wird der Fall 3 A bzw. 3 B (siehe Kapitel 4.4.1) behandelt. Der letzte Wert von *alpha* aus diesem Test wird im Unterprogramm *ausgabezwei* dem Endergebnis zugewiesen.

Einseitiger Run-Test

Das Unterprogramm *testeinslinks* führt den linksseitigen Run-Test durch. Das sieht wie folgt aus:

```
(*DER LINKSSEITIGE RUN-TEST*)

testeinslinks[] := (alpha = sumliste[[rbeob - 1]];);     (*nach Formel 4.4 .4*)
```

Hier wird dem Ausdruck *alpha* das letzte Element aus *sumliste* zugewiesen (laut der Formel 4.4.4).

Das Unterprogramm *testeinsrechts* führt den rechtsseitigen Run-Test durch. Das sieht folgendermaßen aus:

```
(*DER RECHTSSEITIGE RUN-TEST*)

testeinsrechts[] := (alpha = sumliste[[n - rbeob + 1]];); (*nach Formel 4.4 .5*)
```

Bei diesem Test wird dem Symbol *alpha* das letzte Element aus *sumliste* zugewiesen (laut der Formel 4.4.5).

Es wird hingewiesen, dass obwohl dem Ausdruck *alpha* bei jeder Testart jeweils das letzte Element der Liste *sumliste* zugewiesen wird, bedeutet dieses Element in jedem der drei Fällen was anderes (siehe Erklärung zur Summenberechnung in 5.7.3.3).

5.7.3.5 Allgemeine Ausgabe der eingegebenen Parameter

Dieses Unterprogramm ermöglicht die Ausgabe der eingegebenen Variablen *n1*, *n2* und *rbeob*. Dies ist das erste Teil der Bildschirmausgabe im Run-Test.

Hierzu der Ausschnitt aus dem Notebook:

```
(**********************Allgemeine A U S G A B E************************)

      ausgabe[] := (Print[""];
      Print["Anzahl der Beobachtungen vom Typ 1, n1 = ", n1];
      Print["Anzahl der Beobachtungen vom Typ 2, n2 = ", n2];
      Print["Gesamtanzahl aller Beobachtungen,    n = ", n];
      Print["Anzahl der beobachteten Runs,    rbeob = ", rbeob];
      Print["2 ≤ r ≤ ", n];
      Print[""];
                  );
```

5.7.3.6 Ausgabe der Irrtumswahrscheinlichkeit

Bei jeder Testart wird andere Ausgabe der Irrtumswahrscheinlichkeit aufgerufen. Die Variable *alpha* wird in % und mit drei Stellen nach dem Dezimalpunkt ausgegeben. Im zweiseitigen Test muss der letzte Wert von *alpha* zusätzlich mit zwei multipliziert werden. Sonst sehen sich die drei Ausgaben ähnlich:

```
ausgabezwei[] := (Print["Zweiseitiger Run-Test"];
  Print["Mit der Irrtumswahrscheinlichkeit alpha =", PaddedForm[200 alpha, {NS+3, NS}],
  " % wird Ho abgelehnt zugunsten der Alternativhypothese."];);


ausgabeeinslinks[] := (Print["Einseitiger Test links"];
  Print["Mit einer Irrtumswahrscheinlichkeit alpha =",  PaddedForm[100 alpha, {NS+3, NS}],
  " % wird Ho abgelehnt zugunsten der Alternativhypothese."];);


ausgabeeinsrechts[] := (Print["Einseitiger Test rechts"];
  Print["Mit einer Irrtumswahrscheinlichkeit alpha =", PaddedForm[100 alpha, {NS+3, NS}],
  " % wird Ho abgelehnt zugunsten der Alternativhypothese."];);
```

5.7.3.7 Aufruf der Unterprogramme

Je nach Testart werden bestimmte Unterprogramme aufgerufen.

```
If[testwahl == 1, ( berechnungzwei[]; testzwei[]; ausgabezwei[];),

    If[testwahl == 2, (berechnungeinslinks[]; testeinslinks[]; ausgabeeinslinks[];),

                      (berechnungeinsrechts[]; testeinsrechts[]; ausgabeeinsrechts[];)

      ]; (*ende von else von If testwahl=2*)
  ]; (*ende von else von If testwahl=1*)
```

5.7.4 Output des Programms

Ein Exempel des Outputs:

```
Anzahl der Beobachtungen vom Typ 1, n1 = 68
Anzahl der Beobachtungen vom Typ 2, n2 = 54
Gesamtanzahl aller Beobachtungen,    n = 122
Anzahl der beobachteten Runs,    rbeob = 43
2 ≤ r ≤ 122

Zweiseitiger Run-Test
Mit der Irrtumswahrscheinlichkeit alpha =   0.105 %
  wird die Nullhypothese Ho abgelehnt zugunsten der Alternativhypothese.
```

6 Anwendung des Run-Tests

6.1 Einführende Beispiele

Wir gehen hier auf die Beispiele aus der Einleitung zurück.

6.1.1 Kartenspiel

Daten

Die Anzahl der roten und schwarzen Karten ist jedesmal gleich, nur die Anzahl der Runs ändert sich.

1. Fall

S S S R S R R R S S R S R S R R R S S R S S S R R S S S S R R R

$n_1 = 16,\ n_2 = 16,\ rbeob = 16$

2. Fall

S S S S S S S S S S S S S S R R R R R R R R R R R R R R R R

$n_1 = 16,\ n_2 = 16,\ rbeob = 2$

3. Fall

S R S R S R S R S R S R S R S R S R S R S R S R S R S R S R S R S R S R

$n_1 = 16,\ n_2 = 16,\ rbeob = 32$

Hypothesen

H_0 : Die Reihenfolge der Karten ist zufällig.

H_1 : Die Reihenfolge der Karten ist nicht zufällig.

Test

Wir geben die drei Variablen in das Programm zum zweiseitigen Run-Test ein und bekommen folgendes Resultat:

1. Fall

Zweiseitiger Run-Test
Anzahl der Beobachtungen vom Typ 1, n1 = 16
Anzahl der Beobachtungen vom Typ 2, n2 = 16
Gesamtanzahl aller Beobachtungen, n = 32
Anzahl der beobachteten Runs, rbeob = 16
$2 \leq r \leq 32$

Mit der Irrtumswahrscheinlichkeit alpha = 86.222 % wird Ho abgelehnt.

2. Fall

Mit der Irrtumswahrscheinlichkeit alpha = 6.655×10^{-7} % wird Ho abgelehnt.

3. Fall

Mit der Irrtumswahrscheinlichkeit alpha = 6.655×10^{-7} % wird Ho abgelehnt.

Schlussfolgerung:

Im ersten Fall ist die Irrtumswahrscheinlichkeit α gleich 86,222 %. Das heißt, die Reihenfolge der Karten ist zufällig.
Beim zweiten und dritten Mal ist die Irrtumswahrscheinlichkeit sehr klein, α beträgt 6,655 x 10^{-7} %. Die beiden Kartenfolgen sind nicht zufällig. Die Karten wurden entweder nicht gut gemischt oder es liegt hier eine Manipulation vor.

Prüfung auf Normalverteilung:

Die Anzahl der Beobachtungen *n* ist in diesem Experiment sehr klein, das zugehörige Balkendiagramm und die zugehörige Dichtefunktion der Normalverteilung mit μ_r und σ_r ist in Abbildung 6.1 ersichtlich.

Anhand der Graphik sehen wir, dass bei einer kleinen Anzahl von Beobachtungen die Näherung durch die Normalverteilung sehr schlecht ist.
Die Abbildung 6.1 zeigt, dass die übliche Approximation mit der Normalverteilung (besonders für kleine *n*) zu falschen Ergebnissen führen kann.

Prüfung auf Normalverteilung (graphisch)

Wahrscheinlichkeitsfunktion

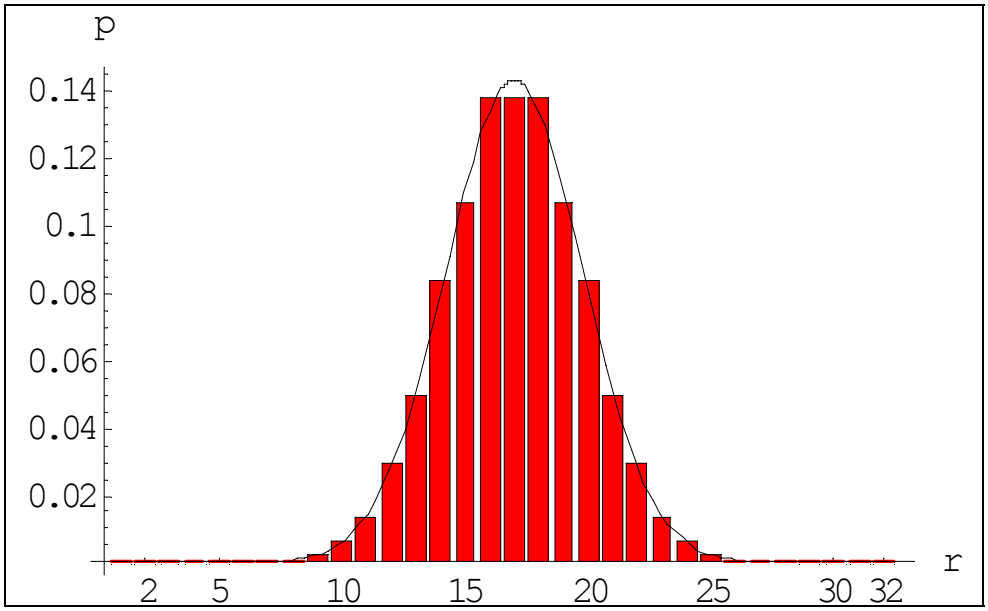

Abbildung 6.1: Approximation der Wahrscheinlichkeitsfunktion der Run-Verteilung durch die Normalverteilung, n1 = n2 = 16, n = 32

Die Abbildung 6.2 zeigt die Angleichung der Run-Verteilungsfunktion durch die Verteilungsfunktion der zugehörigen Normalverteilung.

Verteilungsfunktion

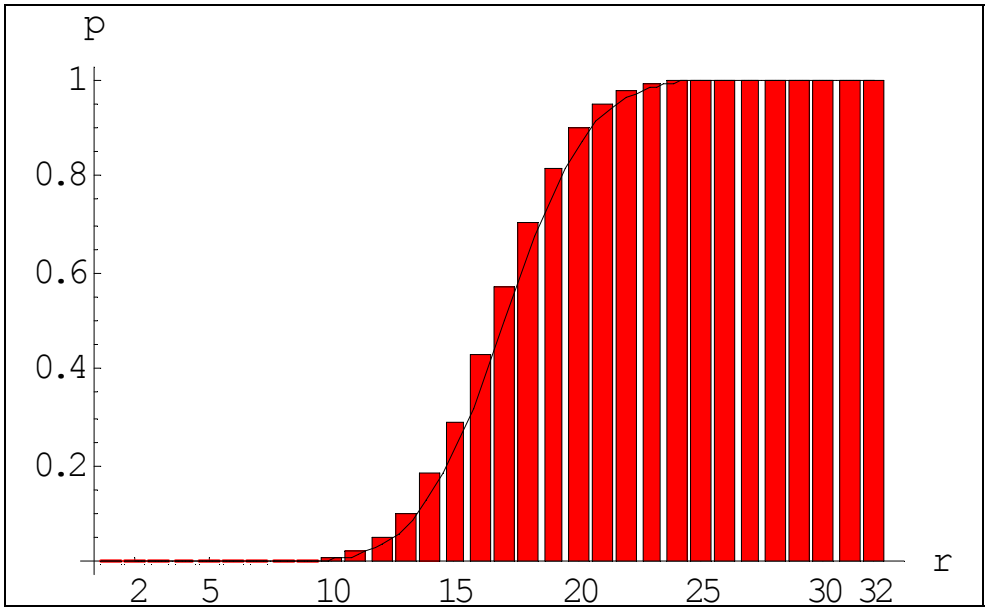

Abbildung 6.2: Approximation der Verteilungsfunktionen der Run-Verteilung und der Normalverteilung, n1 = n2 = 16, n = 32

6.1.2 Schüler vor dem Getränkeautomaten

Daten

Achtzehn Schüler einer Grundschule, davon $n_1 = 8$ Jungs und $n_2 = 10$ Mädchen stellen sich in eine Schlange vor dem Getränkeautomaten an, und zwar in folgender Reihenfolge bezüglich des Geschlechts (J bedeutet Jungen, M Mädchen):

J J J M M M M J J M M M M M J J J M

$n_1 = 8$, $n_2 = 10$,
n = 18, rbeob = 6

Nullhypothese

Die Gruppierung der Kinder ist zufällig bezüglich des Geschlechts.

Alternativhypothese

Die Gruppierung der Kinder ist geschlechtshomogen.

Unter dieser Alternative sind also wenige Runs zu erwarten (im Extremfall nur 2).

(Bei Jugendlichen müsste realistischer Weise die andere einseitige Alternative angenommen werden, dass die Gruppierung geschlechtsinhomogen ist, in diesem Fall wären viele Runs zu erwarten).

Test

Wir geben die Variablen in das Programm zum einseitigen Run-Test links ein und bekommen folgendes Resultat:

```
Einseitiger Test
Die Alternativhypothese lautet:
Gleiche Merkmalswerte treten zu haeufig hintereinander auf, das heisst es gibt zu wenig
```

```
Mit einer Irrtumswahrscheinlichkeit alpha =   4.792 % wird Ho abgelehnt.
```

Schlussfolgerung

Da α nur 4,792 % beträgt, können wir sagen, dass die Reihenfolge der Kinder bezüglich des Geschlechts nicht zufällig ist. Die Nullhypothese wird abgelehnt zugunsten der gestellten Alternativhypothese. Das heißt, die Gruppierung der Kinder ist geschlechtshomogen.

Wahrscheinlichkeitsfunktion für n=18, n1=8, n2=10

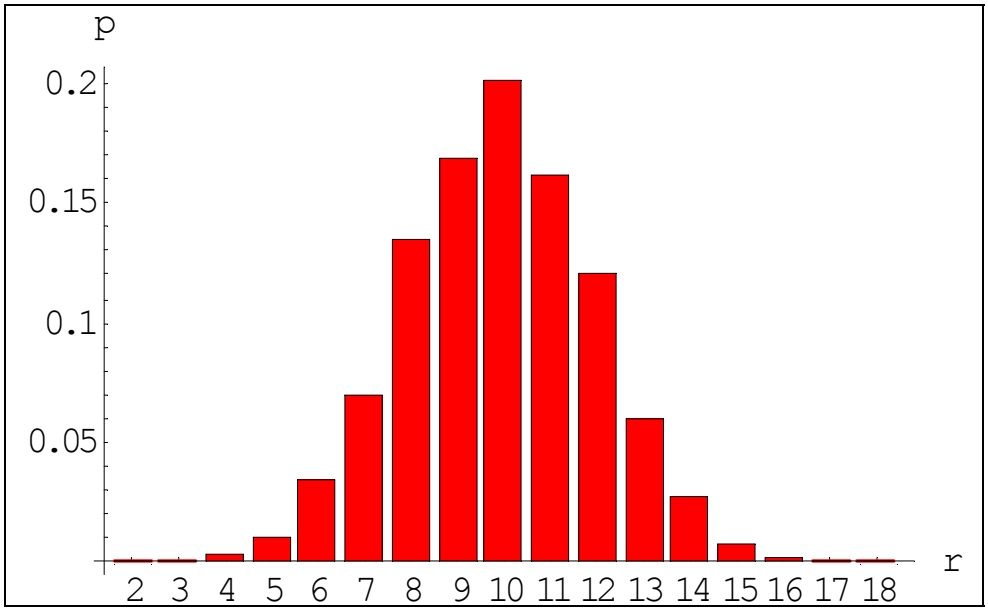

Abbildung 6.3: Balkendiagramm der Run-Verteilung, n1=8, n2=10, n=18

In der nächsten Abbildung sehen wir, dass α für rbeob=6 tatsächlich unter 5 % liegt. Nach der Formel 4.4.4 zeigen nämlich die Balken der Verteilungsfunktion im Falle eines einseitigen Tests genau die Irrtumswahrscheinlichkeiten für den gegebenen Wert von rbeob.

Verteilungsfunktion für n=18, n1=8, n2=10

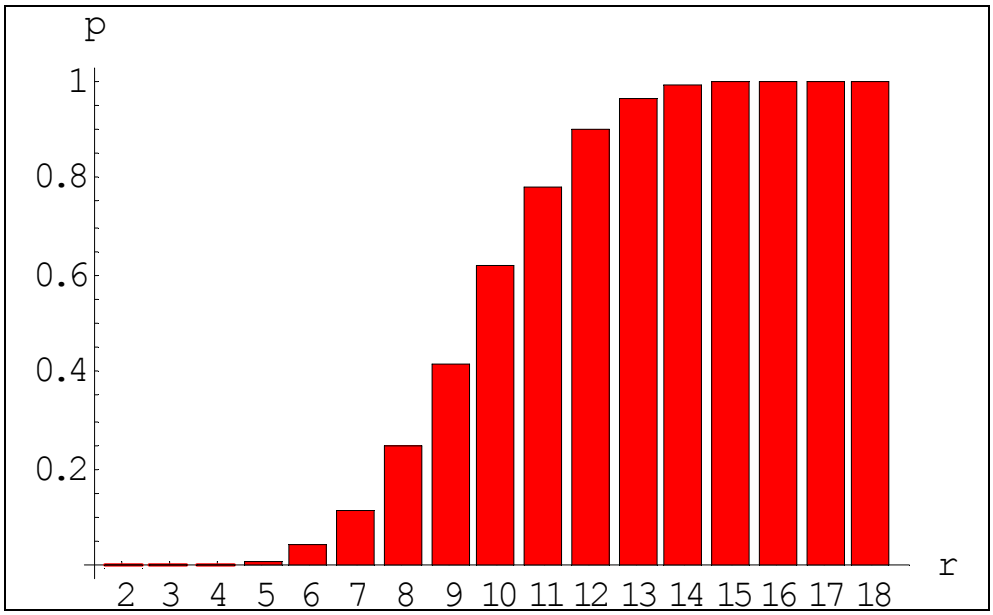

Abbildung 6.4: Verteilungsfunktion der Run-Verteilung, n1=8, n2=10, n=18.

6.2 Treibstoffpreise

In den Kapiteln 6.2, 6.3 und 6.4 wird der Test, der auf der Theorie der „runs up and down“ basiert, durchgeführt. Mit dem zweiseitigen Run-Test wird die nach der „runs up and down“ Methode entstandene Zahlenfolge auf Zufälligkeit geprüft.

Daten

Benzinpreise von Januar 1999 bis März 2001, das sind 27 Monate, *n* ist 27.

Folgende Graphik zeigt die Preisschwankungen der Treibstoffpreise in diesem Zeitintervall.

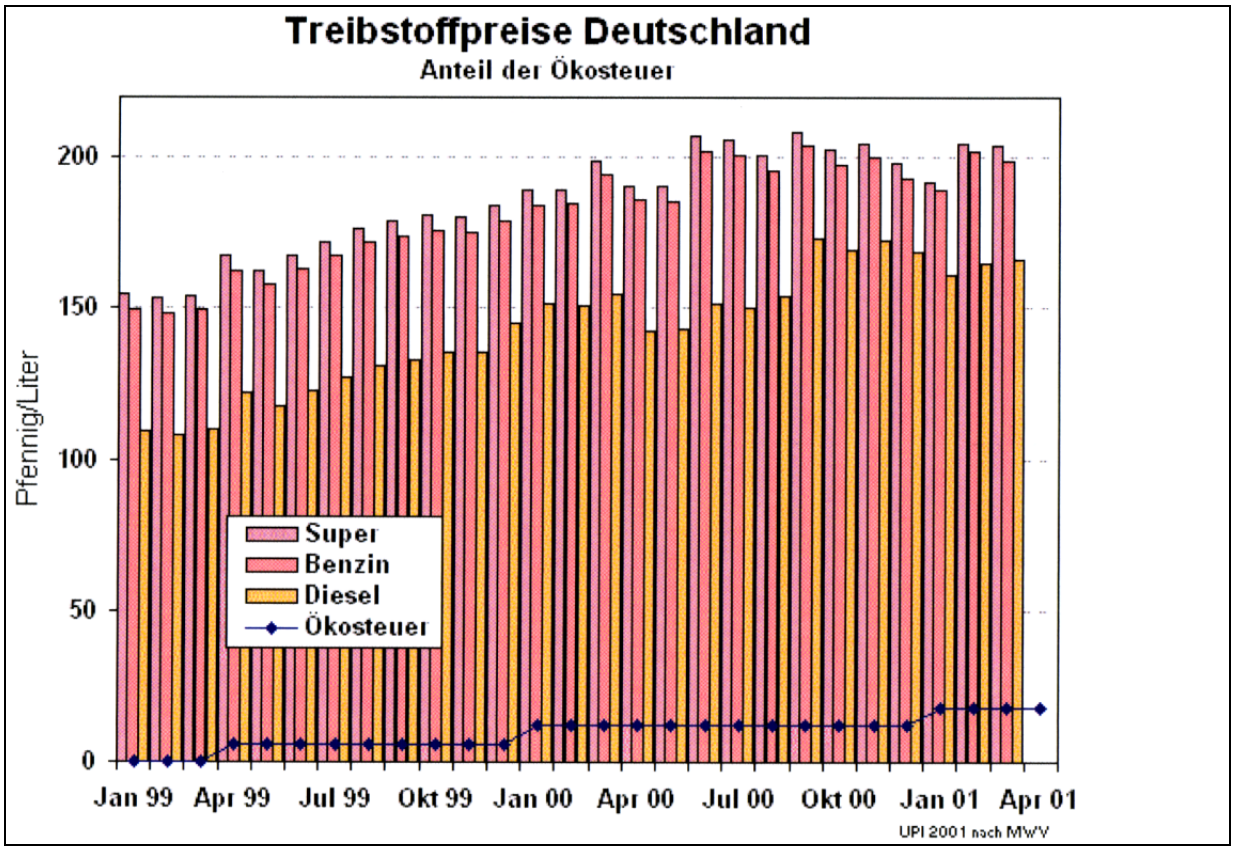

Abbildung 6.5: Schwankungen der Treibstoffpreise Januar 1999 - Juni 2001

Nullhypothese

Die Schwankungen der Treibstoffpreise sind zufällig.

Alternativhypothese

Die Schwankungen der Treibstoffpreise sind nicht zufällig.

TESTDURCHFÜHRUNG

Der erste Schritt

Da Super und Normalbenzin gleichmäßig steigen und fallen, fassen wir diese Daten zusammen. Diesel wird extra getestet.

Der zweite Schritt

Wir schreiben die Schwankungen als Folgen von Nullen und Einsern auf. Dies geschieht nach der „Runs up and down“-Methode (siehe Kapitel 4.5.1).

Bei Super und Benzin entstehen folgende Reihen:

Test 1

1 0 1 1 0 1 1 1 1 1 0 1 1 0 1 0 0 1 0 0 1 0 1 0 0 1 0

Anzahl der Einser n1 = 15
Anzahl der Nullen n2 = 12

Es gibt rbeob = 18 Runs.

Test 2

1 0 1 1 0 1 1 1 1 1 0 1 1 1 1 0 1 1 0 0 1 0 1 0 0 1 0

Anzahl der Einser n1 = 17
Anzahl der Nullen n2 = 10

Es gibt rbeob= 16 Runs.

Bei Diesel entstehen folgende Reihen:

Test 1

1 0 1 1 0 1 1 1 1 1 0 1 1 0 1 0 1 1 0 1 1 0 1 0 0 1 1

Anzahl der Einser n1 = 18
Anzahl der Nullen n2 = 9

Es gibt rbeob = 17 Runs.

Test 2

1 0 1 1 0 1 1 1 1 1 1 1 0 1 0 1 1 0 1 1 0 1 0 0 1 1

Anzahl der Einser n1 = 19
Anzahl der Nullen n2 = 8

Es gibt r = 15 Runs.

Der dritte Schritt

Wir wenden den zweiseitigen Run-Test an und bekommen folgende Irrtumswahrscheinlichkeiten:

Die folgende Tabelle zeigt die Variablen und Ergebnisse der beiden Tests.

	Super + Benzin	Super + Benzin	Diesel	Diesel
Testart	Test 1	Test 2	Test 1	Test 2
n1	15	17	18	19
n2	12	10	9	8
rbeob	18	16	17	15
α	**20,843 %**	**41,922 %**	**12,241 %**	**31,417 %**

Tabelle 6.1: Ergebnisse aus dem Run-Test für Treibstoffpreise, n=27

Entscheidung

Die Nullhypothese wird mit der Irrtumswahrscheinlichkeit α (siehe Tabelle 6.1) zugunsten der Alternativhypothese abgelehnt.
Da die Werte von alpha hoch sind, ist es anzunehmen, dass die Schwankungen der Treibstoffpreise zufällig sind.

Prüfung auf Normalverteilung (rechnerisch)

Die folgende Tabelle vergleicht die exakten Ergebnisse von *alpha* aus unserem Test mit den approximierten Werten der zugehörigen Normalverteilung.

Erklärung zu der Tabelle 6.2

Der Erwartungswert wurde nach der Formel 4.2.3 berechnet (mit Hilfe des Programms *evs*).
Die exakte Irrtumswahrscheinlichkeit α wurde mit dem zweiseitigen Run-Test berechnet.
Die approximierte Irrtumswahrscheinlichkeit α wurde mit Hilfe der Normalverteilung mit Erwartungswert und Varianz der Run-Verteilung (siehe Formeln 4.2.3 und 4.2.4) bestimmt.

Der absolute Fehler ist die Differenz: α exakt - α approximiert.
Der relativer Fehler ist gleich: absoluter Fehler / α exakt.

Die letzten vier Werte der Tabelle wurden in % angegeben.

Nummer	n1	n2	rbeob	Erwartungswert	α exakt	α approx.	absoluter Fehler	relativer Fehler
1	15	12	18	14,333	20,843	14,4849	6,3581	0,3050
2	17	10	16	13,593	41,922	30,9635	10,9585	0,2614
3	18	9	17	13,000	12,241	7,5865	4,6545	0,3802
4	19	8	15	12,259	31,417	19,3500	12,0670	0,3841

Tabelle 6.2: Vergleich der Ergebnisse der Run-Verteilung mit der Normalverteilung, Run-Test für Treibstoffpreise, n=27

Wenn wir die Irrtumswahrscheinlichkeiten vergleichen, stellen wir fest, dass die Unterschiede bei diesen kleinen *n* beträchtlich sind. Der absolute Fehler liegt hier zwischen 4 % und 12 %.

Betrachtet man besonders die Position 3 aus der Tabelle, so sieht man, dass die Approximation durch die Normalverteilung zu falschen Ergebnissen führen kann.

Die exakte Irrtumswahrscheinlichkeit α gleich 12,241 % könnte man zum Beispiel noch akzeptieren, aber die approximierte Irrtumswahrscheinlichkeit von 7,5865 % nicht mehr. Das heißt, der approximierte Wert von alpha könnte sogar zur Annahme der gegensätzlichen Hypothese führen.

Für größere Werte von *n* ist der absolute Fehler entsprechend kleiner (siehe Tabelle 6.6).

Prüfung auf Normalverteilung (graphisch)

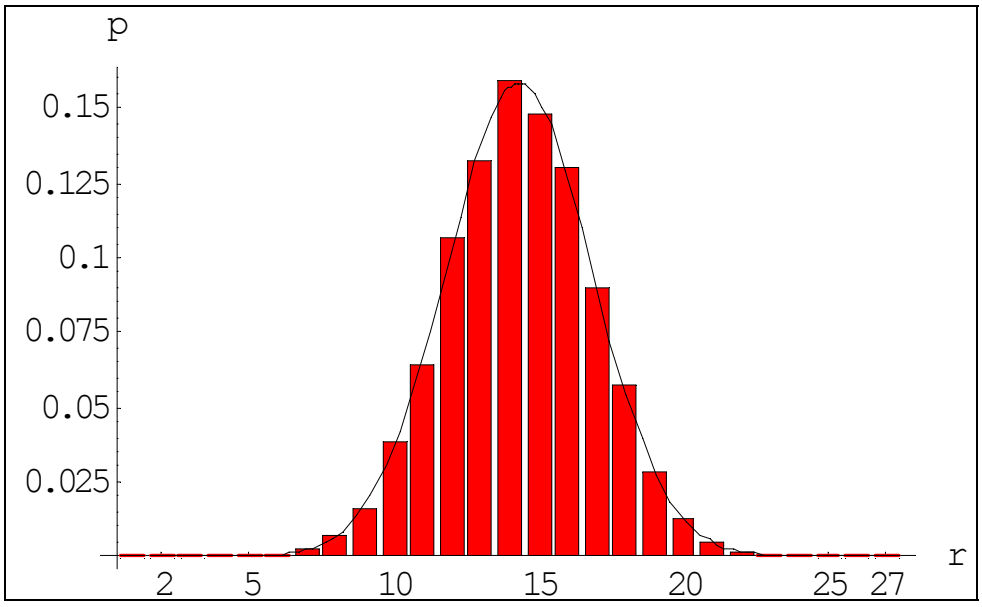

Abbildung 6.6: Approximation der Run-Verteilung durch die Normalverteilung, n=27, n1=15, n2=12

6.3 Dollar-Wechselkurse

In diesem Kapitel werden die Schwankungen des US Dollars auf Zufälligkeit geprüft.

Daten

Es werden die Wechselkurse vom Januar bis Juni 2000 getestet.
Folgende Tabelle gibt uns die US Dollar-Kurse von jedem Tag der ausgewählten Zeitperiode an.

Diese Stichprobe enthält 182 Werte = Anzahl der Tage = n.

Datum	**US Dollar in DM**	**Datum**	**US Dollar in DM**
1 Januar 2000	1,9422	1 April 2000	2,0481
2	1,9451	2	2,0476
3	1,9453	3	2,0468
4	1,9071	4	2,0429
5	1,8978	5	2,0333
6	1,8955	6	2,0299
7	1,8965	7	2,0411
8	1,9008	8	2,0495
9	1,8996	9	2,0495
10	1,9007	10	2,0477
11	1,9079	11	2,0329
12	1,8930	12	2,0397
13	1,8979	13	2,0416
14	1,9075	14	2,0493
15	1,9285	15	2,0363
16	1,9296	16	2,0352
17	1,9289	17	2,0346
18	1,9341	18	2,0538
19	1,9307	19	2,0659
20	1,9322	20	2,0793
21	1,9228	21	2,0843
22	1,9386	22	2,0848
23	1,9386	23	2,0827
24	1,9386	24	2,0804
25	1,9435	25	2,0845
26	1,9537	26	2,1241
27	1,9527	27	2,1179
28	1,9798	28	2,1509
29	2,0033	29	2,1466
30	2,0019	30	2,1453
31	2,0018		
1 Februar 2000	2,0161	1 Mai 2000	2,1464
2	2,0126	2	2,1363
3 Februar 2000	2,0040	3 Mai 2000	2,1530

4	1,9752	4	2,1855
5	1,9909	5	2,1971
6	1,9880	6	2,1790
7	1,9879	7	2,1797
8	1,9957	8	2,1807
9	1,9832	9	2,1773
10	1,9696	10	2,1558
11	1,9867	11	2,1573
12	1,9798	12	2,1701
13	1,9798	13	2,1282
14	1,9833	14	2,1273
15	1,9966	15	2,1284
16	1,9932	16	2,1467
17	1,9836	17	2,1688
18	1,9802	18	2,1872
19	1,9854	19	2,1889
20	1,9844	20	2,1760
21	1,9863	21	2,1809
22	1,9812	22	2,1826
23	1,9499	23	2,1669
24	1,9502	24	2,1577
25	1,9700	25	2,1604
26	2,0072	26	2,1449
27	2,0072	27	2,1012
28	2,0072	28	2,0920
29	2,0126	29	2,0867
		30	2,1107
		31	2,1042
1 März 2000	2,0272	1 Juni 2000	2,0870
2	2,0107	2	2,0989
3	2,0281	3	2,0727
4	2,0389	4	2,0659
5	2,0365	5	2,0678
6	2,0413	6	2,0633
7	2,0400	7	2,0505
8	2,0393	8	2,0341
9	2,0355	9	2,0450
10	2,0243	10	2,0526
11	2,0295	11	2,0519
12	2,0295	12	2,0523
13	2,0324	13	2,0529
14	2,0287	14	2,0366
15	2,0216	15	2,0425
16	2,0207	16	2,0510
17	2,0136	17	2,0254
18	2,0130	18	2,0270
19 März 2000	2,0130	19 Juni 2000	2,0299
20	2,0114	20	2,0428

21	2,0099	21	2,0450
22	2,0283	22	2,0683
23	2,0365	23	2,0874
24	2,0136	24	2,0896
25	2,0019	25	2,0898
26	2,0019	26	2,0893
27	2,0023	27	2,0880
28	2,0262	28	2,0680
29	2,0348	29	2,0751
30	2,0578	30	2,0559
31	2,0353		

Tabelle 6.3: Dollarwechselkurse Januar - Juni 2000, n=182

Nullhypothese

Die Dollar-Wechselkurse kommen zufällig zustande.

Alternativhypothese

Die Schwankungen des US Dollars sind nicht zufällig.

Die Schwankungen der amerikanischen Währung sind in Abbildung 6.7 ersichtlich.

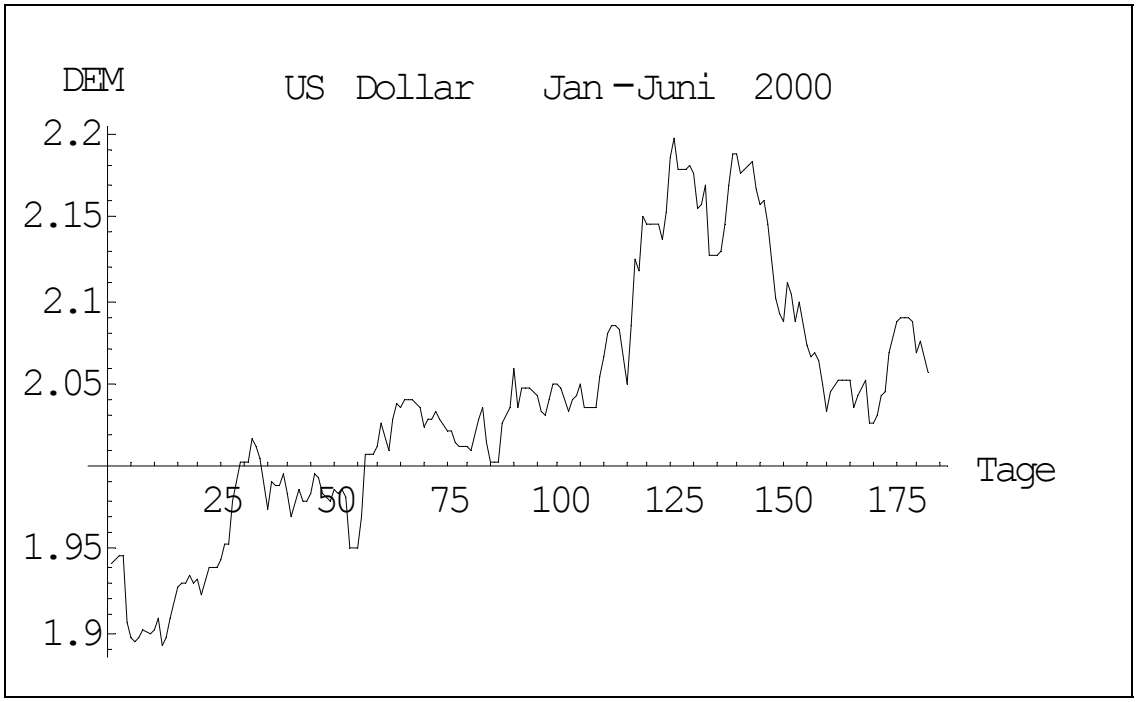

Abbildung 6.7: Wechselkurse vom US Dollar in DM, Januar - Juni 2000

EINFÜHRUNG

Wir führen in diesem Beispiel zwei Tests nach der „runs up and down“-Methode durch und wenden den zweiseitigen Run-Test an.

Beim ersten Test entsteht die Folge folgendermaßen:

Preis vom nächsten Monat $\leq$ Preis vom vorherigen Monat $\rightarrow$ 0
Preis vom nächsten Monat $>$ Preis vom vorherigen Monat $\rightarrow$ 1

Zweiter Test:

Preis vom nächsten Monat $<$ Preis vom vorherigen Monat $\rightarrow$ 0
Preis vom nächsten Monat $\geq$ Preis vom vorherigen Monat $\rightarrow$ 1

TESTDÜRCHFÜHRUNG

SCHRITT 1

Zum besseren Überblick tragen wir die Werte des Dollars zuerst in 6 Listen ein. Eine Liste beinhaltet die Dollarwerte von einem Monat.

```
januar = {1.9422, 1.9451, 1.9453, 1.9071, 1.8978, 1.8955, 1.8965, 1.9008, 1.8996, 1.9007,
          1.9079, 1.8930, 1.8979, 1.9075, 1.9285, 1.9296, 1.9289, 1.9341, 1.9307, 1.9322,
          1.9228, 1.9386, 1.9386, 1.9386, 1.9435, 1.9537, 1.9527, 1.9798, 2.0033, 2.0019,
          2.0018};

februar = {2.0161, 2.0126, 2.0040, 1.9752, 1.9909, 1.9880, 1.9879, 1.9957, 1.9832, 1.9696,
           1.9867, 1.9798, 1.9798, 1.9833, 1.9966, 1.9932, 1.9836, 1.9802, 1.9854, 1.9844,
           1.9863, 1.9812, 1.9499, 1.9502, 1.9700, 2.0072, 2.0072, 2.0072, 2.0126};

maerz = {2.0272, 2.0107, 2.0281, 2.0389, 2.0365, 2.0413, 2.0400, 2.0393, 2.0355, 2.0243,
         2.0295, 2.0295, 2.0324, 2.0287, 2.0216, 2.0207, 2.0136, 2.0130, 2.0130, 2.0114,
         2.0099, 2.0283, 2.0365, 2.0136, 2.0019, 2.0019, 2.0023, 2.0262, 2.0348, 2.0578,
         2.0353};

april = {2.0481, 2.0476, 2.0468, 2.0429, 2.0333, 2.0299, 2.0411, 2.0495, 2.0495, 2.0477,
         2.0329, 2.0397, 2.0416, 2.0493, 2.0363, 2.0352, 2.0346, 2.0538, 2.0659, 2.0793,
         2.0843, 2.0848, 2.0827, 2.0504, 2.0845, 2.1241, 2.1179, 2.1509, 2.1466, 2.1453};

mai = {2.1464, 2.1363, 2.1530, 2.1855, 2.1971, 2.1790, 2.1797, 2.1807, 2.1773, 2.1558,
       2.1573, 2.1701, 2.1282, 2.1273, 2.1284, 2.1467, 2.1688, 2.1872, 2.1889, 2.1760,
       2.1809, 2.1826, 2.1669, 2.1577, 2.1604, 2.1449, 2.1012, 2.0920, 2.0867, 2.1107,
       2.1042};

juni = {2.0870, 2.0999, 2.0727, 2.0659, 2.0678, 2.0633, 2.0505, 2.0341, 2.0450, 2.0526,
        2.0519, 2.0523, 2.0529, 2.0366, 2.0425, 2.0510, 2.0254, 2.0270, 2.0299, 2.0428,
        2.0450, 2.0683, 2.0874, 2.0896, 2.0898, 2.0893, 2.0880, 2.0680, 2.0751, 2.0559};
```

SCHRITT 2

Mit *Join* - Kommando verbinden wir die Listen zusammen, mit Hilfe von *Length* wird die Anzahl der Elemente der neuen Liste mit dem Namen „zuz“ angezeigt. Auf diese Weise können wir prüfen, ob beim Schreiben keine Daten verlorengegangen sind.

```
zuz = Join[januar, februar, maerz, april, mai, juni]
n = Length[zuz]
```

SCHRITT 3

Zum Erzeugen der Zahlenfolgen für diese Tests wird das Hilfsprogramm *runs_up_and_down_Test* verwendet (siehe Anhang Kapitel 7.1.7).

Die Folge für den ersten Test hat folgendes Muster:

```
{0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0,
 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0,
 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0,
 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0,
 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0,
 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0}
```

Mit Hilfe des Programms *runs_zaehlen* erhalten wir folgende Werte der Variablen *n1, n2* und *rbeob*:

```
n1 = 89
n2 = 93
n = 182
Die Anzahl der beobachteten Runs ist rbeob= 91
```

Entsprechend die gleichen Schritte für den zweiten Test

```
{0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0,
 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0,
 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0,
 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0,
 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0,
 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0}
```

und die Resultate:

```
n1 = 98
n2 = 84
n = 182
Die Anzahl der beobachteten Runs ist rbeob= 89
```

SCHRITT 4

Die berechneten Werte geben wir ins RUN-TEST Programm zum zweiseitigen Test ein und bekommen folgende Irrtumswahrscheinlichkeiten:

Beim ersten Test ist α gleich 94,563 %, beim zweiten 55,355 %.

Entscheidung

Die Irrtumswahrscheinlichkeit ist in beiden Fällen groß. Die Dollar-Wechselkurse kommen zufällig zustande.

Prüfung auf Normalverteilung (graphisch)

Der Stichprobenumfang ist bei diesem Test größer als bei den ersten drei Beispielen aus den Kapiteln 6.1 und 6.2.

Die Approximation der Run-Verteilung durch die Normalverteilung ist entsprechend besser. Das zeigt die Abbildung 6.8.

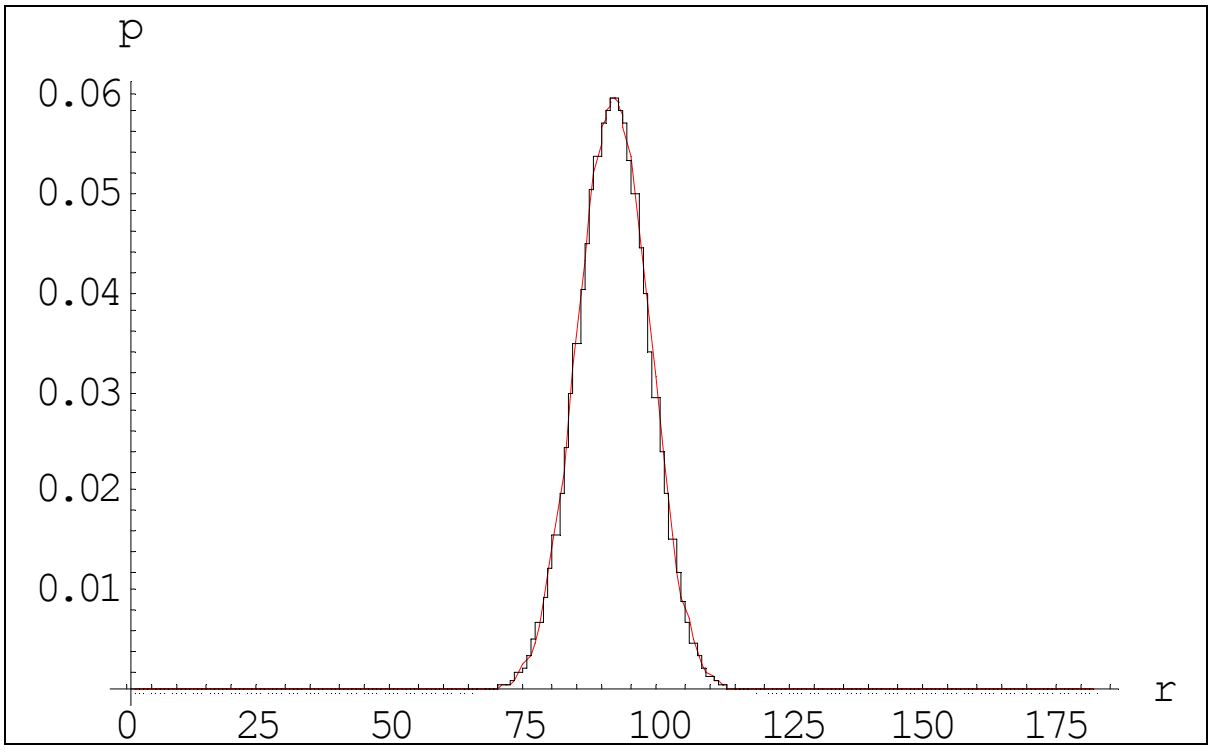

Abbildung 6.8: Approximation der Run-Verteilung durch die Normalverteilung für n=182, n1=89, n2=93 (schwarze Linie - Kurve der Run-Verteilung, rote Linie - Kurve der Normalverteilung)

6.4 Zufallszahlen

6.4.1 Begriff der Zufallszahlen

Zur Untersuchung komplexer Zufallsvorgänge, z.B. bei der Planung von Produktionssystemen oder von Großprojekten, spielen Computersimulationen eine wichtige Rolle. Dabei werden am Computer Zufallszahlen x_1 , ... , x_n mit Hilfe spezieller Algorithmen berechnet. Solche Algorithmen nennt man Zufallsgeneratoren.

Grundlegend ist dabei die Erzeugung von Zufallszahlen, die stochastisch unabhängig und gleichverteilt sind. Jede Ziffer 0,1,...,9 ist von ihren Vorgängern stochastisch unabhängig und jede tritt mit der gleichen Wahrscheinlichkeit p=0,1 auf. Da die Werte x_1 , ... , x_n tatsächlich jedoch berechnet werden, sind sie nicht echt zufällig. Man spricht deshalb auch genauer von Pseudo-Zufallszahlen, die sich fast wie echte verhalten.

6.4.2 Untersuchung der Zufallszahlen auf Zufälligkeit mit dem zweiseitigen Run-Test

Daten

Zufallszahlen erzeugt durch den Zufallsgenerator von Mathematica.

1000 ganze Zahlen zwischen 0 und 9999.

In der unteren Abbildung sieht man, dass die Zufallszahlen gleichmäßig verteilt sind.

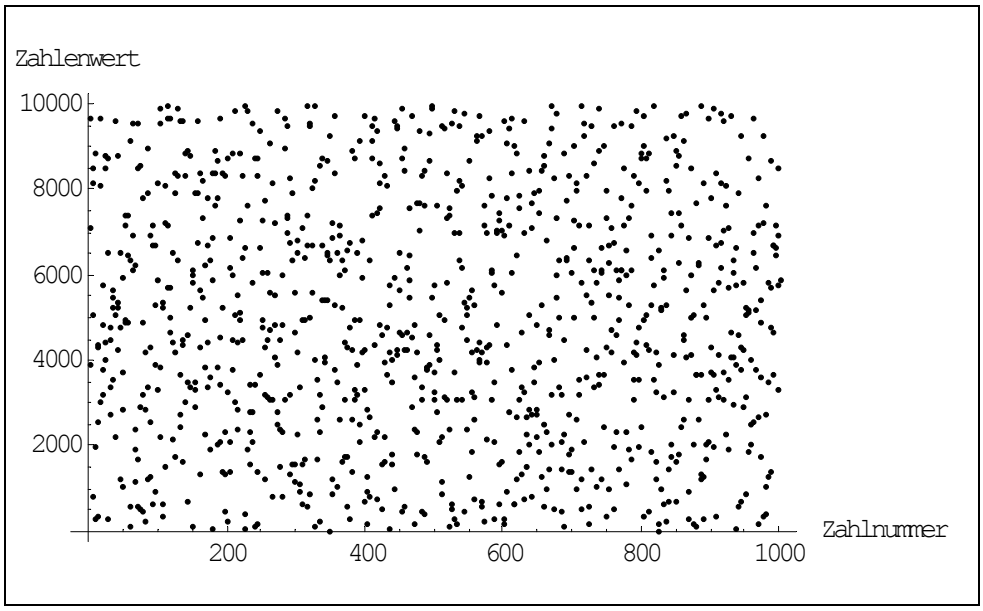

Abbildung 6.9: Punktgrafik der 1000 Zufallszahlen mit Mathematica erzeugt

Wir sortieren die Zufallszahlen aufsteigend und stellen auch dies graphisch dar. Wie man sieht, scheinen die Zufallszahlen auch unter diesem Aspekt zufällig zu sein (siehe Abbildung 6.10).

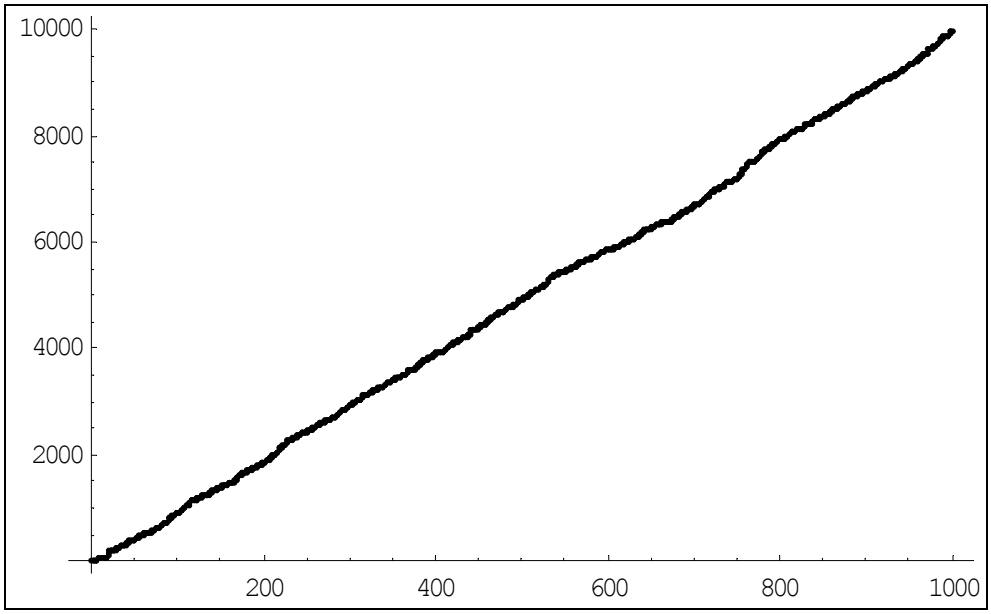

Abbildung 6.10: Punktgrafik der 1000 aufsteigend sortierten Zufallszahlen

Aber sind die Zufallszahlen auch unter anderen Aspekten tatsächlich zufällig? Dies wird mit zwei Tests geprüft.

Nullhypothese

Die Zufallszahlen sind zufällig.

Alternativhypothese

Die Zufallszahlen sind nicht zufällig.

Test A

Prüfen der Zufallszahlen nach der Methode der geraden und ungeraden Zahlen.

SCHRITT 1

Wir erzeugen 1000 Zufallszahlen mit dem folgenden Kommando:

```
zuz = Table[Random[Integer, {0, 9999}], {1000}]
```

Random ermittelt 1000 Pseudo-Zufallszahlen des Typs *Integer* zwischen 0 und 9999.

Das *Table*-Kommando wird verwendet, um gleich eine ganze Liste von Zufallszahlen zu erzeugen. Auf diese Weise kann später auf die einzelnen Zufallszahlen zugegriffen werden.

SCHRITT 2

Wir erzeugen eine Liste von Einsern und Nullen, wobei Null eine gerade und Eins eine ungerade Zahl bedeuten (siehe Kapitel 4.5.1).

Dazu ist das Hilfsprogramm *gerade_ungerade_Test* erforderlich (siehe Programmbeschreibung im Kapitel 5.6.2 und Programmlisting im Kapitel 7.1.8).

SCHRITT 3

Die Anzahl der Nullen, der Einsern und ihrer Iterationen (Runs) wird mit dem Hilfsprogramm *runs_zaehlen* berechnet (siehe Programmbeschreibung im Kapitel 5.6.3 und Programmlisting im Kapitel 7.1.7 oder 7.1.8).

SCHRITT 4

Die erhaltenen Werte der Variablen werden nun ins Programm zum zweiseitigen Run-Test eingegeben. Die Irrtumswahrscheinlichkeit α wird berechnet.

Die Schritte 1-4 müssen mehrmals durchgeführt werden. Da in diesem Kapitel hauptsächlich der Run-Test selber vorgestellt werden soll und kein Beweis für die Zufälligkeit der Zufallszahlen erbracht werden soll, können wir uns auf die 10-malige Durchführung des Tests begrenzen.

Die Ergebnisse zeigt die Tabelle 6.4.

Nummer	**n1**	**n2**	**rbeob**	α **in %**
1	499	501	493	63,518
2	522	478	504	82,599
3	510	490	475	10,921
4	507	493	496	78,058
5	490	510	502	96,468
6	495	505	495	73,017
7	486	514	505	80,524
8	499	501	502	97,469
9	494	506	481	21,884
10	508	492	517	32,258

Tabelle 6.4: Ergebnisse aus dem Test A für die Zufallszahlen, n=1000

Die Anzahl der Runs liegt fast jedesmal nicht weit vom Erwartungswert weg, etwa im Intervall $(\mu - \sigma,\ \mu + \sigma)$.

Schussfolgerung - Test A (Prüfen der Reihenfolge der geraden und ungeraden Zahlen)

Beim mehrmaligen Durchführen des Tests ist α durchschnittlich 63,6716 %. Die Irrtumswahrscheinlichkeit ist groß, das heißt, die Zufallszahlen sind unter diesem Aspekt zufällig.

Test B

Prüfen der Reihenfolge der an- und absteigenden Zufallszahlen nach der „runs up and down"- Methode.

SCHRITT 1 (siehe Schritt 1 im Test A)

SCHRITT 2

Wir erzeugen eine Liste von Einsern und Nullen nach der „runs up and down"-Methode (siehe Kapitel 4.5.1).

Dazu ist das Hilfsprogramm *runs_up_and_down_Test* erforderlich (siehe Programmbeschreibung im Kapitel 5.6.1 und Programmlisting im Kapitel 7.1.7).

SCHRITT 3 (siehe Schritt 3 im Test A)

SCHRITT 4

Das Steigen und Fallen der Zufallszahlen wurde beim ersten und zweiten Test der „runs up and down"-Methode jeweils 10 mal getestet.
Jedes Mal kamen bei beiden Tests die gleichen Werte von *n1, n2* und *rbeob.*

Die folgende Tabelle zeigt die Ergebnisse aus den 10 Testdurchläufen.

Nummer	**n1**	**n2**	**rbeob**	α **in %**
1	502	498	679	$6{,}966 \times 10^{-28}$
2	513	487	664	$2{,}293 \times 10^{-23}$
3	517	483	663	$3{,}595 \times 10^{-23}$
4	496	504	648	$9{,}567 \times 10^{-19}$
5	505	495	647	$1{,}732 \times 10^{-18}$
6	495	505	639	$1{,}940 \times 10^{-16}$
7	496	504	668	$1{,}954 \times 10^{-24}$
8	506	494	675	$1{,}254 \times 10^{-26}$
9	502	498	682	$2{,}804 \times 10^{-32}$
10	508	492	660	$4{,}164 \times 10^{-22}$

Tabelle 6.5: Ergebnisse aus dem Test B für die Zufallszahlen, n=1000

Schlussfolgerung - Test B (Prüfen der Reihenfolge der ansteigenden und absteigenden Zufallszahlen)

Die Anzahl der Runs ist jedesmal hoch, sie bewegt sich zwischen 639 und 682. Wie die zugehörige Wahrscheinlichkeitsverteilung in der Abbildung 6.11 zeigt, ist die Wahrscheinlichkeit P(U) in diesem Bereich nah an der Null (genaue Werte siehe Tabelle 6.5). Das bedeutet, dass die Zufallszahlen unter diesem Aspekt nicht zufällig sind.

Prüfung auf Normalverteilung (graphisch)

Bei diesem großen Wert von *n=1000* ist die Näherung der Run-Verteilung durch die Normalverteilung wesentlich besser als in den vorhergehenden Exempel mit kleinen Stichproben.

Diese Approximation wird in der Abbildung 6.11 graphisch veranschaulicht (die Treppenfunktion für die Wahrscheinlichkeitsfunktion der Run-Verteilung ist schwarz gekennzeichnet, die Dichte der zugehörigen Normalverteilung ist rot).

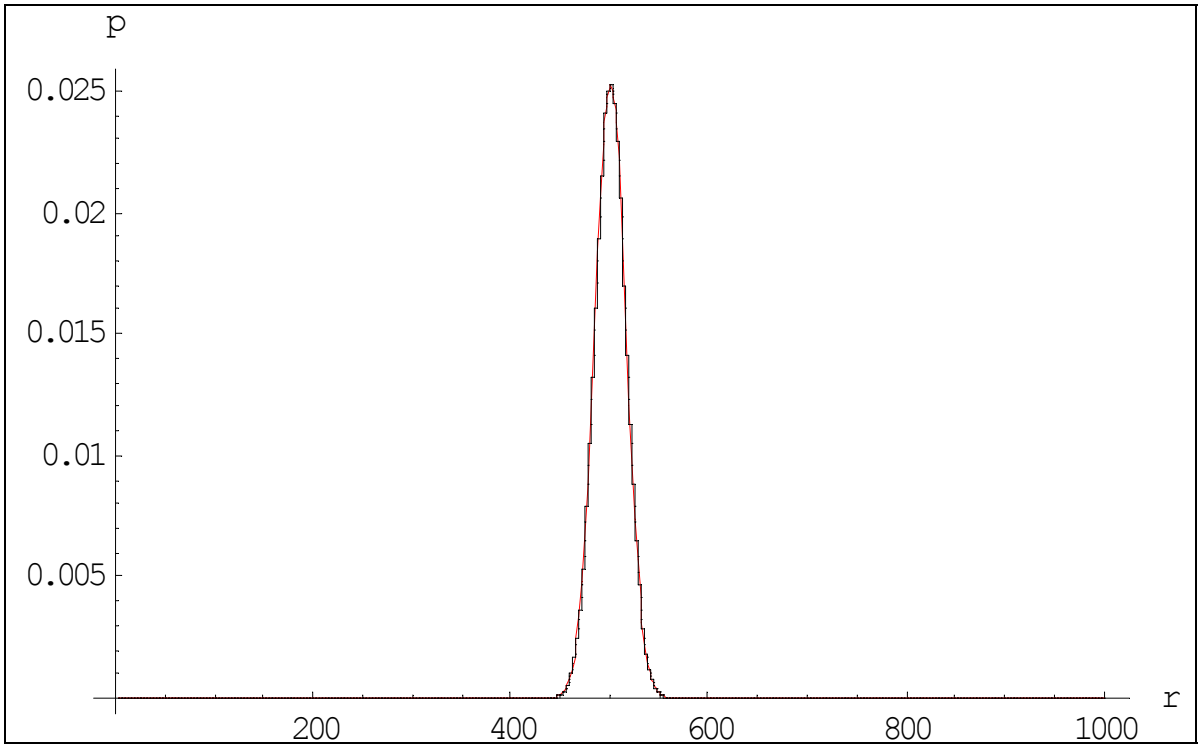

Abbildung 6.11: Approximation der Run-Verteilung durch die Normalverteilung für n1=498, n2=502 (schwarze Linie - Kurve der Run-Verteilung, rote Linie - Kurve der Normalverteilung)

Prüfung auf Normalverteilung (rechnerisch)

Wenn wir die Irrtumswahrscheinlichkeiten rechnerisch vergleichen, stellen wir fest, dass die Unterschiede nicht ohne Bedeutung sind.

Die folgende Tabelle vergleicht exakte Ergebnisse aus dem Test A (siehe Tabelle 6.4) mit den approximierten Werten der Normalverteilung.

Erklärung zu der Tabelle 6.6:

Der Erwartungswert wurde nach der Formel 4.2.3 berechnet.
α exakt wurde mit dem zweiseitigen Run-Test berechnet.
α approximiert wurde mit Hilfe der Normalverteilung bestimmt.
Der absolute Fehler ist die Differenz: α exakt - α approximiert.
Der relativer Fehler ist gleich: absoluter Fehler / α exakt.

Die letzten vier Werte der Tabelle wurden in % angegeben.

Nummer	n1	n2	rbeob	Erwartungswert	α exakt	α approx.	absoluter Fehler	relativer Fehler
1	499	501	493	500,998	63,518	61,279	2,239	0,0352
2	522	478	504	500,032	82,599	80,1372	2,4618	0,0298
3	510	490	475	500,800	10,921	10,2426	0,6784	0,0621
4	507	493	496	500,902	78,058	75,6374	2,4206	0,0310
5	490	510	502	500,800	96,468	93,9448	2,5232	0,0262
6	495	505	495	500,950	73,017	70,6518	2,3652	0,0324
7	486	514	505	500,605	80,524	78,0911	2,4329	0,0302
8	499	501	502	500,998	97,469	94,9445	2,5245	0,0259
9	494	506	481	500,928	21,884	20,7247	1,1593	0,0530
10	508	492	517	500,872	32,258	30,7351	1,5229	0,0472

Tabelle 6.6 : Vergleich der Ergebnisse der Run-Verteilung und der Normalverteilung für n=1000, aus dem Test A für die Zufallszahlen

Der absolute Fehler liegt zwischen 0,6 % und 2,5 %. Er ist deutlich kleiner als der absolute Fehler bei kleinem *n,* wo dieser Fehler 4 %-12 % betrug (siehe Tabelle 6.2).

Wir sehen hier, dass der relative Fehler am größten da ist, wo *rbeob* am weitesten vom Erwartungswert entfernt ist (siehe die Position 3 in der Tabelle). Daraus folgt, die Näherung durch die Normalverteilung ist am schlechtesten für die Randwerte der Run-Verteilung. Für die Werte von *rbeob*, die nicht weit vom Erwartungswert liegen, ist die Approximation am besten, natürlich im gegebenen Fall.

6.5 Vergleich unabhängiger Stichproben

Wir formulieren den Run-Test für das Zweistichprobenproblem.

Daten

Zwei unabhängige Stichproben, Blutdruckwerte (nur der systolischer Wert) von Rauchern und Nichtrauchern.

Die beiden Stichproben wurden in einer Allgemeinarztpraxis erhoben.

Die Blutdruckwerte von Nichtrauchern, n1=68:

140, 110, 120, 120, 120, 120, 130, 125, 180, 160, 130, 140, 170, 140, 110, 110, 120, 120, 110, 110, 90, 110, 120, 110, 115, 114, 109, 127, 124, 138, 132, 160, 105, 130, 110, 108, 120, 148, 130, 105, 115, 175, 170, 136, 158, 147, 134, 121, 135, 130, 110, 158, 149, 125, 126, 111, 128, 147, 104, 114, 129, 136, 163, 130, 179, 128, 126, 147.

Die Blutdruckwerte von Rauchern, n2=54:

133, 120, 164, 115, 127, 120, 136, 153, 127, 173, 135, 132, 146, 153, 130, 146, 126, 137, 108, 129, 120, 112, 127, 147, 114, 165, 131, 144, 114, 135, 130, 162, 109, 136, 150, 109, 130, 117, 146, 151, 131, 127, 115, 128, 132, 137, 117, 147, 118, 144, 155, 153, 120, 110

Gesamtanzahl der Daten: n = n1 + n2 = 68 + 54 = 122

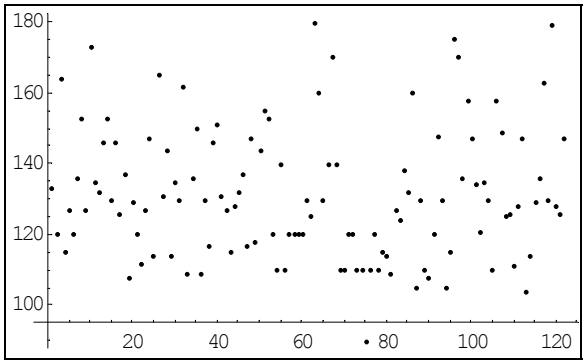

Abbildung 6.12: Punktgraphik, Blutdruckwerte der Raucher und Nichtraucher

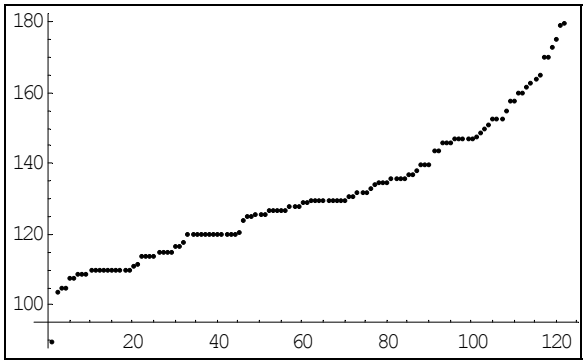

Abbildung 6.13: Punktgraphik, Blutdruckwerte der Raucher und Nichtraucher aufsteigend sortiert

Testproblem

Bei dieser Problemstellung ist ein einseitiger Test (links) angebracht. Wenn Raucher einen höheren Blutdruck haben, werden in der gemischten, aufsteigend sortierten Folge der Blutdruckwerte die Werte von Rauchern hinten und die von Nichtrauchern vorne sein. Wenn die Nichtraucher einen höheren Blutdruck haben, wäre es

umgekehrt. In beiden Fällen wären wenige Runs zu erwarten. Aus diesem Grund wäre in diesem Fall ein einseitiger Test für wenige Runs angebracht.
Wir führen den Run-Test gegen eine einseitige Alternativhypothese durch und zwar bei dem Fünf-Prozent-Signifikanz-Niveau. Es sollen folgende Hypothesen untersucht werden:

Nullhypothese

Die Blutdruckwerte der beiden Gruppen unterscheiden sich nicht. Der Blutdruck der Menschen ist unabhängig vom Rauchen.

Mit anderen Worten: Die Reihenfolge der Blutdruckwerte von beiden Gruppen ist zufällig (nicht zu viele und nicht zu wenige Runs).
Die Stichproben von Rauchern und Nichtrauchern entstammen der gleichen Grundgesamtheit.

Alternativhypothese

Die beiden Beobachtungen (d.h. die Blutdruckwerte der beiden Gruppen) unterscheiden sich. Es besteht ein Zusammenhang zwischen dem Blutdruck und dem Rauchen.

Die Reihenfolge der Blutdruckwerte der beiden Stichproben ist nicht zufällig. Es sind wenige Runs zu erwarten.

Teststatistik

Als Teststatistik wählen wir die Anzahl der beobachteten Runs (*rbeob)* in der kombinierten, geordneten Stichprobe.

TESTDÜRCHFÜHRUNG

SCHRITT 1

Die Werte werden in zwei Listen eingetragen.

```
raucher = {133, 120, 164, 115, 127, 120, 136, 153, 127, 173, 135, 132, 146, 153, 130, 146,
  126, 137, 108, 129, 120, 112, 127, 147, 114, 165, 131, 144, 114, 135, 130, 162, 109, 136,
  150, 109, 130, 117, 146, 151, 131, 127, 115, 128, 132, 137, 117, 147, 118, 144, 155, 153, 120, 110};
```

```
nichtraucher = {140, 110, 120, 120, 120, 120, 130, 125, 180, 160, 130, 140, 170, 140, 110,
  110, 120, 120, 110, 110, 90, 110, 120, 110, 115, 114, 109, 127, 124, 138, 132, 160, 105,
  130, 110, 108, 120, 148, 130, 105, 115, 175, 170, 136, 158, 147, 134, 121, 135, 130, 110,
  158, 149, 125, 126, 111, 128, 147, 104, 114, 129, 136, 163, 130, 179, 128, 126, 147};
```

SCHRITT 2

Wir bilden aus den beiden Stichproben die kombinierte, aufsteigend sortierte Stichprobe und kennzeichnen mit 0 bzw.1, ob es sich jeweils um den Blutdruck von einem Raucher oder Nichtraucher handelt.

Die beiden Listen werden zuerst zu einer Liste vereint.

```
{133, 120, 164, 115, 127, 120, 136, 153, 127, 173, 135, 132, 146, 153, 130, 146, 126, 137, 108, 129, 120,
 112, 127, 147, 114, 165, 131, 144, 114, 135, 130, 162, 109, 136, 150, 109, 130, 117, 146, 151, 131, 127,
 115, 128, 132, 137, 117, 147, 118, 144, 155, 153, 120, 110, 140, 110, 120, 120, 120, 120, 130, 125, 180,
 160, 130, 140, 170, 140, 110, 110, 120, 120, 110, 110, 90, 110, 120, 110, 115, 114, 109, 127, 124,
 138, 132, 160, 105, 130, 110, 108, 120, 148, 130, 105, 115, 175, 170, 136, 158, 147, 134,
 121, 135, 130, 110, 158, 149, 125, 126, 111, 128, 147, 104, 114, 129, 136, 163, 130, 179, 128, 126, 147}
```

Die Elemente der Liste werden aufsteigend sortiert.

```
{90, 104, 105, 105, 108, 108, 109, 109, 109, 110, 110, 110, 110, 110, 110, 110, 110, 110, 110, 111, 112,
 114, 114, 114, 114, 115, 115, 115, 115, 117, 117, 118, 120, 120, 120, 120, 120, 120, 120, 120, 120, 120,
 120, 120, 121, 124, 125, 125, 126, 126, 126, 127, 127, 127, 127, 127, 128, 128, 128, 129, 129, 130, 130,
 130, 130, 130, 130, 130, 130, 130, 131, 131, 132, 132, 132, 133, 134, 135, 135, 135, 136, 136, 136,
 136, 137, 137, 138, 140, 140, 140, 144, 144, 146, 146, 146, 147, 147, 147, 147, 147, 148,
 149, 150, 151, 153, 153, 153, 155, 158, 158, 160, 160, 162, 163, 164, 165, 170, 170, 173, 175, 179, 180}
```

Die Blutdruckwerte von Nichtrauchern werden mit 1 bezeichnet, die von Rauchern mit 0. Wir unterscheiden zwei Fälle:

In den Zahlenfolgen mit den gleichen Blutdruckwerten sind vorne immer die Werte von den Nichtrauchern, danach kommen erst die Blutdruckwerte von den Rauchern. Zum Beispiel: In der Reihe mit dem Blutdruckwert 110 gibt es 10 Beobachtungen, 9 von Nichtrauchern und eine von Rauchern.
In diesem Fall schreiben wir 1 1 1 1 1 1 1 1 1 0.
Im Fall 2 ist es umgekehrt, die Folge sieht dann wie folgt aus: 0 1 1 1 1 1 1 1 1 1.

Fall 1

```
{1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0,
 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1,
 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1,
 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1}
```

Fall 2

```
{1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0,
 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0,
 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0,
 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1}
```

SCHRITT 3

Mit Hilfe des Hilfsprogramms *runs-zaehlen* erhält man die Werte von n1, n2 und rbeob, die dann in das Programm zum einseitigen Run-Test (links) eingegeben werden.

In beiden Fällen sind die Werte gleich:

```
Anzahl der Nichtraucher (der Einser) n1 = 68
Anzahl der Raucher (der Nullen)      n2 = 54
Gesamtanzahl n = 122
Anzahl der beobachteten Runs ist rbeob = 43
```

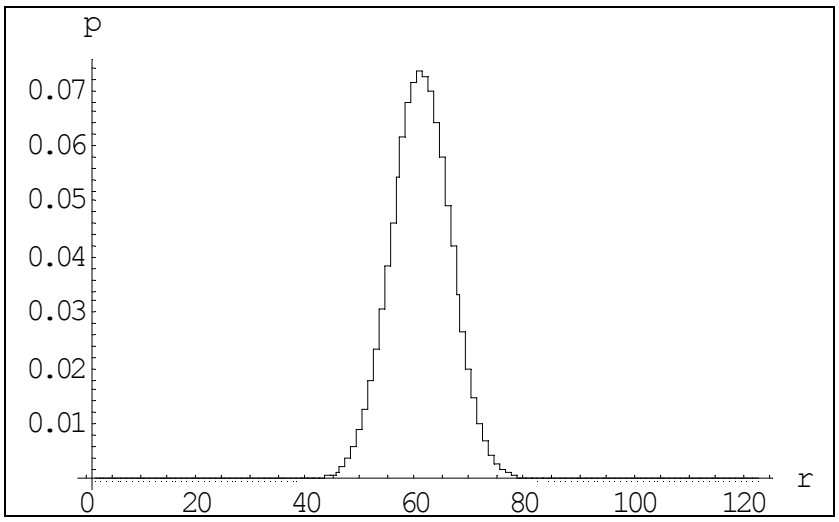

Abbildung 6.14: Wahrscheinlichkeitsfunktion für n=122, n1=68, n2=54

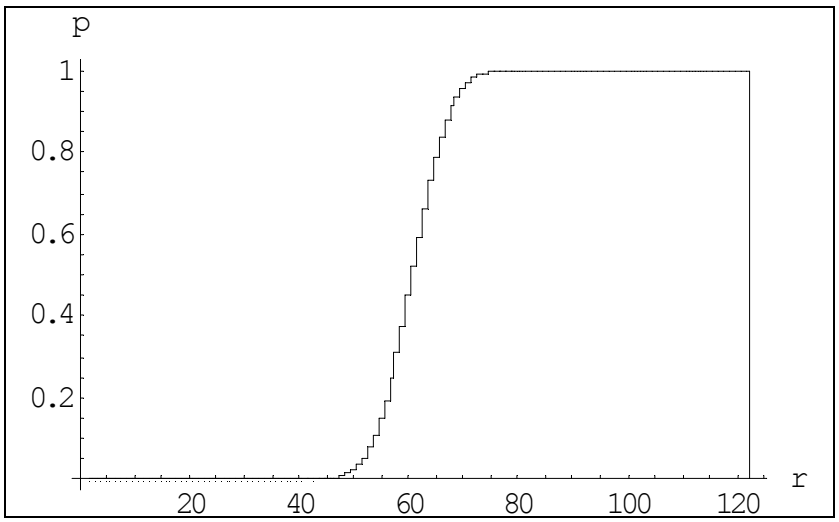

Abbildung 6.15: Verteilungsfunktion für n=122, n1=68, n2=54

Wie man in der Abbildung 6.15 sieht, ist die Summe der Wahrscheinlichkeiten für rbeob=43 nah an der Null (siehe Formel 4.4.4). Mit dem einseitigen Run-Test wurde die Irrtumswahrscheinlichkeit α gleich 0,052 % berechnet.

Schlussfolgerung

Die Einser und Nullen sind nicht „gut gemischt“, das heißt zu wenige Runs. Da das errechnete Risiko alpha gleich 0,052 % ist, wird die Nullhypothese abgelehnt. Man akzeptiert die Alternativhypothese auf dem 5%-Niveau.

Es ist anzunehmen, dass die Blutdruckwerte der Raucher und Nichtraucher zwei unterschiedlichen Grundgesamtheiten entstammen. Die Blutdruckwerte der beiden Gruppen unterscheiden sich. Offensichtlich besteht ein Zusammenhang zwischen dem Rauchen und dem Blutdruck.

Aufgrund der Datenlage kann man sagen, dass die Raucher einen höheren Blutdruck haben als die Nichtraucher.

6.6 Zusammenfassung des Kapitels zur Anwendung des Run-Tests

In dieser Diplomarbeit wurden verschiedene Tests zur Anwendung des Run-Tests durchgeführt. Im folgenden ein Überblick über die wichtigsten Themen des Kapitels:

Thema	**Problemstellung**
Kartenspiel	Sind die Karten gut gemischt?
Kinder in der Schlange vor einem Getränkeautomaten	Ist die Gruppierung der Kinder zufällig bezüglich des Geschlechts?
Treibstoffpreise	Sind die Schwankungen der Preise zufällig?
Dollar - Wechselkurse	Kommen die Wechselkurse zufällig zustande?
Zufallszahlen	Sind die Zufallszahlen unter verschiedenen Aspekten zufällig?
Blutdruckwerte der Raucher und Nichtraucher	Ist der Blutdruck der Menschen unabhängig vom Rauchen?

Tabelle 6.7: Übersicht der Themen zur Anwendung des Run-Tests

In den meisten Exempel wurde die Prüfung auf Normalverteilung durchgeführt (graphisch oder rechnerisch).

Aufgrund dieser Tests kann man sagen, dass die übliche Näherung der Run-Verteilung mit der Normalverteilung tatsächlich zu falschen Ergebnissen führen kann. Im Kapitel 6.2 wurden die Ergebnisse der exakten und approximierten Werte für n=27 in der Tabelle 6.2 verglichen.

Der absolute Fehler lag zwischen 4 % und 12 %. In einem der vier Fälle aus der Tabelle könnte der approximierte Wert der Irrtumswahrscheinlichkeit sogar zur Annahme der gegensätzlichen Hypothese führen.

Auch bei großen Stichproben mit n=1000 (siehe Beispiel mit Zufallszahlen, Kapitel 6.4) ist die Approximation noch nicht so gut. Der absolute Fehler lag zwischen 0,6 % und 2,5 %.

Das heißt, die Run-Verteilung ist erst für sehr große Werte von n approximativ normalverteilt.

Aufgrund der Untersuchung des Run-Tests kann man noch ergänzend dazu sagen, dass die Näherung durch die Normalverteilung am schlechtesten für die Randwerte der Verteilung ist.

Aus diesen Gründen wurde im Rahmen dieser Diplomarbeit das Programm erstellt, das dem Benutzer die Möglichkeit zur exakten Auswertung der dem Run-Test zugrunde liegenden Wahrscheinlichkeitsverteilung gibt.

7 Anhang

7.1 Mathematica-Notebooks

In diesem Kapitel findet man folgende Programme:

7.1.1 Hauptformeln der Verteilung

7.1.2 Erwartungswert, Varianz und Standardabweichung

7.1.3 Balkendiagramm - Wahrscheinlichkeitsverteilung für kleine *n*

7.1.4 Treppenfunktion - Wahrscheinlichkeitsverteilung für große *n*

7.1.5 Approximation der Run-Verteilung durch die Normalverteilung für kleine Werte von *n*

7.1.6 Approximation der Run-Verteilung durch die Normalverteilung für große Werte von *n*

7.1.7 Hilfsprogramm zur „runs up and down“- Methode

7.1.8 Hilfsprogramm zur Methode der geraden und ungeraden Zahlen

7.1.9 **Run-Test Programm**

Hinweis

Im Anhang dieser Diplomarbeit befindet sich ein CD-ROM Rohling, auf dem alle oben aufgeführten Notebooks gespeichert sind. Außerdem findet man dort Programme zur Erstellung der Verteilungsfunktionen (*balkenverteil_klein_n* und *verteil_gross_n)* und zur deren Approximation mit der Normalverteilung (*nv_rv_verteil_klein_n* und *nv_rv_verteil_gross_n*).

7.1.1 Hauptformeln der Verteilung im Programm

Programmname: *Hauptformeln*

Programmlisting:

```
(*    E I N G A B E   *)

n1 = 10 ; (* Eingabe von n1 *)
n2 = 8 ;  (* Eingabe von n2 *)
r = 9 ;   (* Eingabe von r  *)


(*                  B E R E C H N U N G                          *)

(*Falls r gerade Zahl ist, ist u=r/2, falls r ungerade u=(r-1)/2.
               "Floor" bedeutet die Gaussklammer, mit deren Hilfe geprüft wird, ob r gerade oder ungerade Zahl ist.*)

u := If[r / 2 - Floor[r / 2] == 0, r / 2, (r - 1) / 2];


(*** Formel 4.2.1:  P(U) für r gerade***)

formel1[n1_, n2_, u_] := 2 (Binomial[n1 - 1, u - 1] Binomial[n2 - 1, u - 1]) / Binomial[n1 + n2, n1];


(*** Formel 4.2.2:  P(U) für r ungerade ***)

formel2[n1_, n2_, u_] := ((Binomial[n1 - 1, u - 1] Binomial[n2 - 1, u]) +
                          (Binomial[n1 - 1, u] Binomial[n2 - 1, u - 1])) / Binomial[n1 + n2, n1];


(* * *   Ausgabe von P(U)    * * *)

Print["P(U=r) = ", If[r / 2 - Floor[r / 2] == 0, formel1[n1, n2, u] // N, formel2[n1, n2, u] // N]]
```

7.1.2 Erwartungswert, Varianz und Standardabweichung

Programmname: *evs*

Programmlisting:

```
(* E I N G A B E *)

n1 = 50;  (* Eingabe von n1 *)
n2 = 50 ;  (* Eingabe von n2 *)


(* B E R E C H N U N G *)

(*Erwartungswert*)
erwartungswert[n1_, n2_] := (2 n1 n2 / (n1 + n2)) + 1;

(*Varianz*)
varianz[n1_, n2_] := (2 n1 n2 (2 n1 n2 - n1 - n2)) / ((n1 + n2) (n1 + n2) (n1 + n2 - 1));

(*Standardabweichung*)
standab = √varianz[n1, n2];

Print["Der Erwartungswert für n1= ", n1, "  und n2= ", n2, " ist m= ", erwartungswert[n1, n2] // N]
Print["Die Varianz für n1= ", n1, "  und n2= ", n2, " ist v = ", varianz[n1, n2] // N]
Print["Die Standardabweichung für n1= ", n1, "  und n2= ", n2, " ist s = ", standab // N]
```

7.1.3 Balkendiagramme - Wahrscheinlichkeitsverteilung für kleine *n*

Programmname: *balken_klein_n*

Programmlisting:

```
(****B A L K E N D I A G R A M M  f ü r  k l e i n e  n *)


Clear[n1, n2, wertex, ywerte, balken, daten];


(************************ Variablen ****************************)
(*
n1        = Anzahl der Objekte vom Typ 1
n2        = Anzahl der Objekte vom Typ 2
n   = Gasamtanzahl der Objekte vom Typ 1 und 2
r   = Anzahl aller möglichen Runs
*)
(*************************************************************)
(*                   E I N G A B E                           *)
(*************************************************************)

n1 = 18;  (* E I N G A B E der Anzahl der Objekte vom Typ 1   *)
n2 = 18;  (* E I N G A B E der Anzahl der Objekte vom Typ 2   *)

 (*************************************************************)
(*                   E N D E   M e n u e                     *)
(*************************************************************)


(****************B E R E C H N U N G E N *******************)

(*Beschreibung zur Berechnung findet man im Notebook:
                                                  Hauptformeln*)

formel1[n1_, n2_, u_] := 2 (Binomial[n1 - 1, u - 1] Binomial[n2 - 1, u - 1]) / Binomial[n1 + n2, n1];

formel2[n1_, n2_, u_] := ((Binomial[n1 - 1, u - 1] Binomial[n2 - 1, u]) +
        (Binomial[n1 - 1, u] Binomial[n2 - 1, u - 1])) / Binomial[n1 + n2, n1];

u := If[r / 2 - Floor[r / 2] == 0, r / 2, (r - 1) / 2];

formel := If[r / 2 - Floor[r / 2] == 0, formel1[n1, n2, u] // N,
                                        formel2[n1, n2, u] // N];

n = n1 + n2;
```

```
(** E r s t e l l u n g * d e s * B a l k e n d i a g r a m m s **)


(*Hier werden die x-Werte des Diagramms erzeugt (die Werte von r),
    die Liste der x-Werte muss von 2 bis n gehen*)

wertezux = Table[a, {a, 2, n}];


(*Hier wird eine ganze Liste von
        y-Werten  (den jeweiligen Wahrscheinlichkeiten fuer die Verteilung) fuer das Balkendiagramm erzeugt*)

ywerte = Table[formel, {r, 2, n}];

(*Transpose transponiert eine zweidimensionale Liste, auf diese
        weise entstehen Zahlenpaare mit jeweils zuerst dem Wert von y-Werten und dann x-Werten, d.h. in
        genau umgekehrter Reihenfolge, so verlangt BarChart*)

daten = Transpose[{ywerte, wertezux}];

(*********G R A P H I S C H E   D A R S T E L L U N G**********)

                                            (***Balkendiagramm ***)

(*Das Balkendiagramm bekommt den Namen: balken
    BarChart erstellt ein Balkendiagramm fuer die paarweise
    gegebenen Daten
    mit PlotRange->All wird der y-Bereich eingestellt, mit All wird das Diagramm vollständig angezeigt,
 mit AxesLabel werden die Achsen beschriftet (x-Achse mit r,
                                                    y-Achse mit p)
    *)


balken = BarChart[daten, PlotRange -> All, AxesLabel -> {r, p}];
```

7.1.4 Treppenfunktion - Wahrscheinlichkeitsverteilung für große *n*

Programmname: *treppe_gross_n*

Programmlisting:

```
(* T R E P P E N F U N K T I O N  für große n (ist für alle n geeignet)*)

Clear[n1, n2, zuxwerte, wertey, listex, wertevonp, s];

(************************ Variablen ***************************)
(*
n1        = Anzahl der Objekte vom Typ 1
n2        = Anzahl der Objekte vom Typ 2
n    = Gasamtanzahl der Objekte vom Typ 1 und 2
r    = Anzahl der Runs
*)
(*************************************************************************)
(*                          E I N G A B E                                *)
(*************************************************************************)

n1 = 50;      (* E I N G A B E der Anzahl der Objekte vom Typ 1  *)
n2 = 50;      (* E I N G A B E der Anzahl der Objekte vom Typ 2  *)


(**************************B E R E C H N U N G E N*************************)

(*Die Beschreibung zur Berechnung findet man im Notebook: Hauptformeln*)

formel1[n1_, n2_, u_] := 2 (Binomial[n1 - 1, u - 1] Binomial[n2 - 1, u - 1]) / Binomial[n1 + n2, n1]

formel2[n1_, n2_, u_] := ((Binomial[n1 - 1, u - 1] Binomial[n2 - 1, u]) +
                          (Binomial[n1 - 1, u] Binomial[n2 - 1, u - 1])) / Binomial[n1 + n2, n1]

u := If[r / 2 - Floor[r / 2] == 0, r / 2, (r - 1) / 2]

formel := If[r / 2 - Floor[r / 2] == 0, formel1[n1, n2, u] // N, formel2[n1, n2, u] // N]

n = n1 + n2;

(**********Erstellung der Treppenfunktion ***********)

(*Hier werden die x-Werte der Treppenfunktion erzeugt*)
zuxwerte = Table[a, {a, 1, n}];
xwertekopie = zuxwerte;

(*Mit Insert wird jedes Element in die Position 2 i eingefügt*)
For[i = 0, i < n - 1, i++; xwertekopie = Insert[xwertekopie, zuxwerte[[i]], 2 i]];

zus = {n};
```

```
(*Hier entsteht eine Liste mit x-Werten{1,1,2,2,3,3,...,n,n} *)
listex = Join[xwertekopie, zus];


(*Hier werden die y-Werte der Treppenfunktion erzeugt*)

wertevonp = Table[formel, {r, 2, n}];

kopiewertevonp = wertevonp;

(*For-Schleife erzeugt mit Insert-Kommando Liste mit doppelten Elementen
                der Liste wertevonp*)
For[i = 0, i < n - 1, i++; kopiewertevonp = Insert[kopiewertevonp, wertevonp[[i]], 2 i]];

zusatz = {0};
(*Mit Join werden die drei Listen verbunden
  Es entsteht eine Liste mit dem Namen wertey mit{0,p2,p2,p3,p3,...,pn,pn,0}, wobei p2,...,pn
  die entsprechenden Wahrscheinlichkeiten für r=2,...,n bedeuten*)

wertey = Join[zusatz, kopiewertevonp, zusatz];

(*Die Treppenfunktion wird mit dem Kommando Line erstellt,das eine Linie durch die gegebenen Punkte zeichnet.
    Diese Zahlenpaare werden mit Table erzeugt und setzen sich
 aus den x-Werten (aus der Liste listex, allerdings wird 0,5 zu
 jedem Wert der Liste addiert, damit der x-Wert genau unter der
    Mitte der einzelnen Stufe von der Treppenfunktion liegt)
    und y-Werten (aus der Liste wertey) zusammen.*)

i = 1;
s = Line[Table[{listex[[i]] + 0.5, wertey[[i]]}, {i, 2 n}]];

(*und graphisch dargestellt*)

(********G R A P H I S C H E  D A R S T E L L U N G*********)

(*Graphics repräsentiert  eine zweidimensionale graphische Abbildung
  und wird mit Show dargestellt.
       Die Option AxesTrue aktiviert die Darstellung von Achsen.*)

Show[Graphics[s], Axes -> True, PlotRange -> All, AxesLabel -> {r, p},
    AxesOrigin -> {1, 0}]
```

7.1.5 Approximation der Run-Verteilung durch die Normalverteilung für kleine *n*

Programmname: *nv_rv_klein_n*

Programmlisting:

```
(*APPROXIMATION DER RUN-VERTEILUNG DURCH DIE NORMALVERTEILUNG
                        - WAHRSCHEINLICHKEITSFUNKTION FÜR KLEINE WERTE VON N*)

Remove["Global`*"]; (*beseitigt alle bisher definierten Symbole*)
Needs["Graphics`Graphics`"]; (*lädt Packages für BarChart*)


(***************************** Variablen *********************************)
(*
n1        = Anzahl der Objekte vom Typ 1
n2        = Anzahl der Objekte vom Typ 2
*)
(*************************************************************************)
 (*                          E I N G A B E                                *)
(*************************************************************************)

n1 = 15;  (* E I N G A B E der Anzahl der Objekte vom Typ 1   *)
n2 = 15;    (* E I N G A B E der Anzahl der Objekte vom Typ 2   *)


(*********************B E R E C H N U N G E N ************************)

(*Die Beschreibung zur Berechnung findet man im Notebook: Hauptformeln*)

formel1[n1_, n2_, u_] :=
    2 (Binomial[n1 - 1, u - 1] Binomial[n2 - 1, u - 1]) / Binomial[n1 + n2, n1];

formel2[n1_, n2_, u_] := ((Binomial[n1 - 1, u - 1] Binomial[n2 - 1, u]) +
                          (Binomial[n1 - 1, u] Binomial[n2 - 1, u - 1])) / Binomial[n1 + n2, n1];

u := If[r / 2 - Floor[r / 2] == 0, r / 2, (r - 1) / 2];

formel := If[r / 2 - Floor[r / 2] == 0, formel1[n1, n2, u] // N, formel2[n1, n2, u] // N];

n = n1 + n2;

(****E r s t e l l u n g  des  B a l k e n d i a g r a m m s ****)

(*Ausführliche Kommentare zur Erstellung des Balkendiagramms findet
       man im Notebook: balken klein n*)

(*Im Falle der Approximation muss das Balkendiagramm ein zusätzliches Zahlenpaar
    {0,1} erhalten, da Plot der Normalverteilung automatisch bei 1 beginnt.*)
```

```
nvwertex = Table[a, {a, 1, n}];

y = Table[formel, {r, 2, n}];
neuewertey = Prepend[y, 0];

nvdaten = Transpose[{neuewertey, nvwertex}];

nvbalken = BarChart[nvdaten, PlotRange -> All];

(*************************************************************************)

(******Die Wahrscheinlichkeitsdichte der zugehörigen Normalverteilung******)

Needs["Statistics`Master`"]; (*lädt Packages für alle Statistik-Funktionen*)

erwartungswertklein[n1_, n2_] := (2 n1 n2 / (n1 + n2)) + 1;

varianzklein[n1_, n2_] := (2 n1 n2 (2 n1 n2 - n1 - n2)) / ((n1 + n2) (n1 + n2) (n1 + n2 - 1));

(*nvndist bezeichnet die Normalverteilung mit dem Erwartungswert
            und der Standardabweichung der Run-Verteilung*)

nvndist = NormalDistribution[erwartungswertklein[n1, n2], √varianzklein[n1, n2]];
grenze = n;

(*Graphische Darstellung als schwarze Linie.
  Mit PDF wird die Dichtefunktion berechnet, nvrun ist die Kurve
        der zugehörigen Dichte, mit Plot kann man Funktionskurve darstellen*)

nvrunklein = Plot[PDF[nvndist, x] , {x, 2, grenze}, PlotRange -> All];

(*****Approximation der Run-Verteilung durch die Normalverteilung*****)

(*Mit dem Kommando Show können mehrere Grafik-Objekte kombiniert
            gezeigt werden.*)

Show[nvbalken, nvrunklein]
```

7.1.6 Approximation der Run-Verteilung durch die Normalverteilung für große Werte von *n*

Programmname: *nv_rv_gross_n*

Programmlisting:

```
(*APPROXIMATION DER RUN-VERTEILUNG DURCH DIE NORMALVERTEILUNG
        - WAHRSCHEINLICHKEITSFUNKTION    FÜR GROßE WERTE VON N *)

Remove["Global`*"] (*beseitigt alle bisher definierten Symbole*)

(************************** Variablen **********************************)
(*
n1        = Anzahl der Objekte vom Typ 1
n2        = Anzahl der Objekte vom Typ 2
*)

(***********************************************************************)
(*                        E I N G A B E                                *)
(***********************************************************************)

n1 = 16;      (* E I N G A B E der Anzahl der Objekte vom Typ 1  *)
n2 = 16;      (* E I N G A B E der Anzahl der Objekte vom Typ 2  *)

(**********************B E R E C H N U N G E N************************)

(*Die Beschreibung zur Berechnung findet man im Notebook: Hauptformeln *)

formel1[n1_, n2_, u_] := 2 (Binomial[n1 - 1, u - 1] Binomial[n2 - 1, u - 1]) / Binomial[n1 + n2, n1];

formel2[n1_, n2_, u_] := ((Binomial[n1 - 1, u - 1] Binomial[n2 - 1, u]) +
                          (Binomial[n1 - 1, u] Binomial[n2 - 1, u - 1])) / Binomial[n1 + n2, n1];

u := If[r / 2 - Floor[r / 2] == 0, r / 2, (r - 1) / 2];

formel := If[r / 2 - Floor[r / 2] == 0, formel1[n1, n2, u] // N, formel2[n1, n2, u] // N];

n = n1 + n2;

                                (******Erstellung der Treppenfunktion ******)

(*Ausführliche Kommentare dazu findet man in Notebook: treppe gross n*)

xwerte = Table[a, {a, 1, n}];
xwertekopie = xwerte;
For[i = 0, i < n - 1, i++; xwertekopie = Insert[xwertekopie, xwerte[[i]], 2 i]];

zus = {n};
listex = Join[xwertekopie, zus];

wertevonp = Table[formel, {r, 2, n}];
```

```
kopiewertevonp = wertevonp;
For[i = 0, i < n - 1, i++;
       kopiewertevonp = Insert[kopiewertevonp, wertevonp[[i]], 2 i]];
zusatz = {0};
wertey = Join[zusatz, kopiewertevonp, zusatz];

i = 1;
s = Line[Table[{listex[[i]] + 0.5, wertey[[i]]}, {i, 2 n}]];

treppe = Show[Graphics[s], Axes -> True, PlotRange -> All];

(*************************************************************************)

(*Hier wird die Wahrscheinlichkeitsdichte
                der Normalverteilung für die Run-Verteilung erstellt*)

Needs["Statistics`Master`"]; (*lädt Packages für alle Statistik-Funktionen*)

erwartungswert[n1_, n2_] := (2 n1 n2 / (n1 + n2)) + 1;
varianz[n1_, n2_] := (2 n1 n2 (2 n1 n2 - n1 - n2)) / ((n1 +
                     n2) (n1 + n2) (n1 + n2 - 1));

(*ndist bezeichnet die Normalverteilung mit dem Erwartungswert
            und der Varianz der Run-Verteilung*)

ndist = NormalDistribution[erwartungswert[n1, n2], √varianz[n1, n2]];
grenze = n;

(*graphische Darstellung als rote Linie*)

    nvrun = Plot[PDF[ndist, x] , {x, 2, grenze}, PlotRange -> All, PlotStyle -> {RGBColor[1, 0, 0]}];

 (***  Approximation der Run-Verteilung durch die Normalverteilung  ***)

 Show[nvrun, treppe, AxesLabel -> {r, p}, AxesOrigin -> {1, 0}]
```

7.1.7 Hilfsprogramm zur „runs up and down“-Methode

Programmname: *runs_up_and_down_Test*

Programmlisting:

```
(***H I L F S P R O G R A M M  ZUR "RUNS UP AND DOWN" - METHODE ***)

Clear[n1, n2, rbeob, zuz];

(*H I N W E I S  F Ü R  D E N  B E N U T Z E R*)

(*Die Liste ZUZ kann eine beliebige Zahlenliste sein, hier beinhaltet
        sie 1000 Zufallszahlen von 0 bis 9999.*)

zuz = Table[Random[Integer, {0, 9999}], {1000}];
n = Length[zuz];

 (*Hier wird eine Liste von Nullen und Einsern
            nach der "runs up and down"-Methode erzeugt.
            Nächste Zahl <= Vorherige Zahl -> 0
          Nächste Zahl >  Vorherige Zahl -> 1

   Diese Liste besteht aus zwei Listen:
            "zusatz" für das erste Element und "pm" für den ganzen Rest.*)

(*Das erste Element der Liste zuzpm*)
zusatz = {If[zuz[[1]] <= zuz[[2]], 0, 1]};

pm = Table[If [zuz[[i + 1]] <= zuz[[i]], 0, 1], {i, 1, n - 1}];

zuzpm = Join[zusatz, pm];

n1 = Count[zuzpm, 1]; (*Die Einser werden gezählt*)

n2 = n - n1;           (*Die Nullen werden gezählt*)

(*Die Anzahl der beobachteten Runs wird gezählt.*)
rbeob = 1;
Do[If [zuzpm[[i]] != zuzpm[[i + 1]], rbeob = rbeob + 1], {i, 1, n - 1}]

(**A U S G A B E  D E R  V A R I A B L E N **)

Print["Die Anzahl der Einser  n1= ", n1]
Print["Die Anzahl der Nullen  n2= ", n2]
Print["Gesamtanzahl            n= ",  n]
Print["Die Anzahl der beobachteten Runs ist rbeob= ", rbeob];
```

7.1.8 Hilfsprogramm zur Methode der geraden und ungeraden Zahlen

Programmname: *gerade_ungerade_Test*

Programmlisting:

```
(**H I L F S P R O G R A M M  ZUR METHODE DER GERADEN UND UNGERADEN ZAHLEN**)

Clear[n1, n2, rbeob, zuz];


(*H I N W E I S  F Ü R  D E N  B E N U T Z E R*)

(*Die Liste ZUZ kann eine beliebige Zahlenliste sein, hier beinhaltet
        sie 1000 Zufallszahlen von 0 bis 9999.*)

(*********************************************************************)

zuz = Table[Random[Integer, {0, 9999}], {1000}];
n = Length[zuz];      (*Die Länge der Liste wird mit n bezeichnet*)

(*Hier wird eine Liste von Nullen und Einsern
            folgendermaßen erzeugt.:    gerade Zahl -> 0
                                        ungerade Zahl -> 1 *)

zuzug = 2 (zuz / 2 - Floor[zuz / 2]);

n1 = Count[zuzug, 1];(*Die Einser werden gezählt*)
n2 = n - n1;          (*Die Nullen werden gezählt*)

(*Die Anzahl der beobachteten Runs wird gezählt.*)
rbeob = 1;
Do[If [zuzug[[i]] != zuzug[[i + 1]], rbeob = rbeob + 1], {i, 1, n - 1}];


(**A U S G A B E  D E R  V A R I A B L E N **)

Print["Die Anzahl der Einser  n1= ", n1]
Print["Die Anzahl der Nullen  n2= ", n2]
Print["Gesamtanzahl            n= ", n]
Print["Die Anzahl der beobachteten Runs ist rbeob= ", rbeob];
```

7.1.9 R u n - T e s t

Programmname: *RUN-TEST*

Programmlisting:

```
(*************************R U N - T E S T*******************************)

(****** Test zur Berechnung der exakten Irrtumswahrscheinlichkeit ******)


(****************************** Variablen *******************************)
(*

r    = Anzahl aller möglichen Runs
n1         = Anzahl der Objekte vom Typ 1
n2         = Anzahl der Objekte vom Typ 2
n    = Gasamtanzahl der Objekte vom Typ 1 und 2
rbeob= die beobachtete Anzahl von Runs
NS   = Anzahl der Nachkommastellen bei der Ausgabe von alpha
m    = Erwartungswert
alpha     = Irrtumswahrscheinlichkeit alpha

zeitlimit = der vorgegebe Zeitlimit in Sekunden für den Lauf
               des Programms, hauptsächlich bezieht er sich auf
               die Berechnung von Wahrscheinlichkeitssummen

testwahl = bestimmt die Art des Run-Tests, 1: Zweiseitiger Test
                                                        2: Einseitiger Test links
                                                      3: Einseitiger Test rechts
*)

(**********************************************************************)

Clear[n1, n2, rbeob, alpha, testwahl, zeitlimit, NS];(*löscht die Zuordnung von Werten zu den genannten Symbolen*)


(*******************************START**********************************)
(*                          E I N G A B E                             *)
(**********************************************************************)

zeitlimit = 10;(*E I N G A B E vom Zeitlimit in Sekunden *)

n1 = 68;        (* E I N G A B E der Anzahl der Objekte vom Typ 1  *)
n2 = 54;        (* E I N G A B E der Anzahl der Objekte vom Typ 2  *)

rbeob = 43;     (* E I N G A B E der Anzahl der Runs *)

NS = 3;         (* E I N G A B E der Anzahl der Nachkommastellen bei
                       der Ausgabe der Irrtumswahrscheinlichkeit alpha.*)
```

```
(* * *        A U S W A H L   D E S   R U N - T E S T S        * * *)

testwahl = 2;   (*E I N G A B E der Testart *)

 (* 1 = Zweiseitiger Run-Test
    2 = Einseitiger Run-Test links
    3 = Einseitiger Run-Test rechts
 *)

(*********************************************************************)
(*                      E N D E   M e n u e                          *)
(*********************************************************************)

(*   *   *              Z E I T L I M I T              *   *   *)

zuvielzeit[] := (Print["Die vorgegebene Zeit von ", zeitlimit,
        " Sekunden wurde überschritten. Bitte wählen Sie kleinere Werte für n1 und n2 !"];
            Abort[];);


(*************** S I C H E R H E I T S A B F R A G E N **************)

(*Berechnung von n, notwendig schon bei der Sicherheitsabfrage*)
 n = n1 + n2;

If[rbeob > n || rbeob < 2,
     Print["Bitte wählen Sie 2 =< r =< n !"],

 If[testwahl != 1 && testwahl != 2 && testwahl != 3,
   Print["Bitte wählen Sie für die Testwahl die Ziffer 1,2 oder 3!"],



  (**  *****************B E R E C H N U N G E N *****************    **)

  (*Berechnungen für alle Testarten *)

        (*Formel 4.2.1 für gerade Anzahl von r*)
  formel1[n1_, n2_, u_] := 2 (Binomial[n1 - 1, u - 1] Binomial[n2 - 1, u - 1]) / Binomial[n1 + n2, n1];

  (*Formel 4.2.2 für ungerade Anzahl von r*)
  formel2[n1_, n2_, u_] := ((Binomial[n1 - 1, u - 1] Binomial[n2 - 1, u]) +
                            (Binomial[n1 - 1, u] Binomial[n2 - 1, u - 1])) / Binomial[n1 + n2, n1];

        u := If[r / 2 - Floor[r / 2] == 0, r / 2, (r - 1) / 2];

        (*universale Formel*)
        formel := If[r / 2 - Floor[r / 2] == 0, formel1[n1, n2, u] // N, formel2[n1, n2, u] // N];
```

```
summe = 0; (*die Summe wird auf Null gesetzt*)

(*Bei den Berechnungen der Wahrscheinlichkeitssummen wird folgende Option verwendet*)
(*TimeConstrained[ausdruck,t,fehlaudruck] liefert Fehlausdruck zurück, wenn
        die Zeitbeschränkung t nicht eingehalten wird. Hier wird sumliste gleich Null,
    wenn der vorgegebene Zeitlimit überschritten wird.*)

(*Berechnungen für den zweiseitigen Run-Test*)

(*Hier werden für rbeob kleiner Erwartungswert,     die vorderen Balken aufsummiert und in die Liste "sumliste" geschrieben
  und für rbeob größer gleich Erwartungswert, nur die hinteren*)

    berechnungzwei[] := (
    m = (2 n1 n2 / (n1 + n2)) + 1; (*Erwartungswert Formel 4.2.3 wird berechnet*)

    If[rbeob < m,
sumliste = TimeConstrained[Table[summe = summe + formel, {r, 2, rbeob}], zeitlimit, Null],
   sumliste = TimeConstrained[Table[summe = summe + formel, {r, rbeob, n}], zeitlimit, Null]
    ];

    z = Table[formel, {r, rbeob, rbeob}];(*Hier wird die Wahrscheinlichkeit für r=rbeob
                                            berechnet, für den Fall dass alpha > 0.5*)
                );

    (*Berechnungen für den linksseitigen Run-Test*)

    (*Hier werden nur die vorderen Balken (für r von 2 bis rbeob) aufsummiert und in die Liste "sumliste" geschrieben*)

    berechnungeinslinks[] := (
sumliste = TimeConstrained[Table[summe = summe + formel, {r, 2, rbeob}], zeitlimit, Null];);

   (*Berechnungen für den rechtsseitigen Run-Test*)

   (*Hier werden nur die hinteren Balken (für r von rbeob bis n) aufsummiert und in die Liste "sumliste" geschrieben*)

       berechnungeinsrechts[] := (
       sumliste = TimeConstrained[Table[summe = summe + formel, {r, rbeob, n}], zeitlimit, Null];);
```

```
(*************************************************************************)

      (*Nach der Zeitüberschreitung wird das Unterprogramm "zuvielzeit" aufgerufen.*)
      If[ sumliste == Null, zuvielzeit[]];

     (* ****************************DER RUN-TEST***************************** *)

     (*Der zweiseitige Run-Test*)

     testzwei[] := (If[rbeob < m,
     alpha =  (sumliste[[rbeob - 1]]), (*alpha = das letzte Element von sumliste*)
     alpha = (sumliste[[n - rbeob + 1]])];

     (*Weil man nicht genau weiß, bei welchem r die Summe der Balken größer gleich 0.5 ist -> Sicherheitsabfrage*)
     If[alpha >= 0.5, alpha = 1 - alpha + z[[1]]];

     (**Falls alpha immer noch größer gleich 0.5 ist, dann ist alpha = 0.5 **)
  If[alpha >= 0.5, alpha = 0.5];);

     (*Der linksseitige Run-Test*)
     testeinslinks[] := (alpha = sumliste[[rbeob - 1]];);

     (*Der rechtsseitige Run-Test*)
     testeinsrechts[] := (alpha = sumliste[[n - rbeob + 1]];);

 (************************Allgemeine A U S G A B E************************)

  ausgabe[] := (Print[""];
  Print["Anzahl der Beobachtungen vom Typ 1, n1 = ", n1];
                Print["Anzahl der Beobachtungen vom Typ 2, n2 = ", n2];
                Print["Gesamtanzahl aller Beobachtungen,    n = ", n];
             Print["Anzahl der beobachteten Runs,    rbeob = ", rbeob];
  Print["2 ≤ r ≤ ", n];
                Print[""];
              );
```

```
(*************************************************************************)
(*             A U S G A B E der  Irrtumswahrscheinlichkeit alpha      *)
(*************************************************************************)

      (****Ausgabe für den zweiseitigen Run-Test****)

          ausgabezwei[] := (Print["Zweiseitiger Run-Test"];
       Print["Mit der Irrtumswahrscheinlichkeit alpha =",
                  PaddedForm[200 alpha, {NS + 3, NS}], " %",
    "wird die Nullhypothese Ho abgelehnt zugunsten der Alternativhypothese."];);

      (****Ausgabe für den einseitigen Run-Test links****)

 ausgabeeinslinks[] := (Print["Einseitiger Test links"];
          Print["Mit einer Irrtumswahrscheinlichkeit alpha =",
                  PaddedForm[100 alpha, {NS + 3, NS}], " %","wird Ho abgelehnt zugunsten der Alternativhypothese."];);

   (****Ausgabe für den einseitigen Run-Test rechts****)

          ausgabeeinsrechts[] := (Print["Einseitiger Test rechts"];
          Print["Mit einer Irrtumswahrscheinlichkeit alpha =",
                  PaddedForm[100 alpha, {NS + 3, NS}], " %","wird Ho abgelehnt zugunsten der Alternativhypothese."];);

(*************************************************************************)

   (*Hier wird das Unterprogramm mit der allgemeinen Ausgabe aufgerufen*)
   ausgabe[];

  (*Je nach Testart werden hier die entsprechenden Unterprogramme mit der Berechnung,
          Testdurchführung und Ausgabe von alpha aufgerufen*)

  If[testwahl == 1, (berechnungzwei[]; testzwei[]; ausgabezwei[];),

                  If[testwahl == 2, (berechnungeinslinks[]; testeinslinks[]; ausgabeeinslinks[];),

                                    (berechnungeinsrechts[]; testeinsrechts[]; ausgabeeinsrechts[];)

                ]; (*ende von else von If testwahl=2*)
       ]; (*ende von else von If testwahl=1*)


 ] (* ende von else von If testwahl!=1 && testwahl!=2 && testwahl!=3 *)
] (* ende von else von If r > n oder r < 2 *)
```

7.2 Abkürzungsverzeichnis

Die kursiv geschriebenen Bezeichnungen, kennzeichnen Symbole aus den Mathematica-Programmen. Die Abkürzungen in den Klammern wurden in dieser Diplomarbeit abwechselnd verwendet.

σ	Standardabweichung
σ^2	Varianz
μ *(m)*	Erwartungswert
α *(alpha)*	Irrtumswahrscheinlichkeit = Wahrscheinlichkeit für den Fehler 1. Art
n	Stichprobenumfang = Gesamtanzahl aller Objekte
n_1 *(n1)*	Anzahl der Objekte vom Typ 1
n_2 *(n2)*	Anzahl der Objekte vom Typ 2
r_{ij}	Anzahl der Runs des Objekts vom Typ i mit der Länge j
r	Anzahl der Runs
rbeob	Anzahl der beobachteten Runs
P(U)	Wahrscheinlichkeit für U (nach Formeln 4.2.1 und 4.2.2)
U	Gesamtanzahl der Runs (=r)
u	r/2 falls r gerade, (r-1)/2 falls r ungerade
p	Wahrscheinlichkeit
Z	Prüfgröße
H_0	Nullhypothese
H_1	Alternativhypothese
x_m	Beobachtungen der ersten Stichprobe
y_n	Beobachtungen der zweiten Stichprobe

7.3 Abbildungsverzeichnis

7.4 Tabellenverzeichnis

7.5 Literaturverzeichnis

Literaturhinweise in dieser Diplomarbeit sehen wie folgt aus:

- [Literaturnummer] allgemeiner Hinweis
- ([Literaturnummer], Seitenzahl) nach Zitaten.

[1] **Kreyszig, Erwin:** Statistische Methoden und ihre Anwendungen

[2] **Sachs, Lothar:** Angewandte Statistik (1974)

[3] **Bamberg/Baur:** Statistik

[4] **Bosch, Karl:** Statistik für Nichtstatistiker

[5] **Willms, Gerhard:** „C“- Das Grundlagen Buch

[6] **Sprent P.:** Applied nonparametric statistical methods

[7] **Sidney, Siegel:** Nonparametric statistics for the behavioral sciences

[8] **Mood, Alexander, Graybill, Franklin und Boes, Duane:** Introduction to the Theory of Statistics

[9] **Müller, P.H.:** Lexikon der Stochastik

[10] **Wolfram, Stephen:** Mathematica Version 3

[11] **Lohnes, Paul und Cooley, William:** Einführung in die Statistik mit EDV-Übungen

[12] **Kofler, Michael:** Mathematica, Einführung und Leitfaden für den Praktiker

[13] **Stelzer, Ernst H.K.:** Mathematica, Ein systematisches Lehrbuch mit Anwendungsbeispielen

[14] **McCollough, Celeste, Loche von Atta:** Statistik programmiert

[15] **Fahrmeir Ludwig, Künstler Rita, Pigeot Iris, Tutz Gerhard:** Statistik. Der Weg zur Datenanalyse.

[16] **Büning Herbert, Trenkler Götz:** Nichtparametrische statistische Methoden

[17] **Roberts, Harry und Wallis Allen:** Methoden der Statistik

[18] **Bortz, Jürgen:** Statistik für Sozialwissenschaftler (5. Auflage)