Roulette sowie eine Einführung in den Umgang mit dem R-Window-Editor, kurz: `relax`

H. P. Wolf

Version vom 19.5.2005

http://www.wiwi.uni-bielefeld.de/~wolf/software/relax/relax.html

1 Eine alte Geschichte

Im Jahre 1894 untersuchte Karl Pearson Roulette-Spielergebnisse, die angeblich aus den Casinos von Monte Carlo stammten. Die Ergebnisse wiesen Eigenschaften auf, die nicht mit den Gesetzen der statistischen Theorie in Übereinstimmung waren. Es war demnach etwas faul mit dem Roulette in Monte Carlo. Als verantwortungsbewusster Mensch schlug er vor, umgehend die Spielstätten zu schließen und mit den freigesetzten Mitteln die Wissenschaft zu fördern. Wie Sie sicher wissen, gibt es aber immer noch Spielhöllen in Monte Carlo. Und, würden Sie sich jetzt – nach diesem Hinweis und als ein ökonomisch denkender Mensch – ohne weitere Überlegungen an einen Spieltisch setzen und ihre letzten Moneten riskieren? Natürlich nicht, sondern sie würden sich zunächst mit den Argumenten von Herrn Pearson auseinandersetzen oder? Wer sich näher für diese interessiert, sei verwiesen an:

> K. Pearson (1894): Science and Monte Carlo, in: Fortnightly Review, Vol. LV. New Series, London.

1.1 Eine Hilfestellung

Das Verständnis der Pearsonschen Argumentation erfordert einige statistische Grundbegriffe, wie sie zum Beispiel im Rahmen einer statistischen Einführungsvorlesung vermittelt werden. Zunächst benötigt man eine Vorstellung von zu erwartenden Eigenschaften eines Roulette-Spiels. Diese ließen sich experimentell mit einem Modellroulettespiel studieren. Auf diesem Wege könnte man Roulettegesetze herausfinden, ohne die Gefahr großer Geldverluste eingehen zu müssen. Mit den gefundenen Eigenschaften könnte man später wahre Spielergebnisse überprüfen. Gleichzeitig bekommt man so einen Erfahrungsschatz, der einige theoretische Zusammenhänge der Statistik mit Leben zu füllen vermag. Dieses Papier und die zugehörige technische Umgebung sollen hierbei helfen.

1.2 Zweck dieses Papiers

In diesem Papier wird eine Umgebung beschrieben, mit der jeder verschiedene Experimente mit einem virtuellen Roulette-Spiel machen kann. Man benötigt nur einen Rechner und etwas Software aus dem vielgerühmten Internet. Das klingt

einfach und ist es hoffentlich auch. Zwar besitzen die technischen Elemente der Umgebung eine gewisse Kompliziertheit, jedoch sollte die Oberfläche einfach zu hantieren sein. Die Diskussion verläuft auf zwei Ebenen: Vordergründig werden ein paar einfache Fragen zum Roulettespiel diskutiert. Hintergründig wird dem Leser der Umgang mit dem gerade motivierten Instrumentarium beschrieben. Dieses besitzt einige Fähigkeiten, die vielen anderen Software-Produkten leider fehlen. Besonders hinzuweisen ist auf die Eigenart vieler Programme, den Gedankengängen des Anwenders – zum Beispiel des Statistikers – keine Bedeutung beizumessen und folgerichtig dafür keine Unterstützung und keinen Raum einzuräumen. Die Unterstützung der Gedankengänge bildet deshalb den Ausgangspunkt für die hier vorzustellende Umgebung. Wie lässt sich die Grundidee der Umgebung mit dem Namen `relax` beschreiben?

2 Philosophie von `relax` I: Datenanalysen

Die der Umgebung zugrundeliegende Philosophie geht auf den Prozess zurück, auf dem das Tun unter dem Dach der Wissenschaft basiert:

1. Der Gedanke ist das zentrale Element wissenschaftlicher Arbeit, deshalb muss er unbedingt festgehalten werden.
2. Aktionen müssen für Dritte nachvollziehbar geplant und aufgeschrieben werden.
3. Erst nach Erledigung der ersten beiden Punkt darf eine Aktion ausgelöst werden.
4. Die Ergebnisse aus den Handlungen werden beobachtet, beschrieben und lösen neue Gedanken aus.

Aus technischer Sicht reduziert sich der beschriebene Prozess für die statistische Arbeit auf:

> ... beschreibe Gedanken, Idee und Aktion → starte diese → Ergebnisse
> →
> → beschreibe Gedanken ...

Das führt im Protokoll zu dem Erscheinungsbild:

> ... Text → Anweisungen → Ergebnisse → Text ...

Betrachten wir ein Beispiel.

2.1 Häufigkeiten beim Roulettespiel

Stellen wir uns vor, wir wären hinter folgender Frage her:

Wie häufig tritt ROT bei 1000 Wiederholungen des Roulette-Spiels auf?

Wenn man davon ausgeht, dass keine höheren magischen Kräfte im Spiel sind, werden Sie hoffentlich geneigt sein zu glauben, dass es eine feste – wenn auch unbekannte – Wahrscheinlichkeit gibt, mit der die Roulette-Kugel eine rote Zahl trifft. Auch stimmt hoffentlich die Vorstellung, dass bei einer großen Wiederholungszahl die relative Trefferanzahl nur wenig von der unbekannten Wahrscheinlichkeit abweicht. Diesen Zusammenhang können wir mit Hilfe eines virtuellen Roulette-Spiels erforschen, indem wir zunächst die Bedingungen – wie die Wahrscheinlichkeiten – des Spiels fixieren, dann die virtuellen Kugel wiederholt werfen und die Ergebnisse auszählen. Folgende Anweisungen setzen diese Überlegungen in einer noch unbekannten, aber hoffentlich intuitiv verständlichen Sprache um.

1 ⟨* 1⟩ ≡

```
wahrscheinlichkeiten <- c(rot=18/37, schwarz=18/37, gruen=1/37)
farben       <- c("rot", "schwarz", "gruen")
anz.wd       <- 1000
result       <- sample(farben, anz.wd, replace=TRUE,
                   wahrscheinlichkeiten)
table(result)
```

Wir erhalten als Ergebnis:

```
Mon Apr 18 13:13:27 2005
result
  gruen     rot schwarz
     28     502     470
```

In der ersten Zeile im Abschnitt der Anweisungen werden die gewünschten Wahrscheinlichkeiten für ein rotes, ein schwarzes und ein grünes Ergebnis festgelegt und in der zweiten die möglichen Farben der Felder. Zeile drei fixiert die Anzahl der Spielwiederholungen. In Zeile vier wird der Ziehungsprozess beschrieben. Zum Schluss werden die Ergebnisse nach Farben ausgezählt.

Mit etwas Vertrauen wird der Leser glauben, dass diese Anweisungen ein virtuelles Roulette definieren und 1000 Ziehungen simulieren. Wir wollen davon ausgehen, dass die gewählte Notation, nämlich die der Sprache R, vom Rechner verstanden wird.

Nach den Anweisungen folgt die Ausgabe des Rechners. Erstaunlicherweise ist trotz der großen Wiederholungszahl der Unterschied zwischen der Anzahl roter und schwarzer Ausfälle beträchtlich. Vielleicht funktioniert unsere virtuelle Maschine nicht ganz richtig. Deshalb wollen wir das Experiment wiederholen. Wir erhalten:

```
Mon Apr 18 13:14:05 2005
result
  gruen     rot schwarz
     34     504     462
```

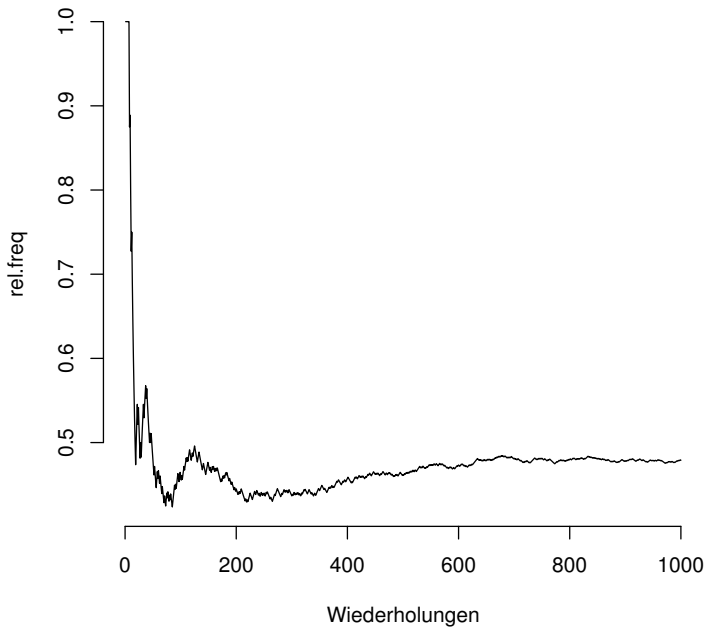
Oh Schreck, was ist mit "schwarz" los? Und noch einmal:

```
Mon Apr 18 13:14:39 2005
result
  gruen     rot schwarz
     33     479     488
```

Jetzt ist rot ein wenig zu kurz gekommen. Zur Illustration soll ein Plot die Entwicklung der relativen Häufigkeiten für die roten Felder folgen. Genauer wollen wir zu jeder Anzahl von Spielen ausrechnen und darstellen, wie viele rote Felder relativ bis zu dieser Spielanzahl aufgetreten sind:

2 ⟨* 1⟩+ ≡

```
plot(cumsum(result=="rot")/(1:anz.wd),type="l",
     xlab="Wiederholungen", ylab="rel.freq", bty="n")
```



Wir sehen, dass die Kurve immer glatter wird. Entspricht das nicht unserer Hoffnung oder Erwartung?

2.2 Zur Technik

In dieser Form könnten wir weiter experimentieren. Wir wollen jedoch nun ein paar Bemerkungen zur Technik machen. Um genau das Beschriebene zu erhalten, bedarf es einer Umgebung, die für uns sowohl Berechnungsaufträge als auch Textverarbeitungsaufgaben erledigt. Und genau das leistet – Wer hätte es anders gedacht? – die Umgebung `relax`. Ganz unten und für den Anwender fast unsichtbar wird übrigens ein Interpreter mit dem Namen "R" verwendet. Doch soll uns hier weniger die Architektur der Umgebung interessieren als der Umgang. Gegenstand dieses Papiers ist ihre Bedienung, nicht jedoch Installation und auch nicht Syntax der verwendete Sprache zur Beschreibung von Aktionen. Da die Erstellung von Gedankenreports ein intendierter Zweck der Umgebung ist, wird das Werkzeug im Folgenden auch als Report-Manager bezeichnet. Wie geht es los und was ist zu tun?

2.3 Start des Report-Managers

Es soll hier beschrieben werden, was der Anwender tun muss und kann, zunächst: Wie lässt sich die Oberfläche in Gang bringen? Folgende Tätigkeiten sind zu erledigen:

1. starte R je nach Installation durch Tastureingabe von

   ```
   R
   ```

   oder durch Doppelklick auf die R-Ikone

   ```
   Doppelklick
   ```

2. lade die notwenige Zusatzfunktionalität, das Paket mit dem R-Editor `relax()`:

   ```
   > library(relax)
   ```

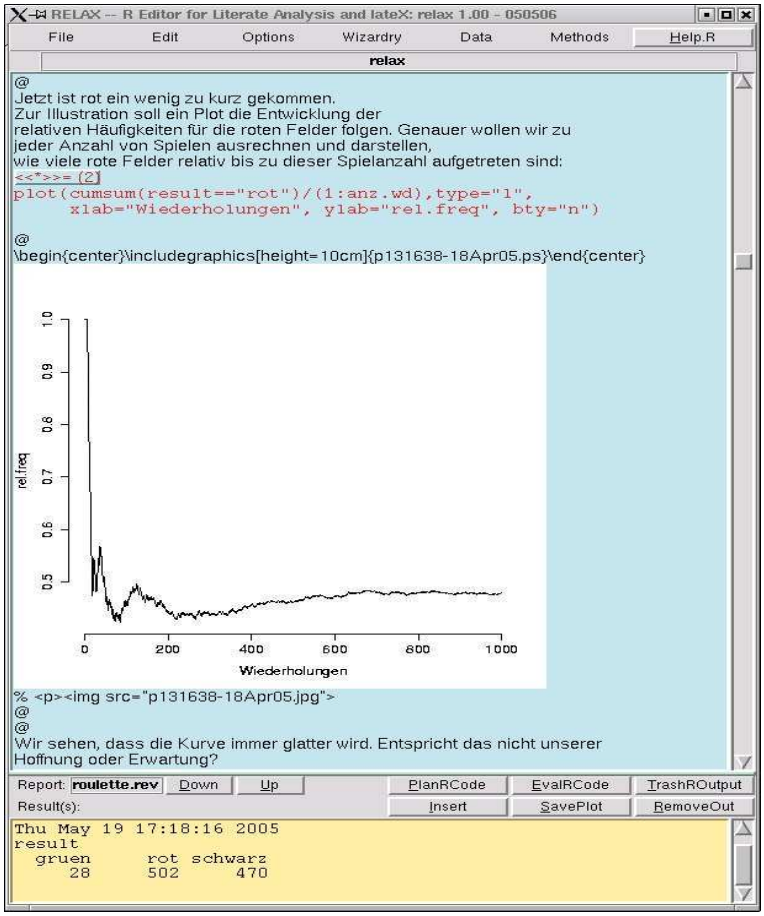
3. starte den Manager:

   ```
   > relax()
   ```

Jetzt sollte sich ein neues Fenster mit der Überschrift

RELAX – R editor for Literate Analysis and lateX: relax – ⟨Versionsdatum⟩

öffnen. Im Zentrum befindet sich eine hellblaue Fläche, in die über die Tastatur Zeichnen eingegeben werden können. Nach einer gewissen Zeit könnte der Editor folgendes Erscheinungsbild besitzen:



Wir erkennen oben in dem Bildschirmabdruck ein Stück Text gefolgt von einer Plot-Anweisung (direkt nach der Zeile mit der Zeichenkette «*»=). Unten ist ein eingeblendeter Plot-Output zu erkennen.

In dem oberen Text-Fenster (himmelblaue Farbe) lassen sich Gedanken eintragen. Zum Beispiel ist die Rohversion des vorliegenden Dokumentes durch Eingabe des Textes und der Anweisungen in diesem Fenster entstanden. Für die Arbeit bzw. das Aufschreiben sind sehr wenige Regeln zu beachten.

2.4 Regeln und Operationen für die Reporterstellung

Fundamental für die Erstellung statistischer Reports ist die Festlegung, wie Text und Anweisungen voneinander getrennt werden:

2.4.1 → Text und Anweisungen trennen

Ein "@" am Anfang einer Zeile leitet einen neuen *Text*-Abschnitt (oder *Text-Chunk*) ein, also zum Beispiel einen niedergeschriebenen Gedanken. Die Zeichenfolge «*»= beendet den vorausgehenden Text und zeigt den Start von auszuführenden Anweisungen an. Der Anweisungsabschnitt (oder emphCode-Chunk) endet bei der nächsten Zeile, in der ganz links ein Klammeraffen (@) steht.

«*»= – trennt Text und nachfolgende Anweisungen.
"@" – trennt Anweisungen und nachfolgenden Text.

2.4.2 → EvalRCode – Code ausführen

Spannend wird die Sache natürlich erst, wenn sich die beschriebenen Aktionen auch ausführen lassen. Für diesen Zweck ist unter dem Eingabefenster ein Aktivierungsknopf mit der Aufschrift EvalRCode angebracht:

EvalRCode – aktiviert die Anweisungen des Reports, auf die der Mauszeiger zeigt.

Der Output einer Auswertung wird im unteren (strandfarbenen) Output-Fenster angezeigt. Falls der Output nicht gefällt, kann er schnell wieder gelöscht werden:

2.4.3 → RemoveOut – Output löschen

Ist der Output im gelben Outputfenster unbrauchbar, lässt er sich durch Druck auf den Knopf RemoveOut löschen:

RemoveOut – löscht das Outputfenster.

2.4.4 → Insert – Output in Report übernehmen

Für den Report geeignete Ergebnisse sollten übernommen bzw. eingefügt werden. Hierfür ist der Knopf Insert zuständig:

Insert – fügt den Inhalt des Outputfensters unter dem aktuellen Code-Chunk ein.

2.4.5 → SavePlot – Graphiken kopieren

Wie lassen sich Bilder integrieren? Hierzu befindet sich unter Edit ein Eintrag, der das aktuelle Bild in das Arbeitsfenster kopiert. Gleichzeitig werden auf Dateiebene Kopien im Jpeg- und im Postscript-Forma abgelegt, um für unterschiedliche Weiterverarbeitungen gerüstet zu sein.

SavePlot – kopiert gelungene Graphiken.

2.4.6 → TrashOutput – Output aus Report entfernen

Falls ein Output in einem Report doch wieder entfernt werden muss, kann der Mauszeiger auf diesen positioniert und durch Druck auf TrashOutput entfernt werden. Dies verkürzt das übliche Vorgehen per Markierungen.

TrashOutput – löscht Output aus Reportfenster.

2.4.7 → PlanNew – erstellt leeren Text-Code-Rohling

Schnell wird es lästig, per Hand die vorgestellten Trennzeichen einzugeben. Zur Vereinfachung kann der Reportersteller den Knopf PlanNew drücken. Hierdurch wird ein leerer Text-Chunk und ein leerer Code-Chunk erzeugt.

PlanNew – erstellt leeren Text- und Code-Chunk.

2.4.8 → Verwaltung – Reportspeicherung

Jetzt ist nur noch zu beantworten, wie niedergeschriebene Inhalte auf dem Rechner als Datei abgelegt werden. Dieses ist – wie PC-Gurus bestimmt schon geahnt haben – unter dem Menü "File" zu finden, das sich wie üblich oben links in der Ecke befindet. Der Menüpunkt SaveReport speichert den Inhalt des Reportfensters in zwei Formaten: Einerseits als html-Datei, die mit einem Browser angeschaut werden kann. Andererseits als Text-Datei, die ziemlich genau der Darstellung im Arbeitsfenster entspricht und weiterverarbeitet werden kann. Vielleicht ist es das Beste an dieser Stelle, nicht auf noch mehr technische Details einzugehen und zunächst den Leser zum eigenen Experimentieren zu ermuntern. Zum Beispiel kann jeder einen Text zu verfassen, diesen dann als html-Dokument speichern und ihn dann mit seinem Lieblingsbrowser zu betrachten.

File – gestattet verschiedene Dateioperationen.
SaveReport – speichert den Report als html- und rev-Datei.

Eine Randbemerkungen seien noch gestattet: Ausgaben werden jenseits einer bestimmten Grenze abgeschnitten, da oft sehr lange Outputs irrtümlich zustande gekommen sind. Die Grenze für den Output kann jedoch über → Options → SetOutputLength gesetzt werden. Nach der Vorstellung von der Entstehung eines handgeschriebenen Protokolls wird in der Regel das untere Ende des Textes vorangetrieben und das Dokument eben unten verlängert. Der Analytiker plant immer wieder neue Analyseschritte, beschreibt seine Ideen, formuliert seine Anweisungen, startet Berechnungen und überträgt geeignete Ergebnisse in den Report, die es wieder zu kommentieren gilt. Wer sich an diese Vorgehensweise hält, wird später automatisch ein korrektes Protokoll der Gedanken in den Händen halten, das die Gedanken in ihrer Entstehungsreihenfolge zeigt – eben den Gedankengang.

An sich ist damit ein wesentlicher Kern des Report-Managers beschrieben. Wenn man noch ein wenig die statistische Sprache zur Beschreibung von Aktionen beherrscht, lassen sich in dieser Form umfangreiche statistische Analysen durchführen und bequem längere statistische Reports erstellen. Jedoch ist damit nur die eine Seite von `relax` beschrieben. Bevor wir die andere Seite beschreiben, wollen wir uns noch etwas mit dem Roulette auseinandersetzen.

3 Philosophie II: Wiederbelebung von Dokumenten

Wir wollen das Roulette-Beispiel fortsetzen, um an diesem den Einsatz zweiter Art zu diskutieren. Gehen wir einmal folgender Frage nach:

3.1 Wie stark variieren eigentlich die Häufigkeiten für ROT?

Die simulierten Ergebnisse zu der Frage, wie häufig man ROT sieht, waren auf den ersten Blick erstaunlich unterschiedlich. Kann das mit rechten Dingen zugehen? Das einfachste ist, dass wir sehr häufig das Experiment wiederholen und uns dann die sich ergebenden Häufigkeiten anschauen. Damit jeder bei der Nachahmung zu identischen Ergebnissen kommt, muss der eingebaute Zufallsmechanismus in gleicher Weise eingerichtet werden. Dieses bewirkt folgende Anweisung:

3 ⟨* 1⟩+ ≡
```
set.seed(13)
```

Nun wollen wir als Experiment 100 Einzelspiele (anz.spiele) spielen und dieses Experiment 100 Male (anz.wd) wiederholen. Insgesamt sind also 10.000 Einzelspiele (spiele) erforderlich. Die Experimente werden in der Anweisungszeile umgesetzt, in der die "Funktion" sample auftaucht. Die verbleibenden Zeilen arrangieren die Ergebnisse in einer Matrix mit anz.spiele Zeilen und anz.wd Spalten, ermitteln für jede Spalte die sich ergebenden relativen Häufigkeiten des Ergebnisses ROT und erzeugen (letzte Zeile) eine Tabelle, aus der abgelesen werden kann, welche relativen Häufigkeiten wie oft eingetreten sind.

4 ⟨* 1⟩+ ≡
```
wahrscheinlichkeiten <- c(rot=18/37, schwarz=18/37, gruen=1/37)
farben       <- c("rot", "schwarz", "gruen")
anz.spiele   <- 100
anz.wd       <- 100
spiele       <- anz.wd*anz.spiele
result       <- sample(farben, spiele, replace=TRUE,
                       wahrscheinlichkeiten)
result       <- matrix(result, anz.spiele, anz.wd)
rel.freq     <- apply(result=="rot", 2, sum)/anz.spiele
table(rel.freq)
```

Hier ist das Ergebnis:

```
Mon Apr 18 13:20:51 2005
rel.freq
0.37 0.39  0.4 0.42 0.43 0.44 0.45 0.46 0.47 0.48 0.49  0.5 0.51 0.52 0.53 0.54
   1    1    6    1    4    2   12    5    7    5   13    7   10    3    7    2
0.55 0.56 0.58 0.59  0.6
   6    2    1    3    2
```

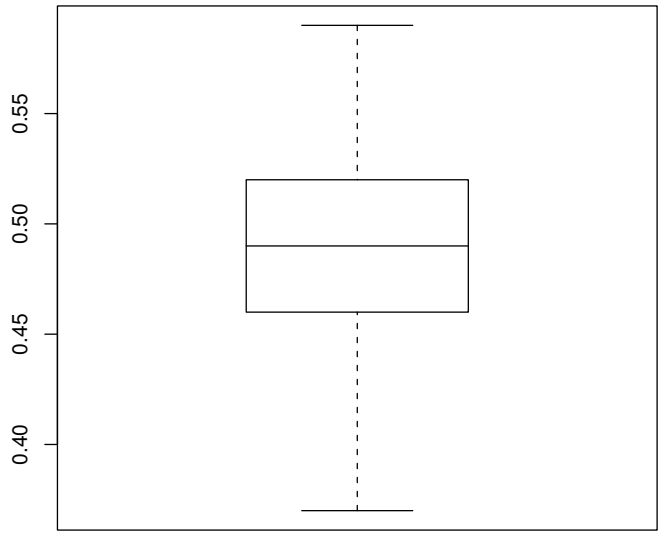
Wir sehen, dass mittlere relative Häufigkeiten häufiger auftauchen als extreme. Wenn wir von den Rändern die jeweils 5 extremsten entfernen, ergibt sich ein Intervall von 0.42 bis 0.57. Dieses besitzt als Mitte 0.495, welche interessanterweise

sehr nahe bei 18/37 liegt. Ganz in der Nähe liegt auch die Stelle, an der sich die größte Anzahl eingestellt hat: 10 mal ergab sich eine relative Häufigkeit von 0.48.

Von der Theorie müsste jetzt jeder erwarten, dass sie ein Gesetz liefert, nach dem sich die beobachtete Variabilität einstellt oder zumindest beschreiben lässt. Ohne weitere theoretische Überlegungen können wir keine Aussage darüber machen, ob die vorliegenden Beobachtung abwegig ist oder nicht. Verwundern mag die große Breite von Werten, die sich eingestellt hat.

Dazu erstellen wir einen Boxplot von den relativen Häufigkeiten:

5 ⟨* 1⟩+ ≡
```
boxplot(rel.freq, bty="n")
```

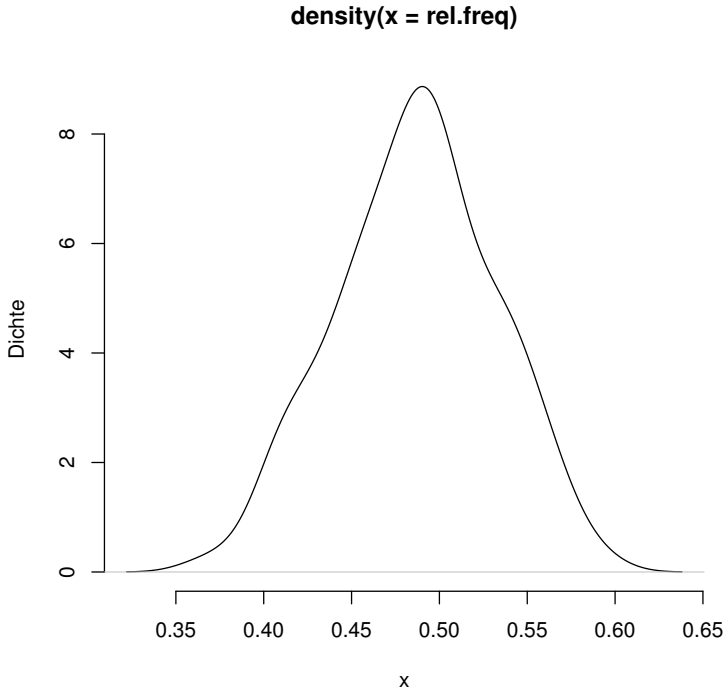


Dieser Plot zeigt am Rand die Extremwerte (0.37, 0.60). Weiter ist nach der Sortierung der Häufigkeiten der Wert, der in der Mitte liegt, durch die waagerechte Linie im Innern der Box dargestellt. Die Box beschreibt die 50 Prozent der Werte, die in der Mitte liegen. Man sieht, dass in der Mitte die Werte dichter zusammenliegen als am Rand.

Zur Verdeutlichung der Gedrängtheit von Beobachtungen, also wie eng Beobachtungen beieinanderliegen, eignet sich die sogenannte Dichtespur. Die Dichtespur ist ein statistisches Verfahren, das ohne Rechner kaum praktikabel ist. Je dichter Werte beeinander liegen, umso höher verläuft die generierte Kurve.

6 ⟨* 1⟩+ ≡
```
plot(density(rel.freq), ylab="Dichte", xlab="x", bty="n")
```

Wir erhalten:

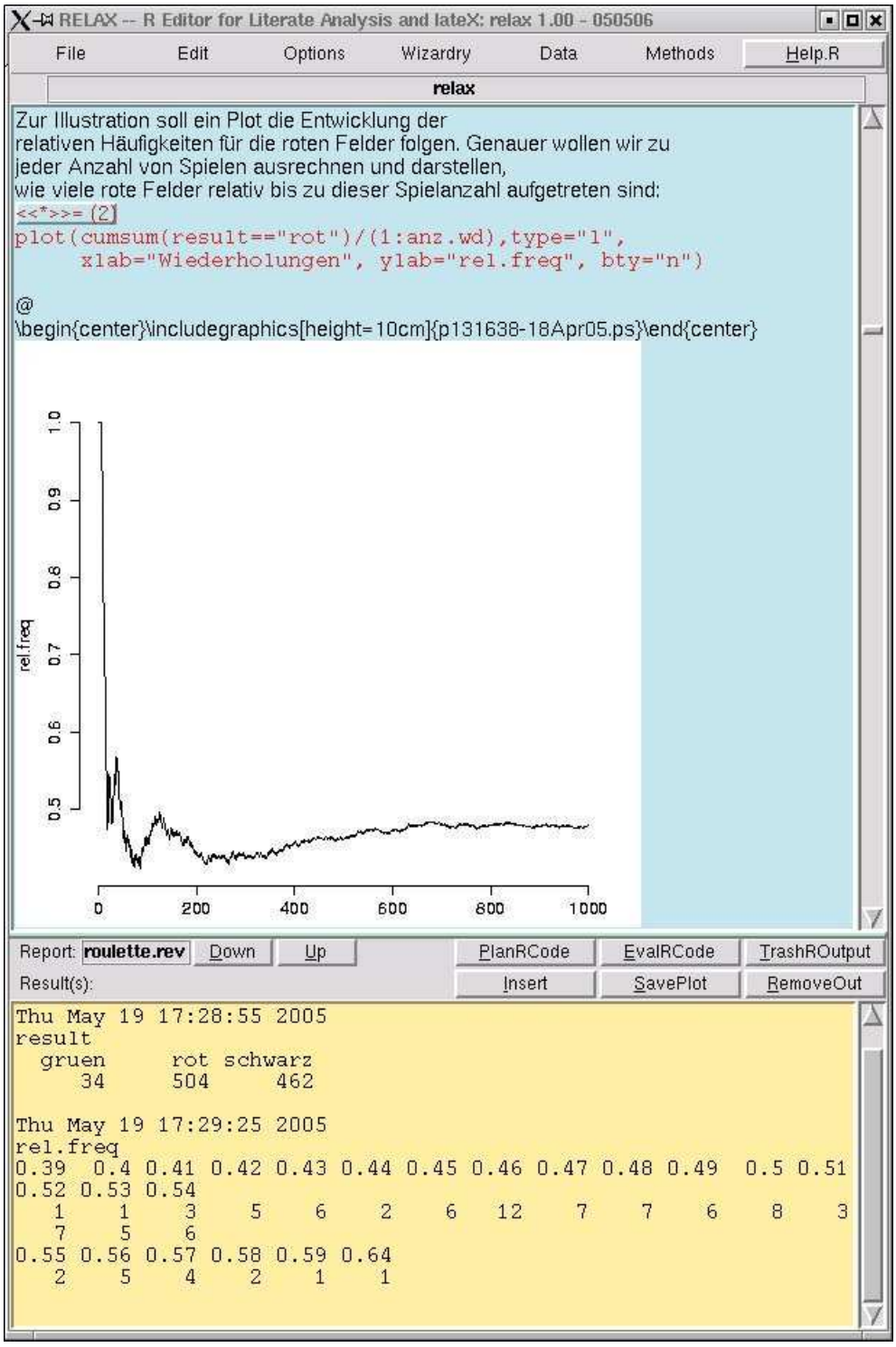


Jetzt wird jeder sagen: Das ist ja alles ganz nett, aber müssen wir das alles noch einmal eintippen, um dasselbe herauszubekommen oder um die vorgeschlagenen Aktionen ein wenig abzuwandeln? Die Antwort lautet: NEIN!!!

Denn mit der Umgebung `relax` lassen sich auch bereits erstellte Reports – wie dieses Skript – hernehmen und die in ihnen beschriebenen Aktionen einzeln aktivieren. Diejenigen, die die Sprache R kennen, können sogar die abgelegten Anweisungen modifizieren und danach starten. Über diese technischen Möglichkeiten handelt der folgende Abschnitt.

3.2 Regeln für das Wiederbeleben von Dokumenten

Fertige Report, die wie oben beschrieben entstanden sind, oder andere geeignete Dokumente lassen sich als zweite Anwendung in das blaue Fenster laden. In diesem Fenster kann man beliebig herumlesen und dann eine Aktion durch Positionierung der Maus auswählen. Die gewählte Aktion lässt sich wieder durch EvalRCode ausführen. Die Ergebnisse werden in dem Output-Fenster eingetragen und können bei Bedarf in das obere Fenster übertragen werden



Welche Operationen sind für die Wiederbelebung von fertigen Report (vielleicht bescheidener: für die exakte Wiederholung von in einem Report abgelegten Aktionen) oder anderen verträglichen Dokumenten erforderlich?

3.2.1 → Aktionswahl

Zuerst gilt es, eine Anweisungssequenz auszuwählen. Dies kann durch Platzierung des Mauszeigers oder mit Hilfe zweier Navigationsknöpfe geschehen.

> Down – bewegt den Mauszeiger zum nachfolgenden Code-Chunk
> Up – bewegt den Mauszeiger zum vorhergehenden Code-Chunk

3.2.2 → Anfügung von Ergebnissen

Manchmal will man nicht das vorliegende Papier durch neue Outputs verändern, sondern diese lieber unten anhängen. Hierfür finden folgender Knopf Verwendung:

> Edit → CopyToEnd – fügt Inhalt aus Output-Fenster unten ans Arbeitsfenster an

3.2.3 → Suche nach besonderen Punkten

Die wesentlichen Dinge sind gesagt, alle weiteren Möglichkeiten gehören unter die Überschrift zusätzliche Serviceleistungen. Zum Beispiel kann man nach Texten oder Chunks suchen. Diese Operationen sind über Edit adressierbar.

> FindReportText – sucht im Arbeitsfenster nach einem Text
> FindReportChunk – unterstützt die Suche nach einen Code-Chunk

3.2.4 → Kopieren

Kopieren kann mit der bekannten Cut-and-Paste-Technik durchgeführt werden. Zusätzlich gibt es den Trick, den Code-Chunk auf dem die Maus residiert durch StrgBild↓ ans Ende des Arbeitsfensters zu bewegen. Weiter sind hinter den Funktionstasten für LaTeX-Freaks ein paar oft benötigte LaTeX-Elemente, die sich ggf. modifizieren lassen, untergebracht. Diese können durch andere Textbausteine ersetzt werden. Hier gilt: Probieren geht über Studieren.

3.2.5 Reflexion

Auf Basis des zweiten Teils der Philosophie lassen sich sehr vielfältige Dinge anstellen. Im einfachsten Fall können durchgeführte Analysen zur Überprüfung wiederholt werden. Interessanter ist natürlich das Probieren alternativer Lösungswege. Außerdem lassen sich beispielsweise speziell für die Lehre Dokumente definieren, die dem Lernenden einen Rahmen zum Experimentieren vorgeben. Je nach Wissensstand ist dann ein Anwender in der Lage, sich mehr oder weniger weit von der Vorlage zu entfernen. Auf diese Weise kann Personen mit unterschiedlichem Vorwissen ein Raum zur kreativen Entfaltung gegeben werden.

4 Roulette — Fortsetzung

Nachdem die Reaktivierungstechnik beschrieben wurde, können wir jetzt hemmungslos schwierige Fragestellung, die auch mit schwierigeren Aktionen einhergehen, so beschreiben, dass selbst ein ungeübter Leser diese am Rechner wiederholen kann. Wir wollen uns zum Beispiel einmal anschauen, wie lang die verschiedenen Serien von roten oder schwarzen Ergebnissen ausfallen. Eine Serie von gleichen Elementen wird in der Statistik als Run (Lauf) bezeichnet. Auch für die Längen

der Runs lassen sich theoretisch Wahrscheinlichkeiten berechnen. An dieser Stelle wollen wir es bei der Beschreibung belassen.

Da wir bereits 10.000 Ergebnisse simuliert haben, können wir diese verwenden. Zunächst wollen wir zur Vereinfachung die grünen Ergebnisse entfernen. Dann werden die Längen der Runs ermittelt und tabelliert. Achtung: An dieser Stelle sollten nur Kenner versuchen, die Anweisungen zu verstehen.

7 ⟨* 1⟩+ ≡
```
result.og<-result[result != "gruen"]
wechselstellen<-which(c(T,0!=diff(result.og=="rot")))
runs <- diff(wechselstellen)
table(runs)
```

```
Mon Apr 18 13:22:11 2005
runs
   1    2    3    4    5    6    7    8    9   10   11   12
2390 1199  617  295  149   79   44   14   13    6    6    1
```

Diese Tabelle wurde auch als Vorbereitung für die gleich folgende letzte Aktion dieses Dokumentes erstellt, mit der der Bogen zu dem Papier von Pearson geschlagen wird. Pearson hatte nämlich eine sehr ähnliche Tabelle erstellt, die sich jedoch auf 4274 Runs bezieht. Durch einfache Arithmetik können wir eine direkte Vergleichbarkeit herstellen. Dazu teilen wir unsere Häufigkeiten durch die Anzahl unserer Runs, multiplizieren das Ergebnis mit 4272 und runden.

8 ⟨* 1⟩+ ≡
```
round(table(runs)/length(runs)*4272)
```

Wir erhalten:

```
Mon Apr 18 13:22:24 2005
runs
   1    2    3    4    5    6    7    8    9   10   11   12
2121 1064  548  262  132   70   39   12   12    5    5    1
```

Die entsprechenden, schon 1894 theoretisch abgeleiteten Zahlen waren:

```
   1    2    3    4    5    6    7    8    9   10   11   12
2137 1068  534  267  134   67   33   17    8    4    2    1
```

Die Übereinstimmung mit unserem virtuellen Roulette ist offenbar ganz gut, jedenfalls viel besser als mit den beobachteten Ergebnissen von 1894:

```
   1    2    3    4    5    6    7    8    9   10   11   12
2462  945  333  220  135   81   43   30   12    7    5    1
```

Hiermit wollen wir es bewenden lassen. Es bleibt zu hoffen, dass der Leser ein wenig neugierig geworden ist und nun danach dürstet, durch weitere Studien reicher zu werden statt durch Spielen ärmer.

5 Übersicht über die wichtigsten Regeln / Operationen

allgemein:	
"@" und «*»=	trennen Text und Aktionen
Arbeitsfenster:	hält das aktuelle Dokument
– PlanNew	erstellt leeren Text- und Code-Chunk
– EvalRCode	aktiviert die Anweisungen, auf die der Mauszeiger zeigt
– TrashOutput	löscht Output aus Reportfenster
– Up	bewegt den Mauszeiger zum vorhergehenden Code-Chunk
– Down	bewegt den Mauszeiger zum nachfolgenden Code-Chunk
Output-Fenster:	nimmt Berechnungsergebnisse auf
– RemoveOut	löscht den Inhalt des Output-Fensters
– Insert	fügt Outputfensterinhalt unter aktuellem Code-Chunk ein
– SavePlot	kopiert Graphiken
File-Menü:	gestattet verschiedene Dateioperationen
– OpenReport	hängt gespeichertes Dokument ans Arbeitsfenster an
– SaveReport	speichert den Report als html- und rev-Datei
– ViewHtml	zeigt gespeicherten Html-Report
– Exit	beendet Arbeit mit `relax`
Edit-Menü:	stellt weitere Bearbeitungsangebote zur Verfügung
– FindReportText	sucht im Arbeitsfenster nach einem Text
– FindReportChunk	unterstützt Suche nach einen Code-Chunk
– CopyToEnd	fügt Inhalt aus Output-Fenster unten ans Arbeitsfenster an
– EditReport	startet einen Editor zum Editieren des aktuellen Dokumentes

6 Eine kleine technische Restarbeit

Es sollten alle, die dieses Papier am Rechner überprüfen wollen, dazu in der Lage sein. Das erfordert, den intern verwendeten Zufallsmechanismus sofort zu Beginn so einzustellen, dass für die Simulation des Roulette für jeden identische Zufallszahlen zum Einsatz kommen. Dieser Wunsch lässt sich durch die Definition eines Start-Moduls – eines Code-Chunks mit dem Namen "start" – erreichen, in dem der Mechanismus initialisiert wird:

9 ⟨*start* 9⟩ ≡

```
set.seed(62)
cat("PDF-File siehe:",
    file.path(.path.package("relax"),"rev","roulette.pdf"))
```