Statistik für Informatiker, SS 2019

Organisatorisches, ein Startbeispiel und eine sehr knappe R-Einführung

Matthias Birkner

http://www.staff.uni-mainz.de/birkner/StatInfo19/

15.4.2019

JOHANNES GUTENBERG UNIVERSITÄT MAINZ

Herzlich willkommen zur Statistik für Informatiker im SS 2019!

1. Organisatorisches
2. Ein Startbeispiel
3. Erste Schritte mit R

Inhalt

Orte, Zeiten, Zulassungskriterien, . . .

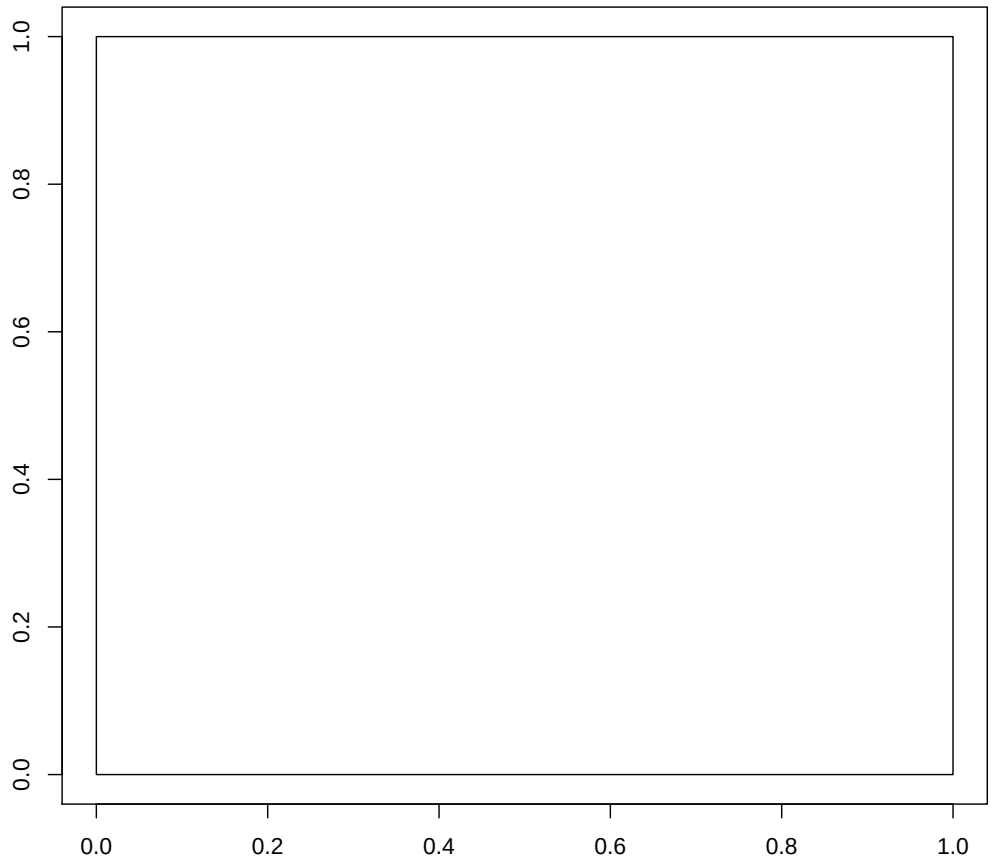
- Mo. 16–18h, HS 20
- Homepage: http://www.staff.uni-mainz.de/birkner/StatInfo19/
  (dort werden die Folien/Notizen, benutzter R-Code/Datensätze, etc. stehen)
- Übungsbetrieb:

  Organisiert von Frederik Klement, Informationen unter https://www.stochastik.mathematik.uni-mainz.de/statistik-fuer-informatik-ss-19/

  (verwenden [auch] R http://www.r-project.org/, dazu später mehr ...)

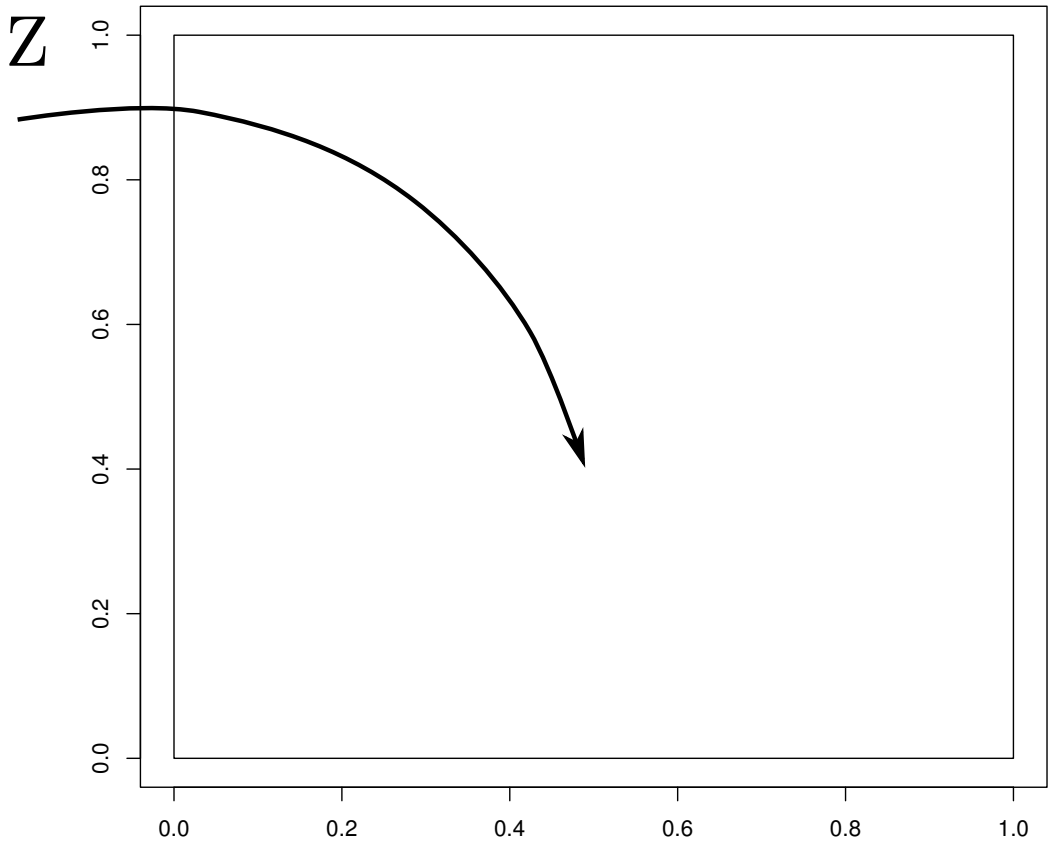
Inhalt

Sei $Z = (X, Y)$ ein „rein zufällig“ gewählter Punkt im Einheitsquadrat $S = \{(x, y) : 0 \leq x, y \leq 1\}$ (geschrieben in kartesischen Koordinaten).



Einheitsquadrat S

Sei $Z = (X, Y)$ ein „rein zufällig“ gewählter Punkt im Einheitsquadrat $S = \{(x, y) : 0 \leq x, y \leq 1\}$ (geschrieben in kartesischen Koordinaten).



Einheitsquadrat S

Für die praktische Implementierung stellen wir uns etwa vor, dass S in (sehr kleine) disjunkte Quadrate („Pixel“) zerlegt wird, und dass wir unter allen möglichen Pixeln eines wählen, wobei jedes dieselbe Chance hat, gezogen zu werden.

Für die praktische Implementierung stellen wir uns etwa vor, dass S in (sehr kleine) disjunkte Quadrate („Pixel“) zerlegt wird, und dass wir unter allen möglichen Pixeln eines wählen, wobei jedes dieselbe Chance hat, gezogen zu werden.

Eine Möglichkeit, dies in `R` zu implementieren, wäre

```
Z <- c(runif(1),runif(1))
```

Hierbei generiert der Befehl `runif(1)` eine (Pseudo-)Zufallszahl im Einheitsintervall $[0,1]$.

Für die praktische Implementierung stellen wir uns etwa vor, dass S in (sehr kleine) disjunkte Quadrate („Pixel“) zerlegt wird, und dass wir unter allen möglichen Pixeln eines wählen, wobei jedes dieselbe Chance hat, gezogen zu werden.

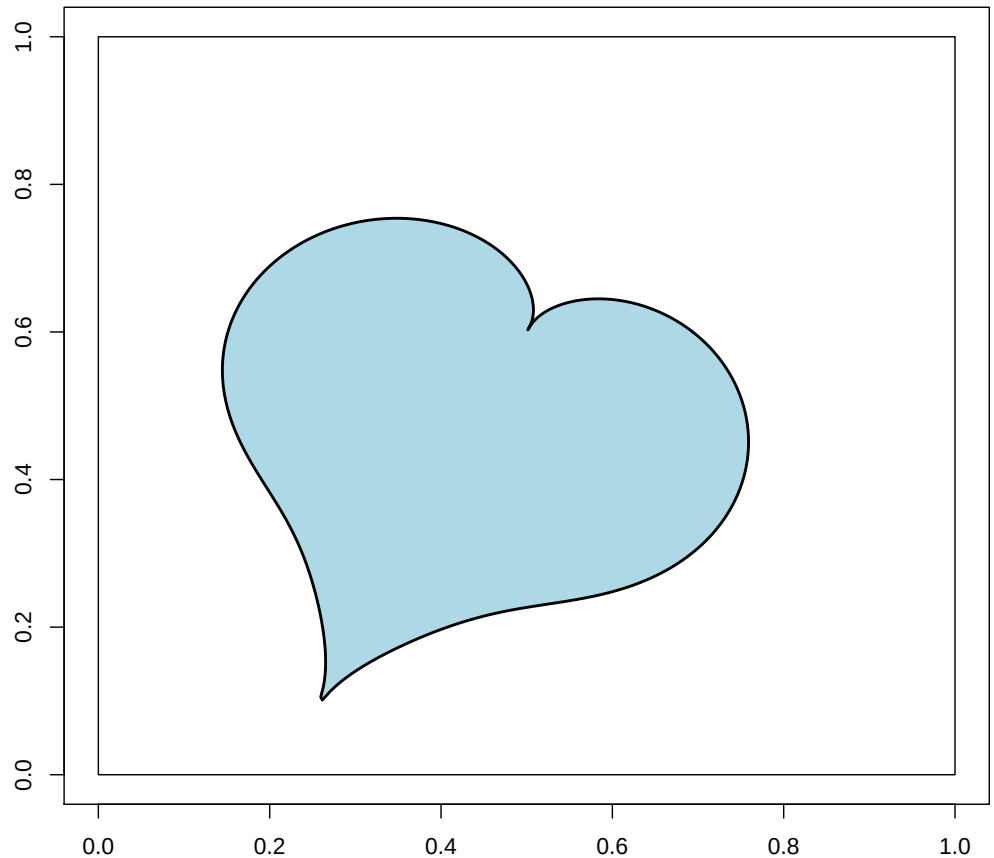
Eine Möglichkeit, dies in `R` zu implementieren, wäre

```
Z <- c(runif(1),runif(1))
```

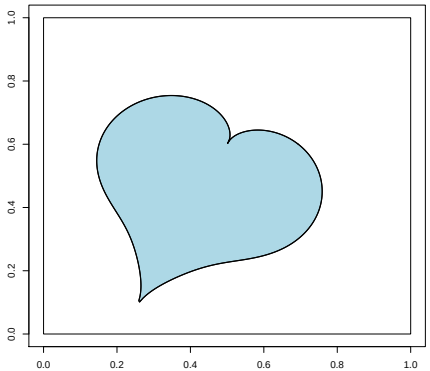
Hierbei generiert der Befehl `runif(1)` eine (Pseudo-)Zufallszahl im Einheitsintervall $[0,1]$.

Wir nennen Z eine *Zufallsvariable*, die möglichen Werte S ihren *Wertebereich*.

Sei $B \subset S$ eine gewisse Teilmenge.



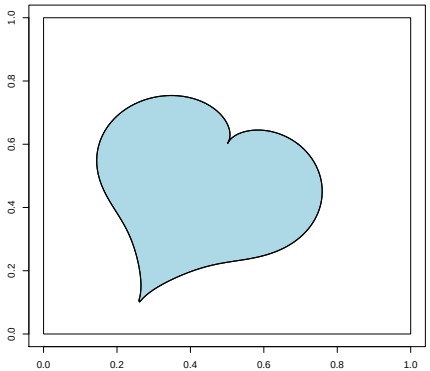
Einheitsquadrat S und Teilmenge B



Dann können wir das *Ereignis*

$$\{Z \in B\}$$

(ausgesprochen als „Z nimmt einen Wert in B an“) betrachten.

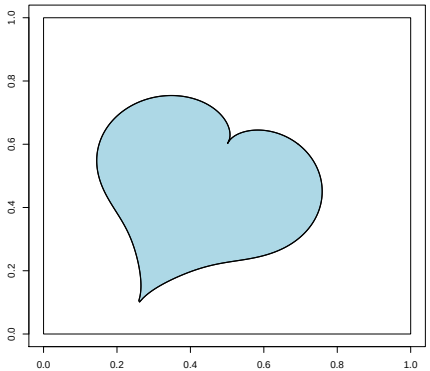


Dann können wir das *Ereignis*

$$\{Z \in B\}$$

(ausgesprochen als „Z nimmt einen Wert in B an“) betrachten.

Je nach Ausgang des zufälligen Experiments
(für das Computerexperiment hängt dies vom internen Zustand des Pseudo-Zufallsgenerators und damit implizit vom gewählten “random seed” ab)
wird Z in B liegen oder nicht



Dann können wir das *Ereignis*

$$\{Z \in B\}$$

(ausgesprochen als „Z nimmt einen Wert in B an“) betrachten.

Je nach Ausgang des zufälligen Experiments
(für das Computerexperiment hängt dies vom internen Zustand des Pseudo-Zufallsgenerators und damit implizit vom gewählten “random seed” ab)
wird Z in B liegen oder nicht,

d.h. das Ereignis $\{Z \in B\}$ tritt ein oder nicht.

Die *Wahrscheinlichkeit* des Ereignisses $\{Z \in B\}$ ist plausiblerweise

$$P\big(\{Z \in B\}\big) = \frac{\text{Anzahl Pixel in } B}{\text{Anzahl Pixel in } S} = \frac{\text{Fläche von } B}{\text{Fläche von } S} = \frac{\text{Fläche von } B}{1}$$

(das P erinnert an Englisch “probability” oder Französisch «probabilité», die natürlich beide vom Lateinischen Wort probabilitas abstammen).

Sei $\mathbf{1}_B$ die *Indikatorfunktion* von B, d.h. für $z \in S$ ist

$$\mathbf{1}_B(z) = \begin{cases} 1, & z \in B, \\ 0, & z \notin B. \end{cases}$$

Sei $\mathbf{1}_B$ die *Indikatorfunktion* von B, d.h. für $z \in S$ ist

$$\mathbf{1}_B(z) = \begin{cases} 1, & z \in B, \\ 0, & z \notin B. \end{cases}$$

Wir können eine weitere / neue Zufallsvariable $W := \mathbf{1}_B(Z)$ mit Wertebereich $\{0, 1\}$ bilden: W ist gleich 1, wenn der Wert von Z in B liegt, sonst gleich 0 (man nennt eine solche Zufallsvariable auch eine *Indikatorvariable*).

Sei $\mathbf{1}_B$ die *Indikatorfunktion* von B, d.h. für $z \in S$ ist

$$\mathbf{1}_B(z) = \begin{cases} 1, & z \in B, \\ 0, & z \notin B. \end{cases}$$

Wir können eine weitere / neue Zufallsvariable $W := \mathbf{1}_B(Z)$ mit Wertebereich $\{0, 1\}$ bilden: W ist gleich 1, wenn der Wert von Z in B liegt, sonst gleich 0 (man nennt eine solche Zufallsvariable auch eine *Indikatorvariable*).

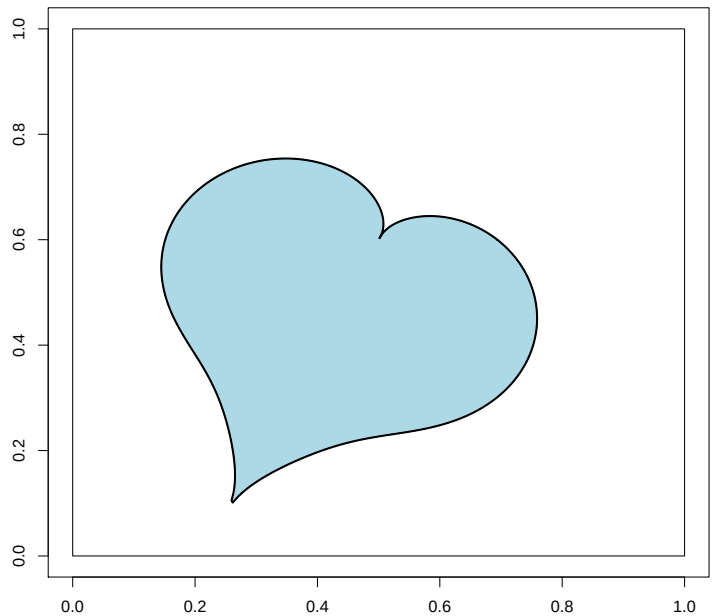
Mit Ereignissen ausgesprochen also

$$\{W = 1\} = \{Z \in B\}, \quad \{W = 0\} = \{Z \in B^c\}$$

(hierbei ist $B^c = S \setminus B = \{z \in S : z \notin B\}$ die Komplementmenge von B) und somit auch

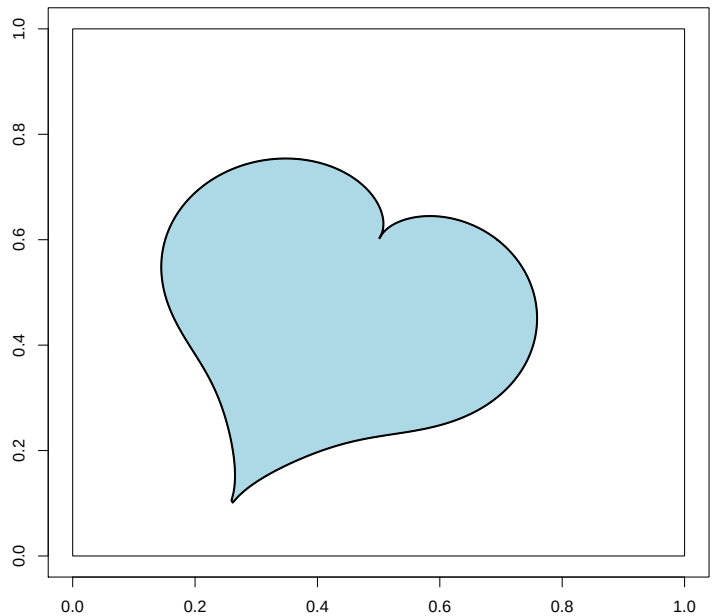
$$P\big(\{W = 1\}\big) = P\big(\{Z \in B\}\big) = \frac{\text{Fläche von } B}{1}.$$

Anwendung: eine einfache Monte Carlo-Methode zur Integration



Sei $p = P\big(\{Z \in B\}\big)$ = Fläche von B.

Anwendung: eine einfache Monte Carlo-Methode zur Integration



Sei $p = P\big(\{Z \in B\}\big)$ = Fläche von B.

Wir können den Zufall verwenden, um p (wenigstens approximativ) zu bestimmen.

Stellen wir uns vor, wir wiederholen obiges Zufallsexperiment n-mal, wobei der Zufall „jedes mal neu wirkt“.

Stellen wir uns vor, wir wiederholen obiges Zufallsexperiment n-mal, wobei der Zufall „jedes mal neu wirkt“.

Seien $Z_1, Z_2, \ldots, Z_n$ die Ergebnisse dieser n Experimente (im Jargon: die Z_i sind *unabhängige Kopien* von Z), setze $W_i := \mathbf{1}_B(Z_i)$.

Stellen wir uns vor, wir wiederholen obiges Zufallsexperiment n-mal, wobei der Zufall „jedes mal neu wirkt“.

Seien $Z_1, Z_2, \ldots, Z_n$ die Ergebnisse dieser n Experimente (im Jargon: die Z_i sind *unabhängige Kopien* von Z), setze $W_i := \mathbf{1}_B(Z_i)$.

Die Zufallsvariable

$$\widetilde{M}_n := \sum_{i=1}^{n} W_i$$

gibt an, wieviele der n zufällig gewählten Punkte in B gelandet sind.

Stellen wir uns vor, wir wiederholen obiges Zufallsexperiment n-mal, wobei der Zufall „jedes mal neu wirkt“.

Seien $Z_1, Z_2, \ldots, Z_n$ die Ergebnisse dieser n Experimente (im Jargon: die Z_i sind *unabhängige Kopien* von Z), setze $W_i := \mathbf{1}_B(Z_i)$.

Die Zufallsvariable

$$\widetilde{M}_n := \sum_{i=1}^{n} W_i$$

gibt an, wieviele der n zufällig gewählten Punkte in B gelandet sind.

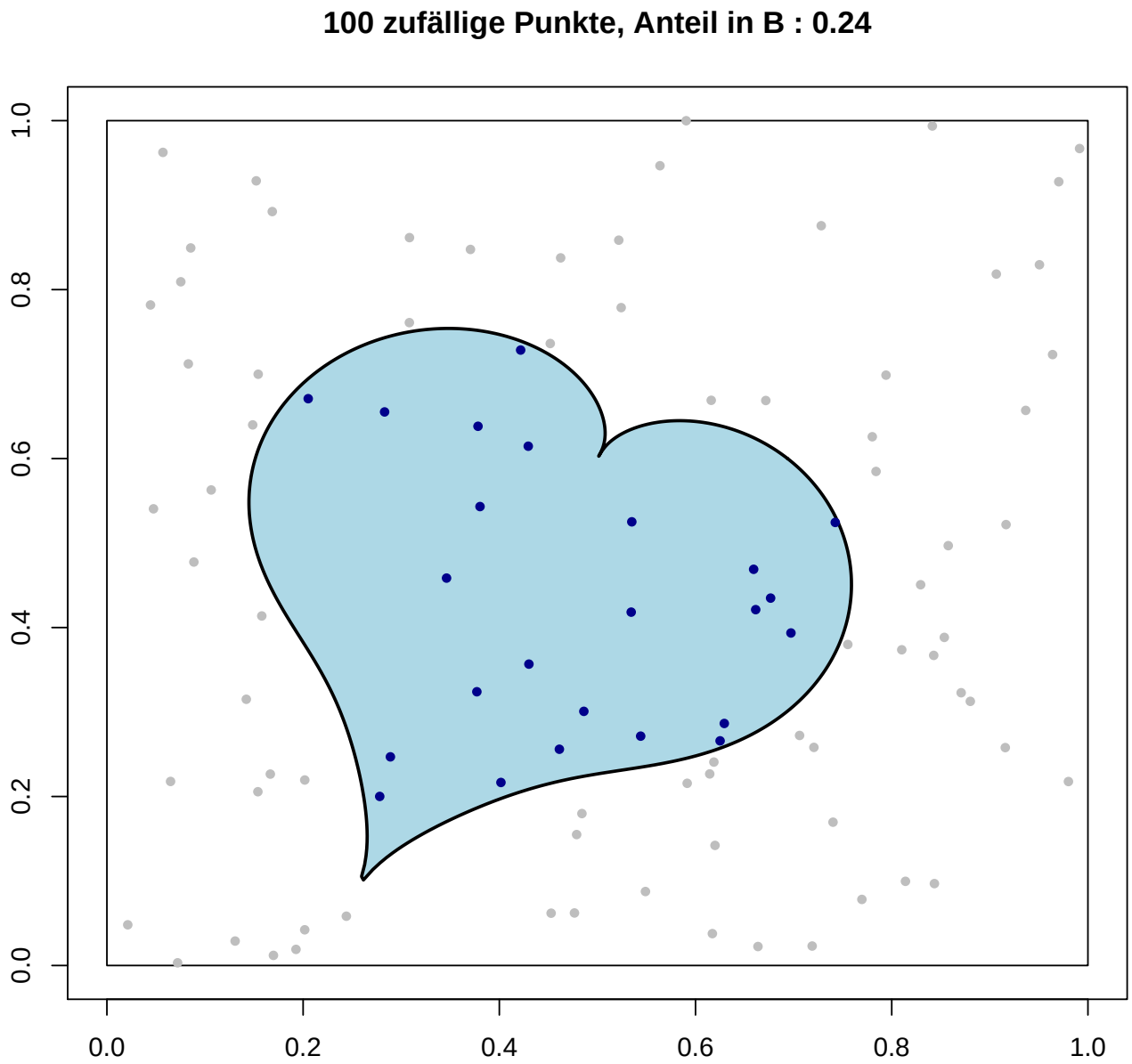
Der empirische Anteil

$$M_n := \frac{1}{n} \sum_{i=1}^{n} W_i \quad \left(= \frac{1}{n} \widetilde{M}_n \right)$$

ist ein (plausibler) „Schätzwert“ für p.

(Der Wertebereich von M_n ist offenbar $\{0, \frac{1}{n}, \frac{2}{n}, \ldots \frac{n-1}{n}, 1\}$.)

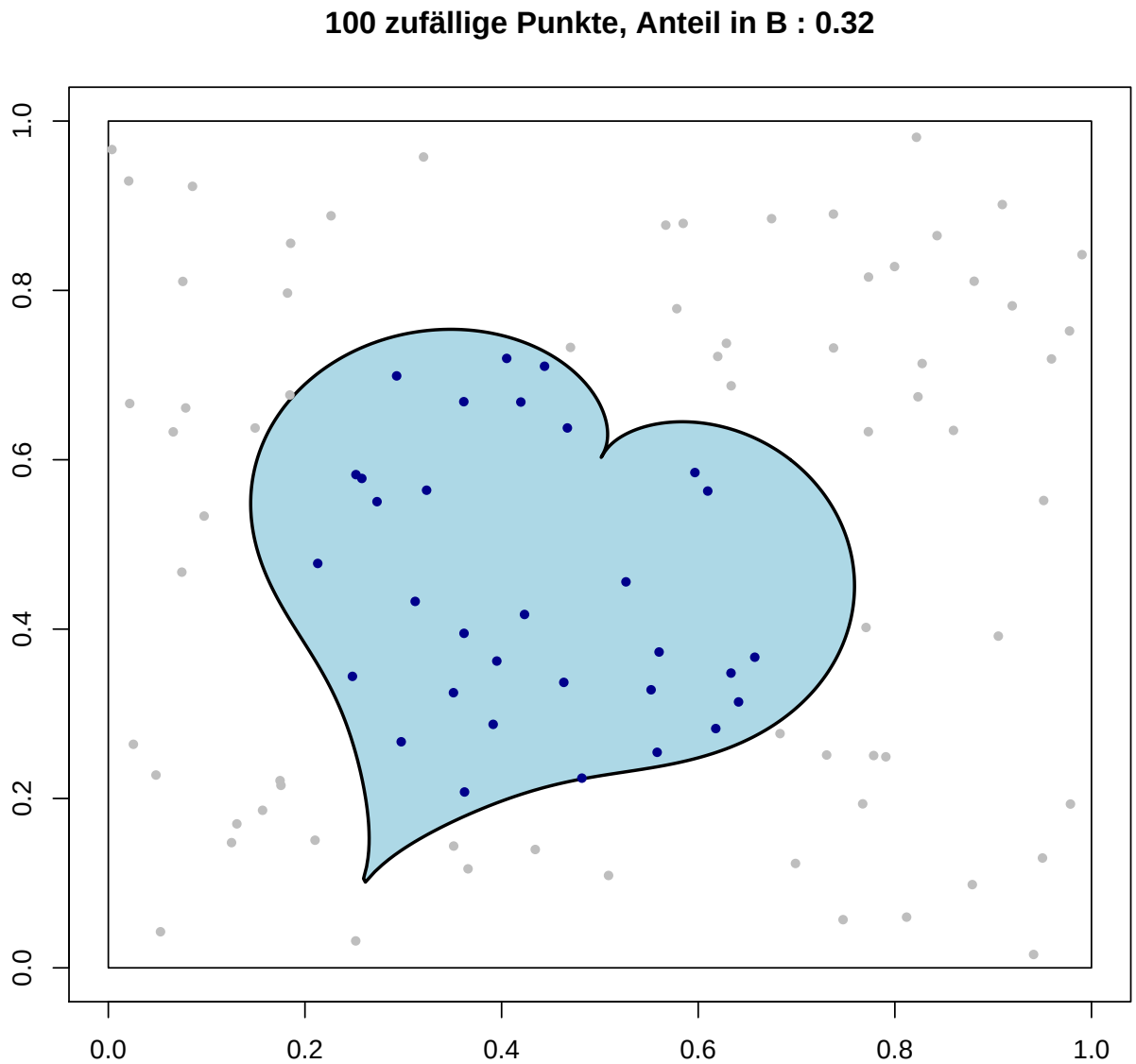
Visualisierung eines solchen Versuchs für $n = 100$



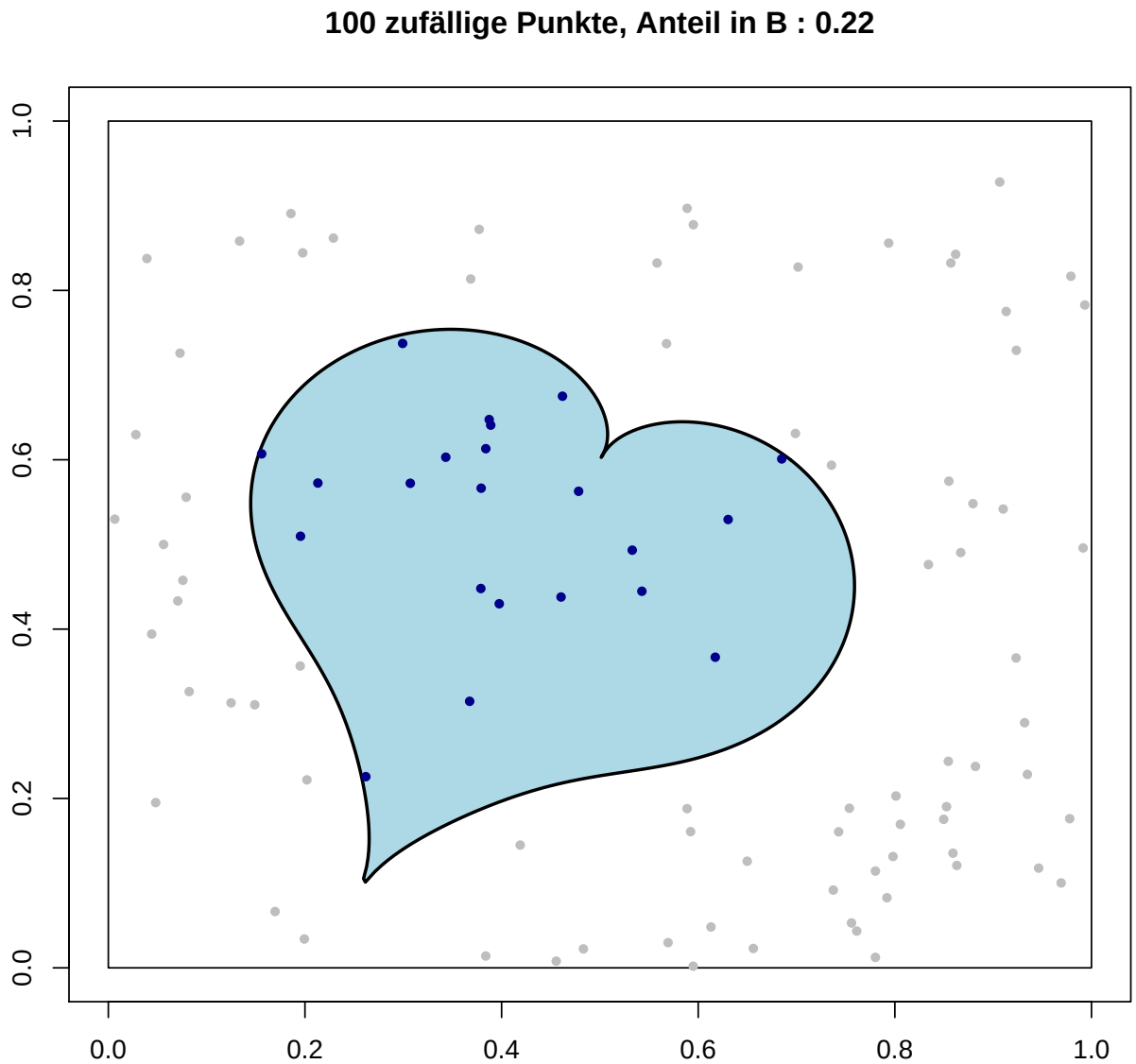
Bemerkung. Man kann dies (nämlich die Simulation von M_n und Ausgabe des berechneten Werts) als ein einfaches Beispiel eines sogenannten Monte Carlo-Algorithmus betrachten:

Das Verfahren bestimmt zwar nicht genau den Wert von p, wir werden aber quantifizieren können, wie (un-)wahrscheinlich es ist, dass es um mehr als ein vorgegebenes ε „daneben liegt“.

Zwei weitere Wiederholungen mit jeweils $n = 100$ zufälligen Punkten in $[0, 1]^2$



Zwei weitere Wiederholungen mit jeweils $n = 100$ zufälligen Punkten in $[0,1]^2$



Verteilung von $\widetilde{M}_n$

Die Zufallsvariable

$$\widetilde{M}_n := \sum_{i=1}^{n} \mathbf{1}_B(Z_i)$$

gibt an, wieviele der n zufällig gewählten Punkte $Z_1, \ldots, Z_n$ in B gelandet sind.

Verteilung von $\widetilde{M}_n$

Die Zufallsvariable

$$\widetilde{M}_n := \sum_{i=1}^{n} \mathbf{1}_B(Z_i)$$

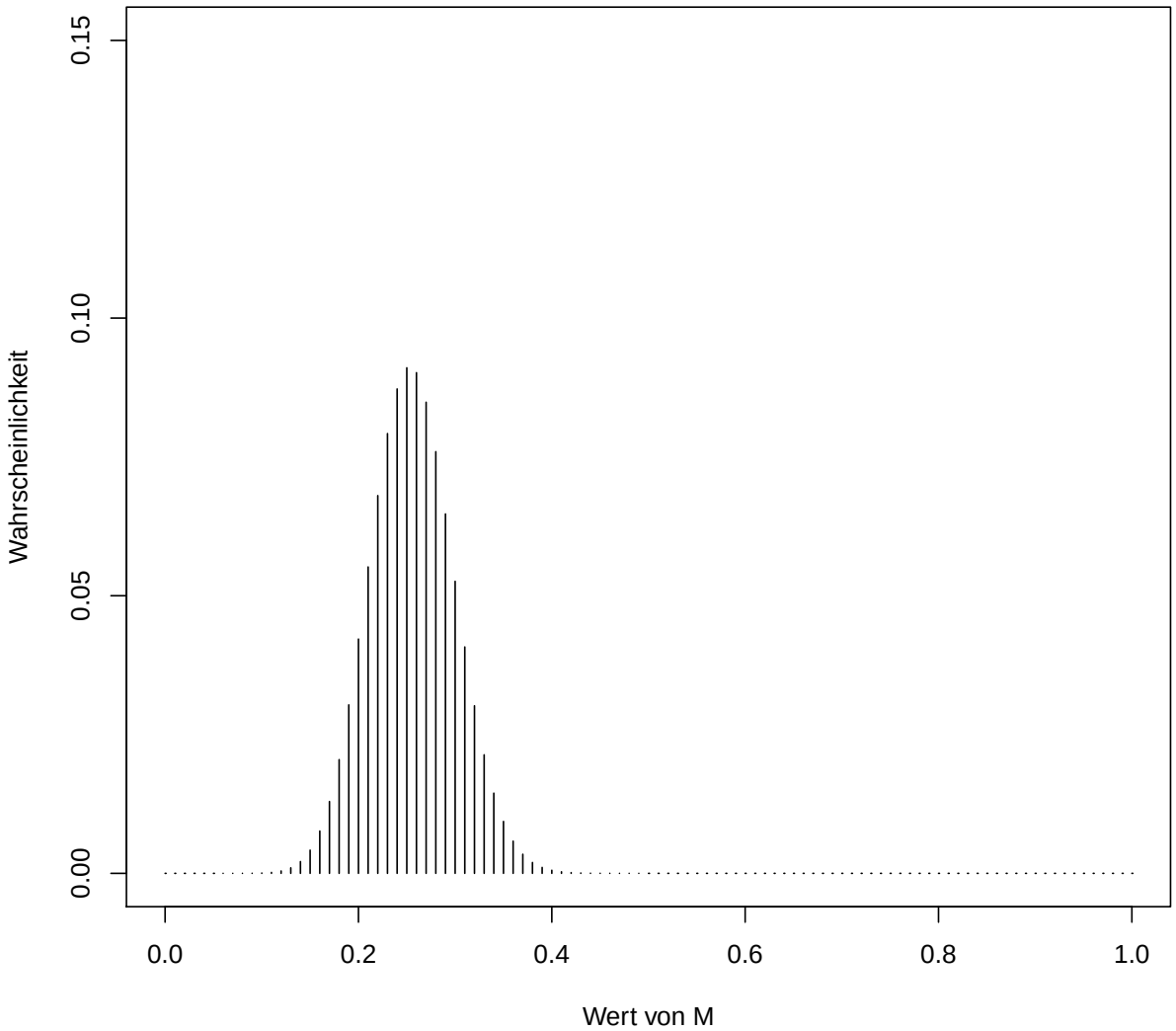
gibt an, wieviele der n zufällig gewählten Punkte $Z_1, \ldots, Z_n$ in B gelandet sind.

$\widetilde{M}_n$ ist Binomial-verteilt mit Parametern n und p (abgekürzt $\mathrm{Bin}_{n,p}$-verteilt), d.h.

$$P\big(\{\widetilde{M}_n = k\}\big) = \binom{n}{k} p^k (1-p)^{n-k} \quad \text{für } k = 0, 1, \ldots, n.$$

(Wir werden dies später noch genauer betrachten).

Damit ergibt sich für die Verteilungsgewichte von M_{100} folgendes (Balken-)Diagramm
(an die Stelle k/n zeichnen wir einen Balken der Höhe $P(\{M_n = \frac{k}{n}\})$):



Bemerkung. Tatsächlich ist der Wert von p in diesem Beispiel

$$p = \int_{-\pi}^{\pi} \frac{1}{98} \left(2 - 2\sin(w) + \frac{\sin(w)\sqrt{|\cos(w)|}}{\sin(w) + 7/5} \right)^2 dw \approx 0.25557221...$$

Bemerkung. Tatsächlich ist der Wert von p in diesem Beispiel

$$p = \int_{-\pi}^{\pi} \frac{1}{98} \left(2 - 2\sin(w) + \frac{\sin(w)\sqrt{|\cos(w)|}}{\sin(w) + 7/5} \right)^2 dw \approx 0.25557221...$$

Denn für

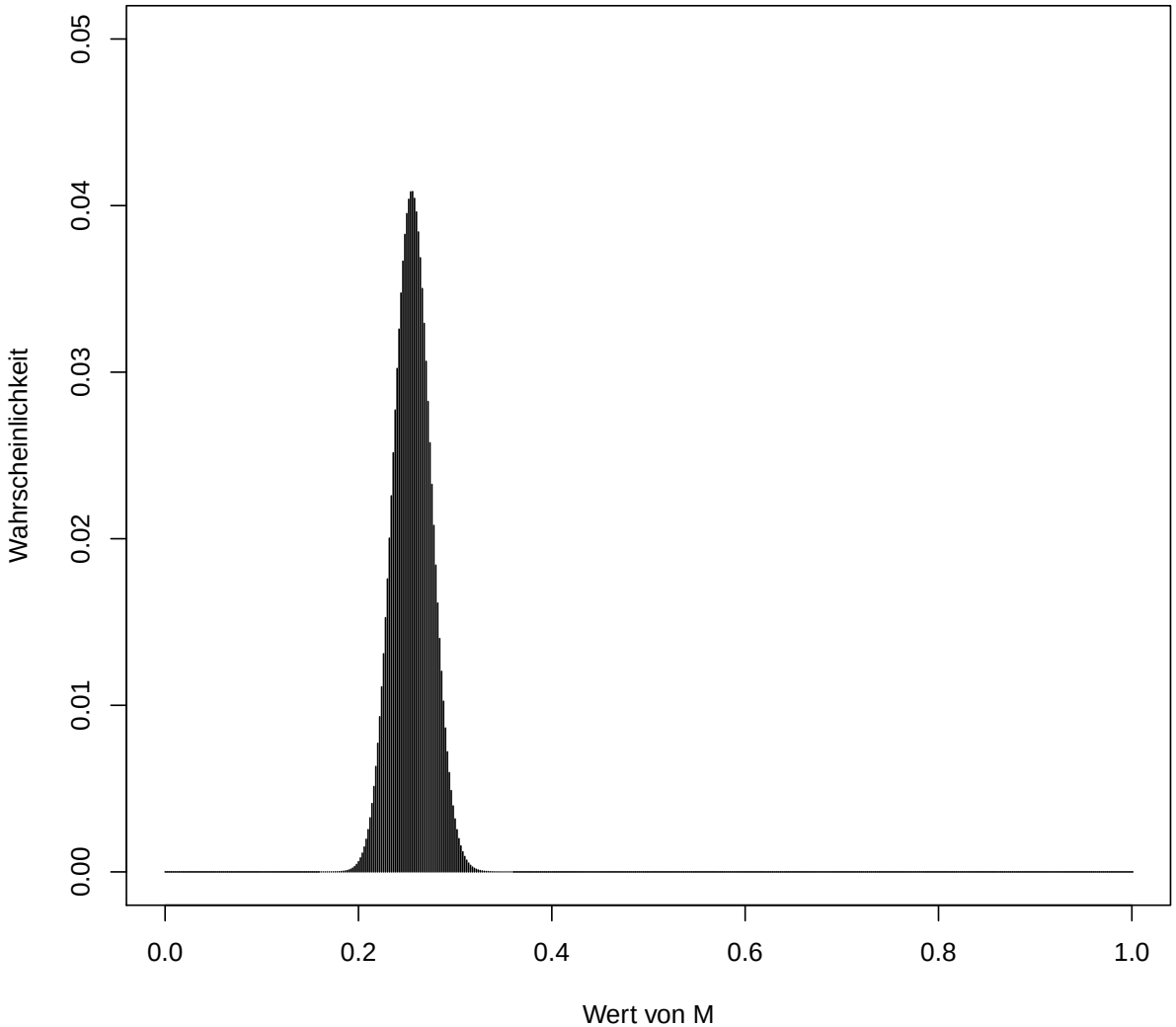
$$B = \left\{ (x, y) \in \mathbb{R}^2 : (x^2 + y^2)^{1/2} \leq f(\arg(x, y)) \right\}$$

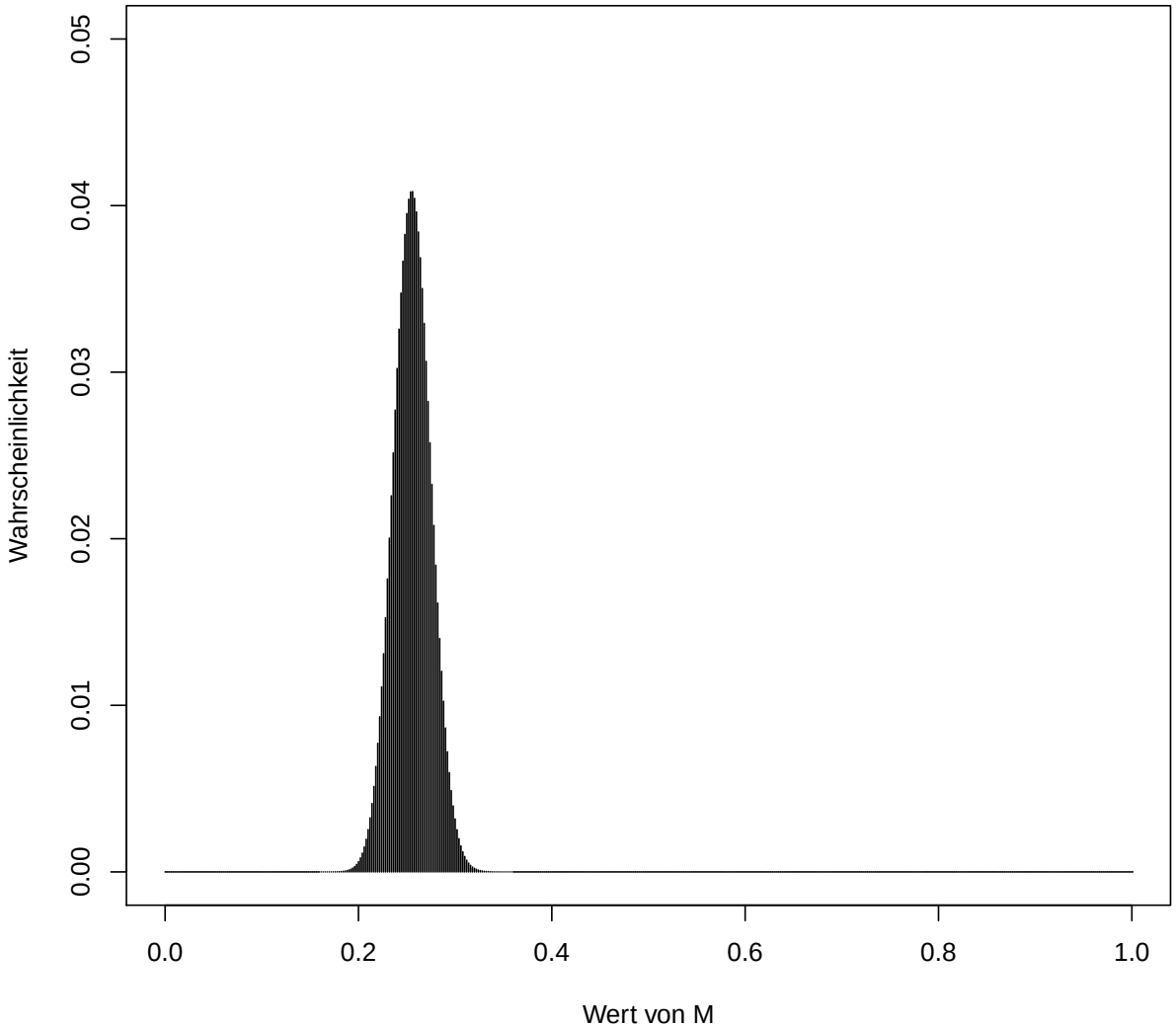
wo $f : [-\pi, \pi] \to \mathbb{R}_+$ und
$\arg(x, y) = \operatorname{sign}(x) \arcsin(y/\sqrt{x^2 + y^2}) + \pi \operatorname{sign}(y) \mathbf{1}_{(-\infty, 0)}(x)$ (dies ist der Winkel, den der Strahl vom Ursprung durch den Punkt (x, y) mit der x-Achse einschließt) gilt

$$\text{Fläche von } B = \int_{-\pi}^{\pi} \frac{1}{2} f(w)^2 \, dw$$

Durch Erhöhung von n können wir die Genauigkeit der Approximation erhöhen.

Wir werden dies später quantifizieren können, für den Moment betrachten wir das Histogramm der Verteilungsgewichte von M_{500} .

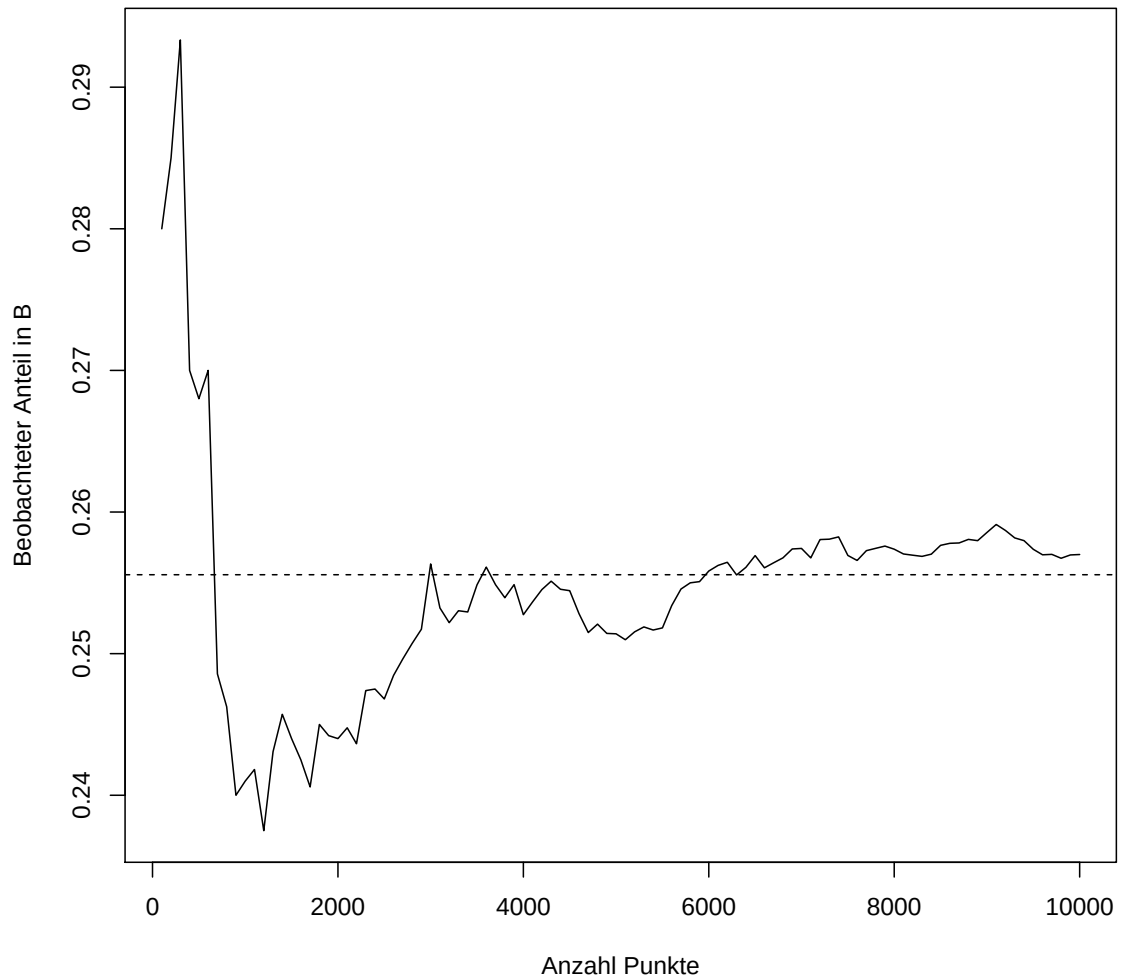

Wahrscheinlichkeit
0.00
0.01
0.02
0.03
0.04
0.05
0.0
0.2
0.4
0.6
0.8
1.0
Wert von M



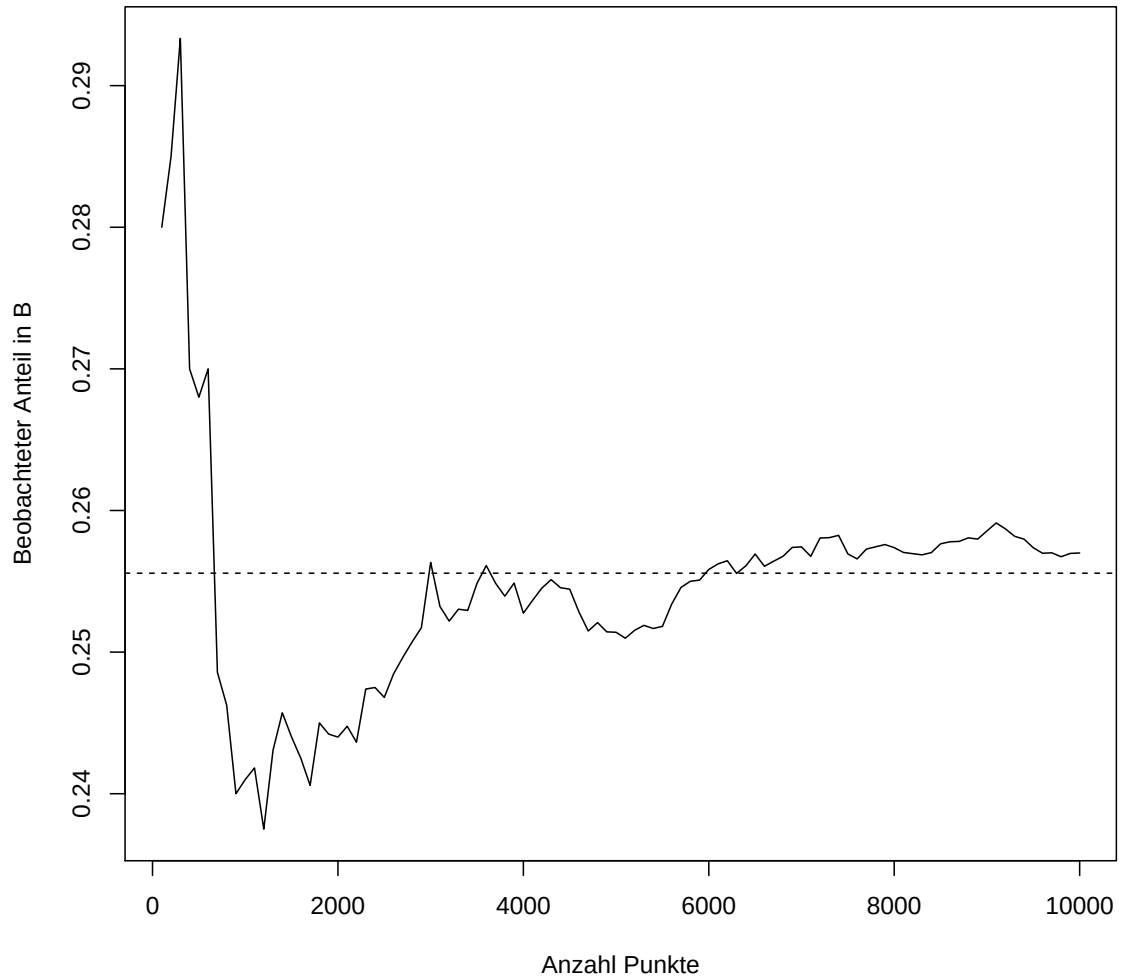
(Wir sehen: Die Verteilung von M_{500} ist deutlich stärker um p konzentriert als die von M_{100}).

Betrachten wir zum Abschluss (für eine Folge von $Z_1, Z_2, \ldots, Z_{10000}$) die Folge der M_n als Funktion von n :

Betrachten wir zum Abschluss (für eine Folge von $Z_1, Z_2, \ldots, Z_{10000}$) die Folge der M_n als Funktion von n :



(die gestrichelte Linie ist der Wert p)



Wir werden im Lauf der Vorlesung sehen: Für $n \to \infty$ konvergiert M_n (in geeignetem Sinne) gegen p. Dies folgt aus dem sogenannten *Gesetz der großen Zahlen*.

Inhalt

Was ist R?

- R ist eine für die Statistik und für stochastische Simulation entwickelte Programmiersprache, zudem sind viele statistische Standardverfahren bereits in R implementiert oder als Zusatzpaket verfügbar.
- R hat eine sehr aktive Benutzer- und Entwicklergemeinde, die nahezu alle Bereiche der Statistik und viele Anwendungsbereiche (z.B. Populationsgenetik, Finanzmathematik) überdeckt.
- R ist frei verfügbar unter der GNU general public license, für (nahezu) alle Rechnerarchitekturen erhältlich: http://www.r-project.org/

Literatur zu R

„Standardreferenz“:

W.N. Venables et al, *An Introduction to R*,
http://cran.r-project.org/manuals.html

Zur ersten Einführung auch die Hinweise zu R, „nullte Schritte“ auf der Vorlesungs-Homepage.

Weiterführend (z.T. wesentlich über den Stoff der Vorlesung hinaus):

- Günther Sawitzki, *Einführung in R*,
  http://sintro.r-forge.r-project.org/
- William N. Venables, Brian D. Ripley, *Modern applied statistics with S* („Standardwerk“, UB Lehrbuchsammlung)
- Lothar Sachs and Jürgen Hedderich, *Angewandte Statistik – Methodensammlung mit R* (E-Book, UB)
- Christine Duller, *Einführung in die nichtparametrische Statistik mit SAS und R : ein anwendungsorientiertes Lehr- und Arbeitsbuch* (E-Book, UB)
- Helge Toutenburg, Christian Heumann, *Deskriptive Statistik : Eine Einführung in Methoden und Anwendungen mit R und SPSS* (E-Book, UB)
- Uwe Ligges, *Programmieren mit R* (E-Book, UB)

R installieren, starten, anhalten

Installation: Windows, Mac OS: Binaries von http://www.r-project.org/ (siehe Link Download, Packages, CRAN dort)

Linux: Für die meisten Distributionen gibt es fertige Pakete (entweder bei CRAN oder im Repository der Distribution)
Bei Fragen oder Problemen: Übungsgruppen

R starten: Windows, Mac OS: Icon (ggf. aus Menu) anklicken, Linux/Unix: `> R` auf einer Konsole (oder z.B. mit ESS aus Emacs heraus).

R beenden: q() (fragt, ob Daten gespeichert werden sollen)
laufende Rechnungen unterbrechen: CTRL-C

Hinweis: RStudio https://www.rstudio.com/ bietet eine integrierte Entwicklungsumgebung für R [für Privatanwender frei verfügbar].

Einige einfache (und wichtige) R-Befehle

Mathematische Operatoren und Funktionen:
`+, -, *, /, ^, exp, sin, log,` etc.

Hilfe aufrufen:
`help(Befehl)` oder `?Befehl`

Online-Hilfe starten:
`help.start()`

Variable einen Wert zuweisen:
```
x <- 5
y <- "Zeichenkette"
z <- TRUE
w <- -2+7.5i
```

Vektor erzeugen, Elementzugriff:
`v <- c(1,2,3.1415,-17); v[3]`

Rezyklierungs-Regel bei Vektoren:
`v+2` liefert `3,4,5.1415,-15`

Liste erzeugen, Elementzugriff:
```
l <-list(1.2,"text",-5,FALSE);
l[[2]]
```

(Pseudo-)Zufallszahl generieren:
`runif`

Demos starten (und bestaunen):
`demo()`

Einlesen eines Skripts:
`source`

Ausgabe in Datei umlenken:
`sink`

Grafikausgaben:
`plot,...`

Grafik„geräte“ öffnen:
`x11, postscript, pdf, dev.copy2eps, dev.copy2pdf, ...`

Grafikparameter setzen: `par`