Datenanalyse mit R für Administratoren

Beispiele aus der Praxis

Stefan Möding

23. August 2014

FrOSCon
Free and Open Source Software Conference

Inhalt

- R in aller Kürze
- Beispiel 1: Antwortzeiten eines Tomcat
- Beispiel 2: Prognosen mit RRD
- Beispiel 3: Universal Scalability Law

R in aller Kürze

R und RStudio

R

- Programmiersprache & -umgebung für statistisches Rechnen und Grafik
- GNU General Public License
- UNIX/Linux, MacOS, Windows
- `http://www.r-project.org`

RStudio

- Integrated Development Environment
- GNU Affero General Public License
- Linux, MacOS, Windows
- `http://www.rstudio.com`

R in aller Kürze

Stärken

Stärken von R

- „Code“ statt „Click“
  - nachvollziehbar
  - reproduzierbar
  - versionierbar
- Schnittstellen
  - Import: CSV, XML, Excel, Access, Oracle, PostgreSQL, MySQL
  - Export: PNG, JPG, SVG, PDF, HTML, Markdown, LaTeX
- CRAN — Comprehensive R Archive Network
  - $\approx$ 5800 Pakete (August 2014)

Beispiel 1: Antwortzeiten eines Tomcat

Aufgabenstellung

Antwortzeiten eines Apache Tomcat

- Typische Werte
- Verteilung
- Ausreißer

Beispiel 1: Antwortzeiten eines Tomcat

Logfile als Datenquelle

```
<Valve className="org.apache.catalina.valves.AccessLogValve"
       resolveHosts="false" directory="logs"
       prefix="tomcat_access_log." suffix=".txt"
       pattern="%h %l %u %t &quot;%r&quot; %s %b %D" />
```

Verarbeitungsdauer des Requests (msec)

Beispiel 1: Antwortzeiten eines Tomcat

Datenimport

```
# Load data from file
tomcat <- read.table(file = "data/tomcat-01.log.gz")
```

Data Frame: Zeilen & Spalten

Spaltennamen anpassen

```
names(tomcat) <- c("ip", "rusr", "ausr", "time", "tz", "request",
                   "status", "size", "service_time")
```

Beispiel 1: Antwortzeiten eines Tomcat

Einfache Kennzahlen

Wertebereich

```
range(tomcat$service_time)

[1] 38 78
```

Mittelwert

```
mean(tomcat$service_time)

[1] 43.96
```

Beispiel 1: Antwortzeiten eines Tomcat

Einfache Kennzahlen

Median

```
median(tomcat$service_time)

[1] 43
```

Streuung

```
IQR(tomcat$service_time)

[1] 4
```

Quantile (. . . in 99% der Fälle liegt die Antwortzeit unter . . .)

```
quantile(tomcat$service_time, probs = 0.99)

  99%
73.84
```

Beispiel 1: Antwortzeiten eines Tomcat

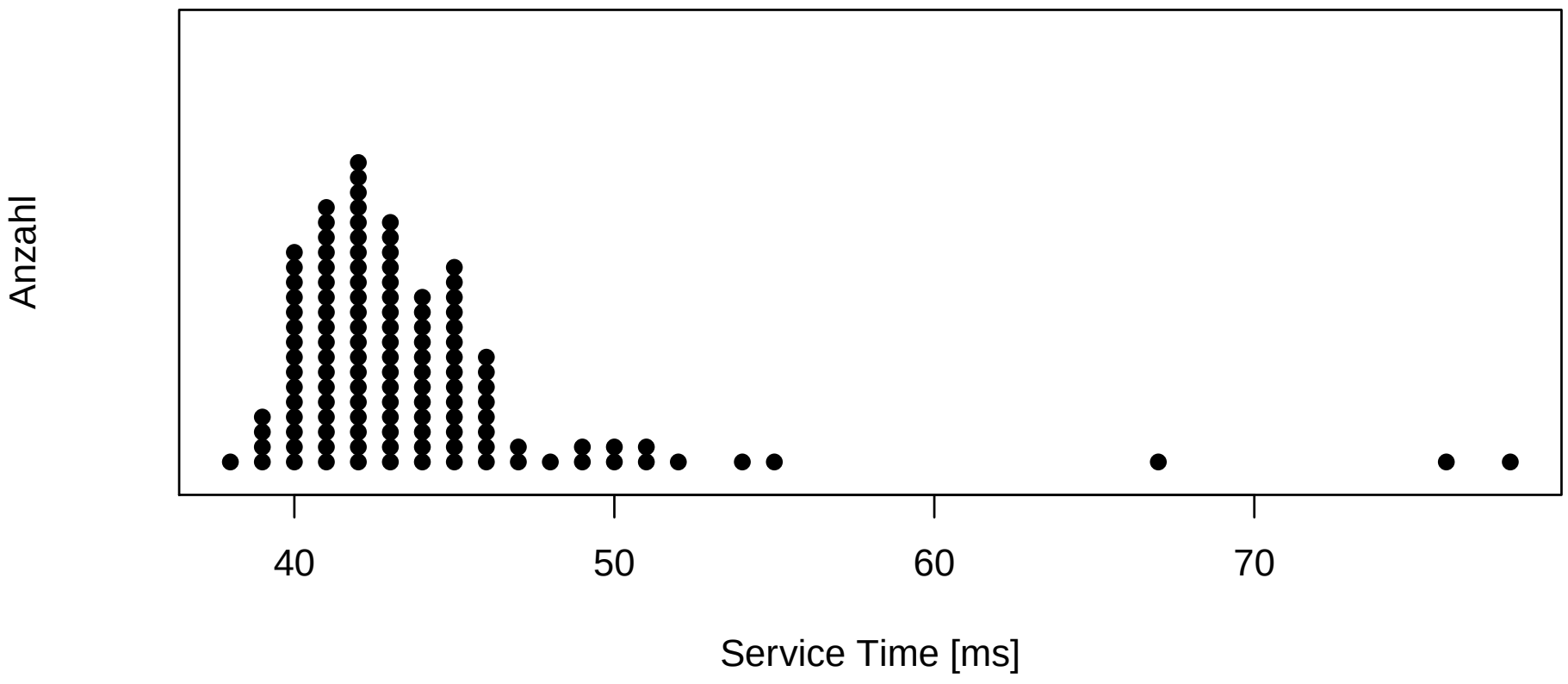
Häufigkeitsverteilung: Stamm-Blatt-Diagramm

```
stem(tomcat$service_time, scale = 0.5, width = 70)


  The decimal point is 1 digit(s) to the right of the |

  3 | 89999
  4 | 000000000000001111111111111111122222222222222222223333+13
  4 | 55555555555556666666677899
  5 | 001124
  5 | 5
  6 |
  6 | 7
  7 |
  7 | 68
```

Beispiel 1: Antwortzeiten eines Tomcat

Häufigkeitsverteilung: Stripchart

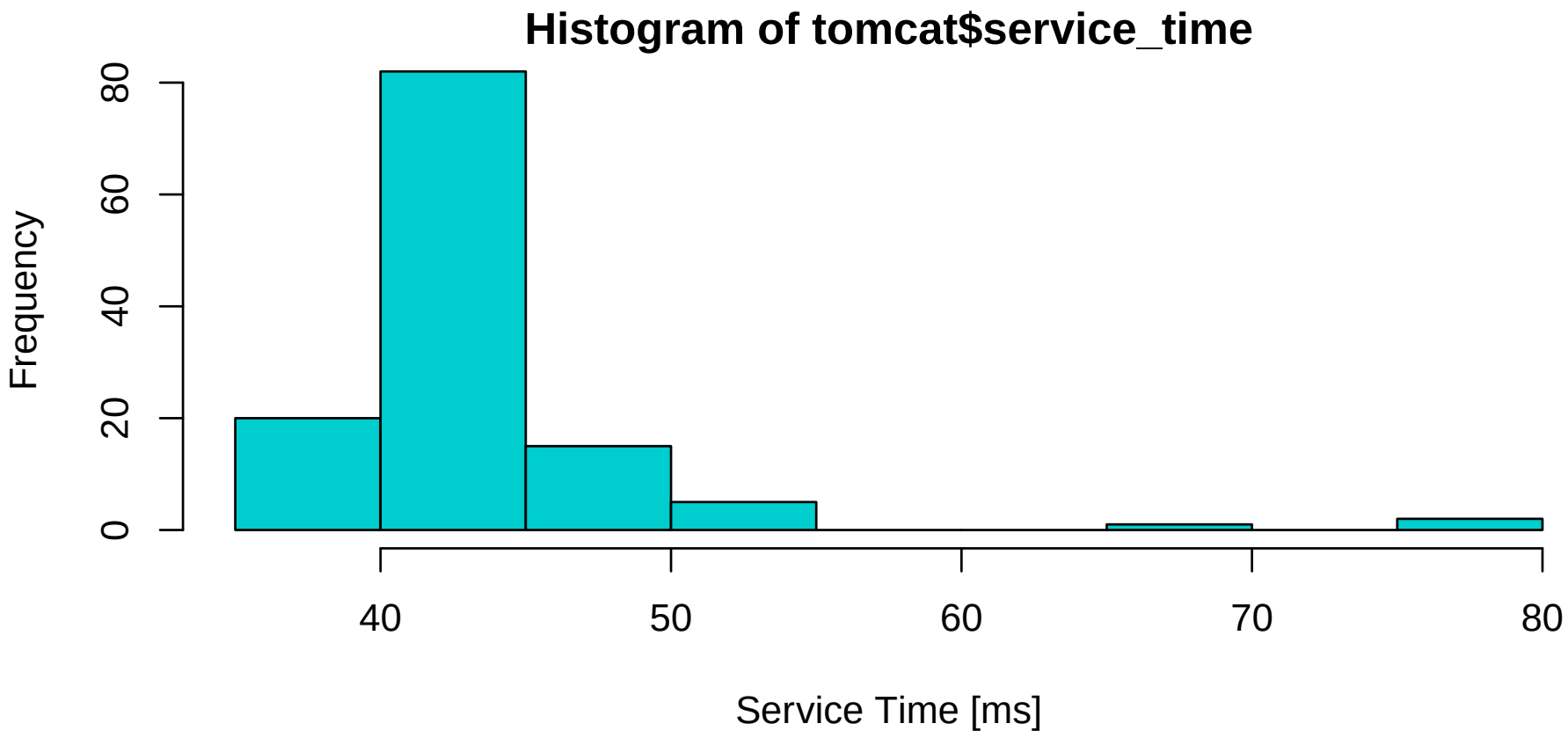
```
stripchart(tomcat$service_time, method = "stack", ylim = c(0, 30),
           xlab = "Service Time [ms]", ylab = "Anzahl", pch = 16)
```



Beispiel 1: Antwortzeiten eines Tomcat

Häufigkeitsverteilung: Histogramm

```
hist(tomcat$service_time, xlab = "Service Time [ms]", col = "cyan3")
```

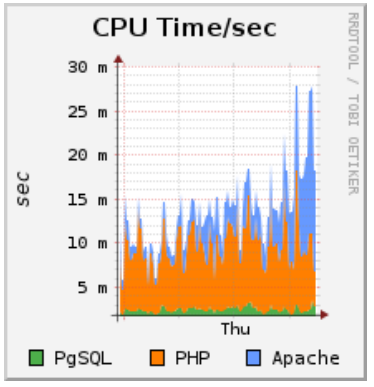


Beispiel 2: Prognosen mit RRD

Aufgabenstellung

RRD — Round-Robin-Database

- Aggregation & Visualisierung von Zeitreihen
- Cacti, Ganglia, Icinga, Munin, Nagios, . . .



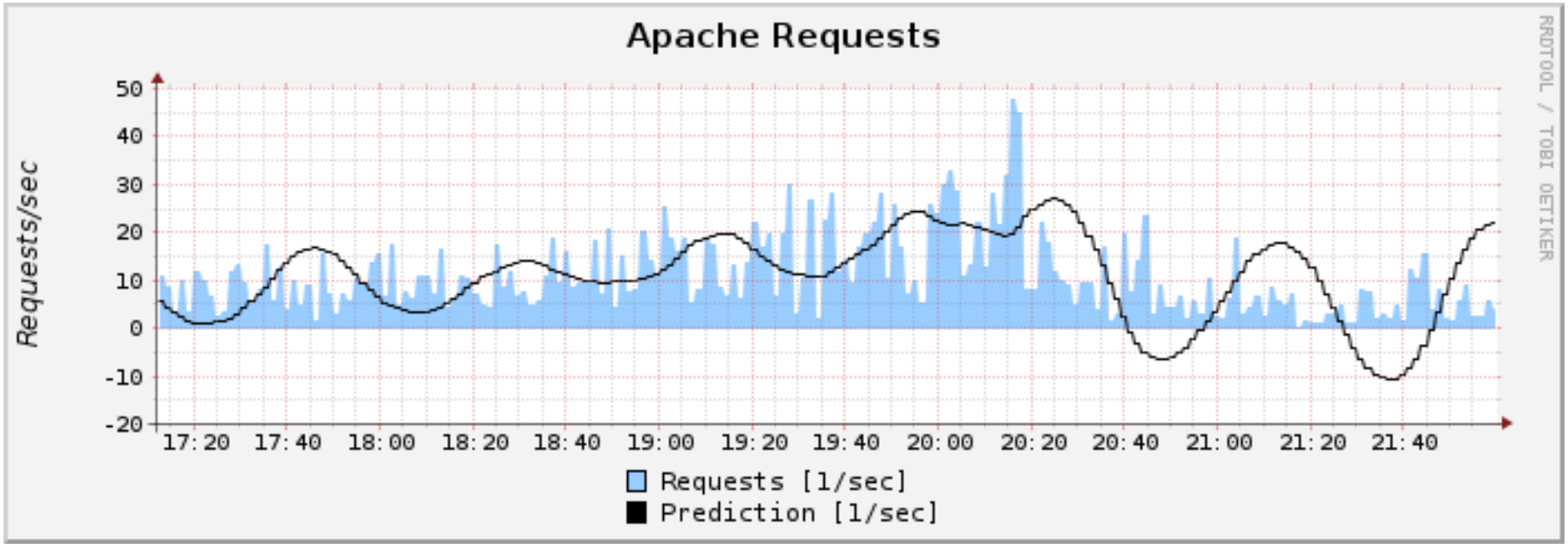
Prognosen mit RRD

- Möglichkeit zur Erkennung unüblicher Auslastung
- Holt-Winters-Algorithmus
  Parameter α (Niveau), β (Trend), γ (Saison)

Beispiel 2: Prognosen mit RRD

Aufgabenstellung

Parameter α, β und γ raten?



Beispiel 2: Prognosen mit RRD

Import der Daten

```
web <- read.csv(file = "data/apache_requests.csv.gz", header = TRUE)
```

```
head(web)
```

```
                 time requests_p_sec
1 2014-01-13 00:00:00           5.38
2 2014-01-13 00:00:10           8.30
3 2014-01-13 00:00:20           0.52
4 2014-01-13 00:00:30           5.90
5 2014-01-13 00:00:40           9.00
6 2014-01-13 00:00:50           0.44
```

Beispiel 2: Prognosen mit RRD

Bestimmung der Parameter

Periodische Zeitreihe mit 8640 Werten pro Periode

```
web.ts <- ts(web$requests_p_sec, frequency = 86400 / 10)
```

Parameter ermitteln

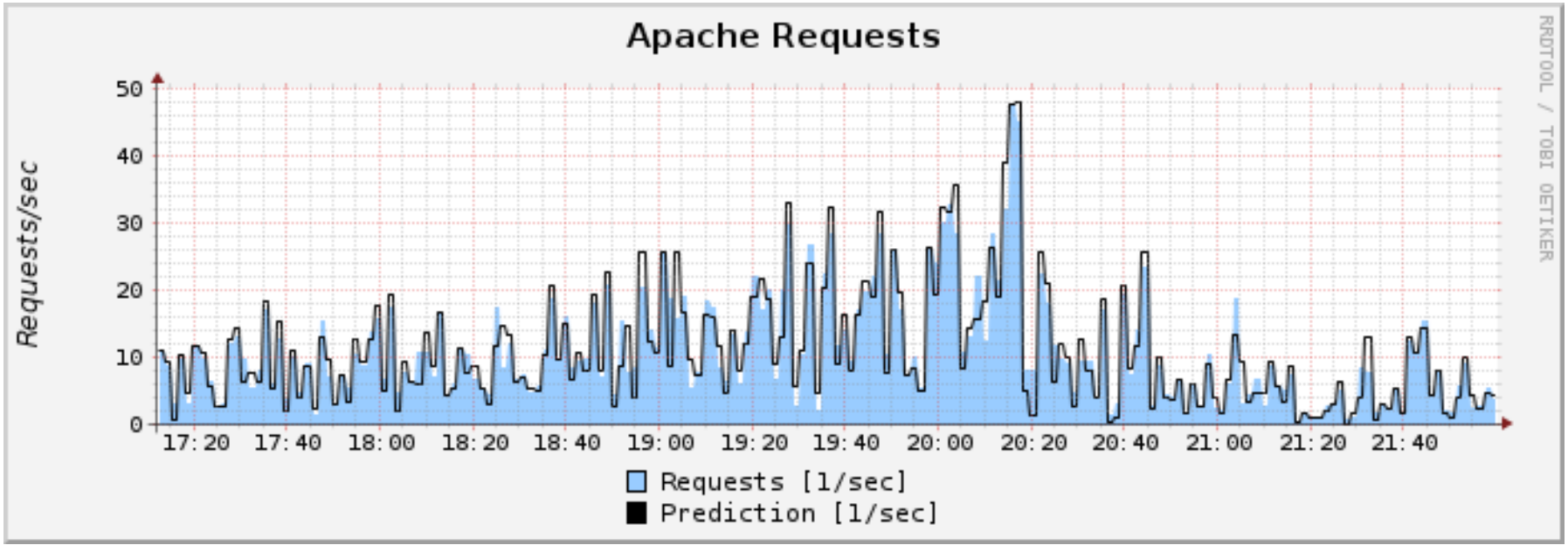
```
hw <- HoltWinters(web.ts)

c(hw$alpha, hw$beta, hw$gamma)

 alpha   beta  gamma
0.7291 0.0000 0.8835
```

Beispiel 2: Prognosen mit RRD

Anwendung der ermittelten Parameter

Bessere Prognose

- $\alpha = 0.729$
- $\beta = 0$
- $\gamma = 0.884$



Beispiel 3: Universal Scalability Law

Aufgabenstellung

Skalierbarkeit eines Tomcat Servers

- Maximaler Durchsatz
- Limitierung der Skalierbarkeit
- Prognose einer Optimierung

Beispiel 3: Universal Scalability Law

Einordnung

USL — Universal Scalability Law

- Modell zur Quantifizierung der Skalierbarkeit
  - Hardware: Anzahl Hosts, CPUs, Cores, . . .
  - Software: Anzahl Clients, Threads, . . .
- Entwickelt von Dr. Neil J. Gunther
  - *Guerrilla Capacity Planning*, Springer, 2007
  - `http://www.perfdynamics.com`

Beispiel 3: Universal Scalability Law

Messreihe als Ausgangspunkt

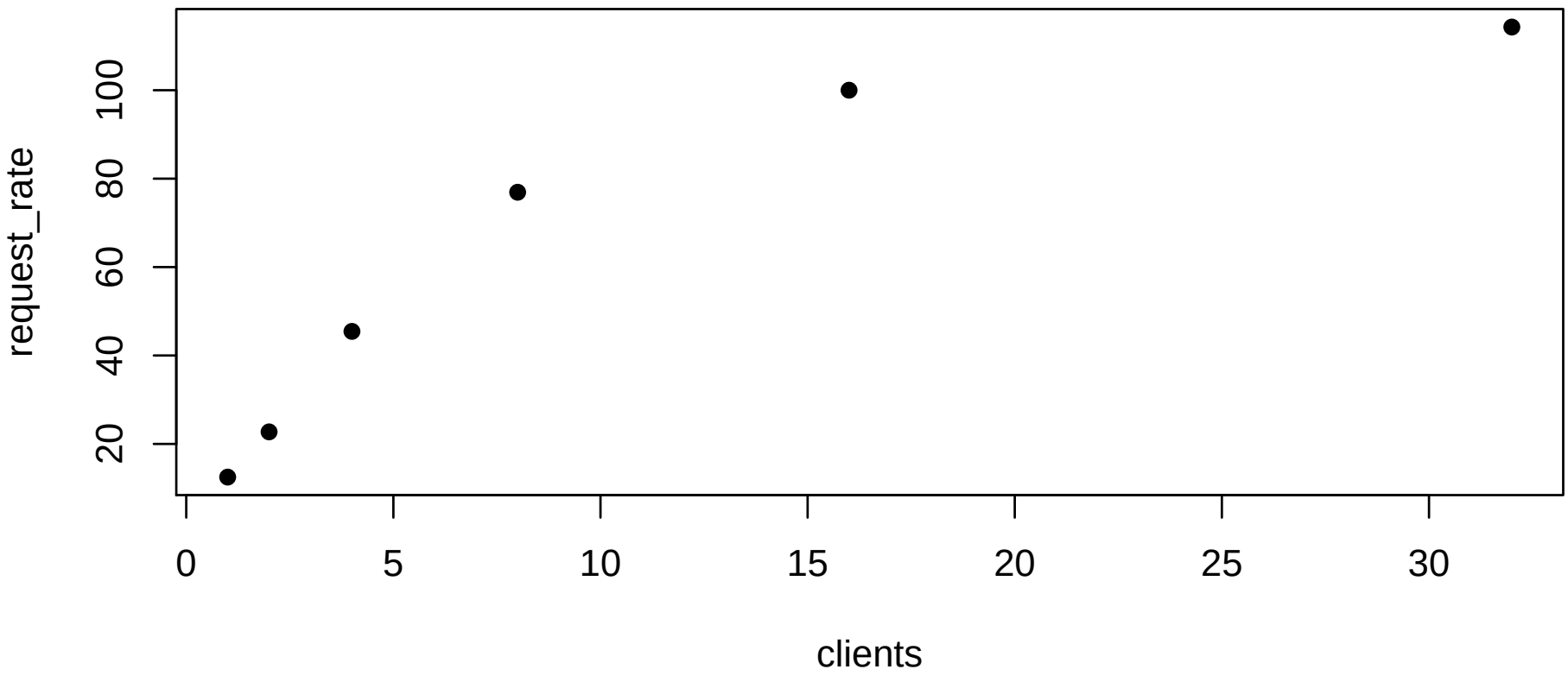
Messreihe für zunehmende Anzahl Clients

```
print(loadtest)

  clients request_rate service_time
1       1        12.50        43.96
2       2        22.73        49.69
3       4        45.45        54.06
4       8        76.92        74.49
5      16       100.00       128.51
6      32       114.29       238.73
```

Beispiel 3: Universal Scalability Law

Scatterplot der Messreihe

```
plot(request_rate ~ clients, data = loadtest, pch = 16)
```



Beispiel 3: Universal Scalability Law

Erstellung des USL Modells

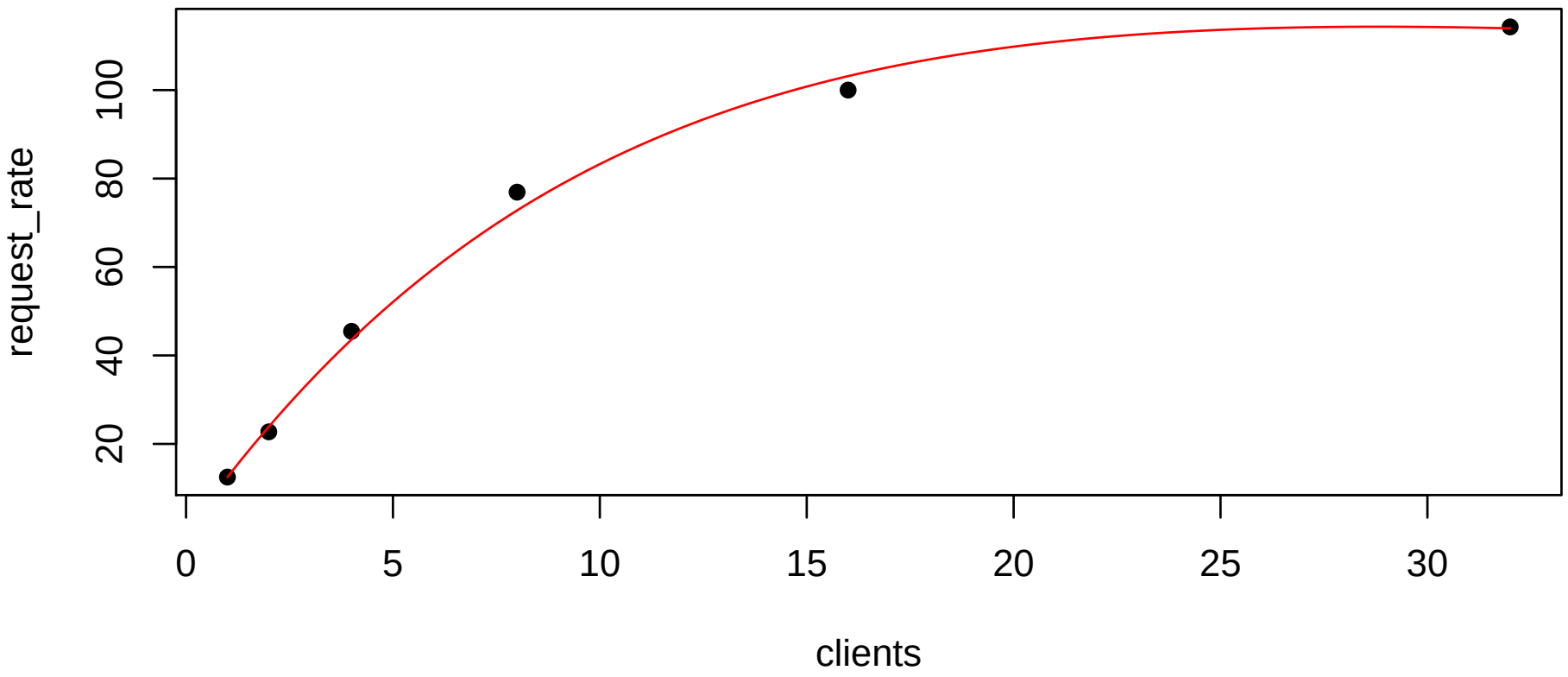
USL Modell erstellen

```
library(usl)

loadtest.usl <- usl(request_rate ~ clients, data = loadtest)
```

Beispiel 3: Universal Scalability Law

Darstellung des USL Modells

```
plot(request_rate ~ clients, data = loadtest, pch = 16)
plot(loadtest.usl, col = "red", add = TRUE)
```



Beispiel 3: Universal Scalability Law

Interpretation der Parameter

```
coef(loadtest.usl)

   sigma    kappa
0.044202 0.001149
```

Contention, *Serial Fraction*:
Overhead durch rein serielle Ausführung

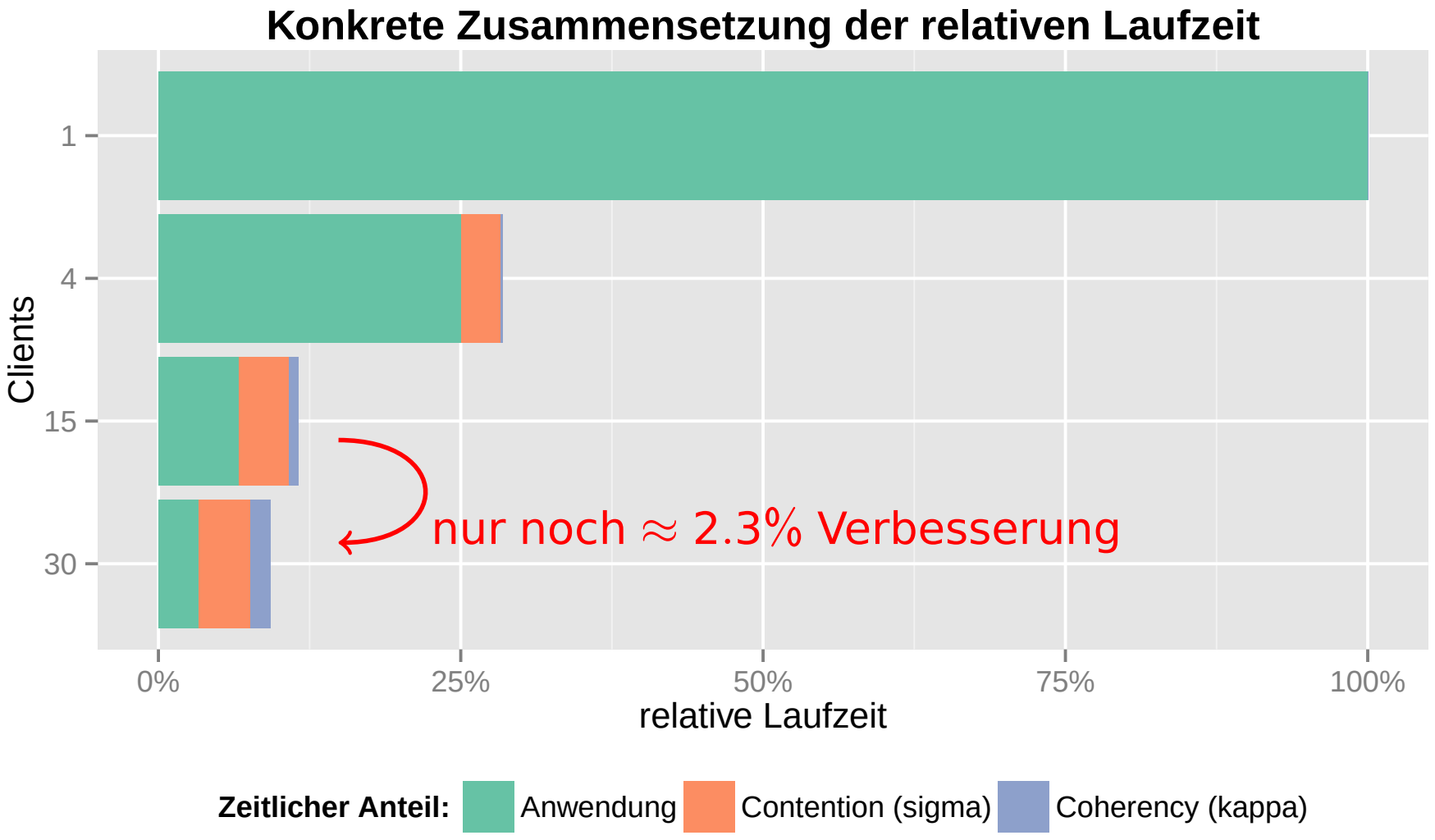
- Queues
- Locks
- ...

Coherency Delays:
Overhead zur Gewährleistung eines kohärenten und konsistenten Zustandes

- Punkt-zu-Punkt Synchronisierung
- Abgleich von Caches
- ...

Beispiel 3: Universal Scalability Law

Interpretation der Parameter



Beispiel 3: Universal Scalability Law

Auswertung des USL Modells

Prognose für Durchsatz

```
scf <- scalability(loadtest.usl)

# Predict throughput for 30, 36 and 40 clients
scf(c(30, 36, 40))

[1] 114.3 112.6 110.7
```

Maximaler Durchsatz

```
peak.scalability(loadtest.usl)

[1] 28.84

scf(peak.scalability(loadtest.usl))

[1] 114.3
```

Beispiel 3: Universal Scalability Law

Was-wäre-wenn-Analyse

Annahme: Optimierung reduziert Zeit in serieller Verarbeitung auf 2%

```
# Assume a smaller sigma
peak.scalability(loadtest.usl, sigma = 0.02)

[1] 29.2
```

```
scf <- scalability(loadtest.usl, sigma = 0.02)

scf(peak.scalability(loadtest.usl, sigma = 0.02))

[1] 145.4
```

Vielen Dank!

stm@kill-9.net

@UnixMagus

http://www.moeding.net

https://github.com/smoeding

Referenz


Jake D. Brutlag.
Aberrant behavior detection in time series for network monitoring.
In *Proceedings of the 14th USENIX Conference on System Administration*, LISA '00, pages 139–146, Berkeley, CA, USA, 2000. USENIX Association.


Neil J. Gunther.
Guerrilla Capacity Planning: A Tactical Approach to Planning for Highly Scalable Applications and Services.
Springer, Heidelberg, Germany, 1st edition, 2007.


Neil J. Gunther.
A general theory of computational scalability based on rational functions.
CoRR, abs/0808.1431, 2008.


Neil J. Gunther and Stefan Möding.
USL: Analyze system scalability with the Universal Scalability Law, 2013.
R package version 1.3.1.


Dan Leech.
Simple icons.
http://simpleicons.org/.

Anhang

Universal Scalability Law

Kapazität X (z.B. Durchsatz) bei p-facher Parallelisierung

$$\frac{X(p)}{X(1)} = \frac{p}{1 + \sigma\,(p-1) + \kappa\,p(p-1)}$$

Contention, Serial Fraction — *Coherency Delays*

Ideale Laufzeit bei p-facher Parallelisierung	$1/p$
Overhead durch Contention (Queues, Locks, . . .)	$\frac{p-1}{p} \times \sigma$
Overhead durch Coherency (P2P-Sync, Caches, . . .)	$\frac{p-1}{2} \times \kappa$