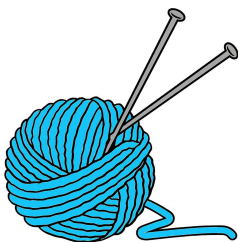


# Dynamische Dokumente mit R und $\text{\LaTeX}$

## Eine Einführung in knitr

Ulf Mertens

12. April 2014



- 1 Was ist knitr?
- 2 Chunk-Optionen
- 3 Tabellen
- 4 Graphiken
- 5 Default-Settings manipulieren
- 6 Code Externalization

---

Autor	Yihui Xie ( <a href="http://yihui.name/knitr/">http://yihui.name/knitr/</a> )
Zweck	Einbinden von R-Code in $\text{\LaTeX}$ -Dokumente
Geschichte	Weiterentwicklung des Pakets Sweave von Friedrich Leisch
Vorteile	kein copy-paste; 'research gets reproducible'

---

*Obviously the package name knitr was coined with weave in mind, and it also aims to be neater*

- 1 Erstellen eines Rnw-Dokumentes (angelehnt an noweb: Dokumentation und Quellcode in einem Dokument)
- 2 Schreiben eines klassischen LaTeX-Dokuments mit eingebettetem R-Code
- 3 R starten, und mithilfe der Funktion knit() die fertige .tex-Datei generieren
- 4 pdf-Dokument mit Editor der Wahl (Emacs, RStudio, TeXworks, LyX etc.) erstellen.

Manche Editoren haben knitr bereits implementiert. Falls nicht kann zur Erleichterung der Arbeit ein kleines Skript geschrieben werden.

<http://yihui.name/knitr/demo/editors/>

# Inline-Code und Chunks

## Einfaches Beispiel

```
\documentclass{article}  
\begin{document}
```

Der Wert von  $\pi$  ist  $\pi$ .

```
<<label>>=
```

```
paste(letters[c(11,14,9,20,18)],collapse="")
```

```
@
```

```
\end{document}
```

---

Der Wert von  $\pi$  ist 3.1416.

```
paste(letters[c(11, 14, 9, 20, 18)], collapse = "")
```

```
## [1] "knitr"
```

# Chunk-Optionen

echo

echo kontrolliert, ob der Code dargestellt wird.

```
\documentclass{article}
\begin{document}
<<label,echo=FALSE>>=
paste(letters[c(11,14,9,20,18)],collapse="")
@

\end{document}
```

---

```
## [1] "knitr"
```

results kontrolliert, ob der Output angezeigt wird.

```
\documentclass{article}
\begin{document}
<<label,results='hide'>>=
paste(letters[c(11,14,9,20,18)],collapse="")
@
\end{document}
```

---

```
paste(letters[c(11, 14, 9, 20, 18)], collapse = "")
```

# Chunk-Optionen

## Warning

Mithilfe von `warning` kann kontrolliert werden, ob Warnmeldungen angezeigt werden.

```
\documentclass{article}
\begin{document}
<<label,echo=FALSE>>=
ifelse(require(mosaic),"loaded","not available")
@

\end{document}
```

---

```
## Loading required package: mosaic
## Warning: there is no package called 'mosaic'

## [1] "not available"
```



# Chunk-Optionen

## warning

Mithilfe von `warning` kann kontrolliert werden, ob Warnmeldungen angezeigt werden.

```
\documentclass{article}
\begin{document}
<<label,echo=FALSE,warning=FALSE>>=
ifelse(require(mosaic),"loaded","not available")
@

\end{document}
```

---

```
## Loading required package: mosaic
```

```
## [1] "not available"
```

# Chunk-Optionen

tidy

Mithilfe von tidy kann entschieden werden, wie der Code im fertigen Dokument angezeigt werden soll (hierfür wird formatR-package verwendet).

```
\documentclass{article}
\begin{document}
<<label, tidy=TRUE, eval=FALSE>>=
if(require(mosaic){print("loaded")} else{print("not available")})
@

\end{document}
```

---

```
if (require(mosaic)) {
  print("loaded")
} else {
  print("not available")
}
```

# Chunk-Optionen

eval

Mithilfe von `eval` kann entschieden werden, ob der Code ausgeführt wird, bzw. welche Zeilen ausgeführt werden.

```
\documentclass{article}
\begin{document}
<<label,eval=FALSE>>=
ifelse(require(MASS),"loaded","not available")
@

\end{document}
```

---

```
ifelse(require(MASS), "loaded", "not available")
```

# Chunk-Optionen

eval

Mithilfe von `eval` kann entschieden werden, ob der Code ausgeführt wird, bzw. welche Zeilen ausgeführt werden.

*Hinweis:* Bei `tidy=TRUE` bezieht sich `eval` auf den Ausdruck, nicht die Zeile.

```
\documentclass{article}
\begin{document}
<<label ,eval=-2 ,echo=FALSE>>=
paste("Zeile1")
paste("Zeile2")
paste("Zeile3")
@

\end{document}
```

---

```
## [1] "Zeile1"
## [1] "Zeile3"
```

# Chunk-Optionen

## cache

In größeren Dokumenten mit vielen Chunks kann es sehr zeitintensiv sein, alle Chunks bei jedem Vorgang erneut auszuführen. In `knitr` können mit der Option `cache` die Ergebnisse (Plot, Tabelle etc.) zwischengespeichert werden. Es wird im weiteren Verlauf automatisch erkannt, ob sich an den Daten etwas verändert hat, ob also der Code ausgewertet werden muss oder nicht. Dabei werden auch Abhängigkeiten der Chunks untereinander berücksichtigt.

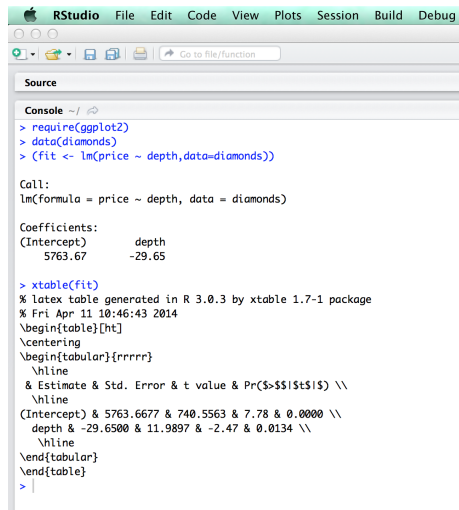
```
\documentclass{article}
\begin{document}
<<label , eval=-2, echo=FALSE , cache=TRUE >>=
#Rechnung
@

\end{document}
```

# Tabellen

mit xtable

Das Package xtable erstellt  $\text{\LaTeX}$ -Tabellen aus R-Objekten. Die Tabellen können dann direkt mit knitr in das Dokument eingebunden werden.



```
RStudio File Edit Code View Plots Session Build Debug
Source
Console ~/
> require(ggplot2)
> data(diamonds)
> (fit <- lm(price ~ depth,data=diamonds))

Call:
lm(formula = price ~ depth, data = diamonds)

Coefficients:
(Intercept)      depth
  5763.67         -29.65

> xtable(fit)
% latex table generated in R 3.0.3 by xtable 1.7-1 package
% Fri Apr 11 10:46:43 2014
\begin{table}[ht]
\centering
\begin{tabular}{rrrrr}
\hline
& Estimate & Std. Error & t value & Pr(>|t|>|t|) \\
\hline
(Intercept) & 5763.6677 & 740.5563 & 7.78 & 0.0000 \\
depth & -29.6500 & 11.9897 & -2.47 & 0.0134 \\
\hline
\end{tabular}
\end{table}
> |
```

# Tabellen

mit xtable

Die Option `results='asis'` printet den raw Code in das Dokument.

```
\documentclass{article}
\begin{document}
<<table1,echo=FALSE,results='asis'>>=
fit <- lm(price ~ depth,data=diamonds)
xtable(fit)
@

\end{document}
```

---

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	5763.6677	740.5563	7.78	0.0000
depth	-29.6500	11.9897	-2.47	0.0134

---

Die Option `sanitize.text.function=function(x){x}` erlaubt das Schreiben von TeX-Code in R.

```
\documentclass{article}
\begin{document}
<<table2,echo=FALSE,results='asis'>>=
vals <- data.frame(matrix(rnorm(4),ncol=2))
colnames(vals) <- c("\$\\mu\$", "\$\\sigma\$")
rownames(vals) <- c("\$Var_{1}\$", "\$Var_{2}\$")
print(xtable(vals),sanitize.text.function=function(x){x})
@

\end{document}
```

---

	$\mu$	$\sigma$
$Var_1$	0.70	-0.37
$Var_2$	-0.43	0.93



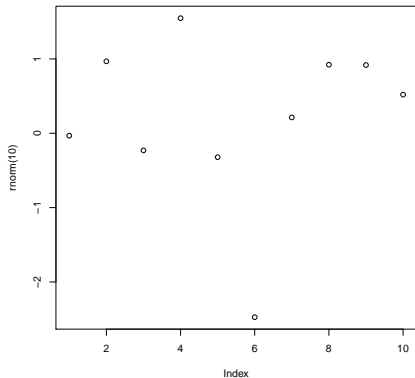
knitr erleichtert das Einbinden von Plots durch zahlreiche hilfreiche Optionen.

- `out.width`: Größe des Plots im Dokument
- `fig.height`: Größe des Plots bzgl. des Device
- `dev`: spezifiziert den graphical device (pdf (default),png,jpeg etc.)
- `fig.show`: Plots erst am Ende zeigen?
- `fig.keep`: Sollen nur bestimmte Plots gezeigt werden (zB. der Letzte)?
- `fig.cap`: bindet Plot in figure-Umgebung ein und vergibt Titel

# Graphiken

```
out.width='0.45\\textwidth'
```

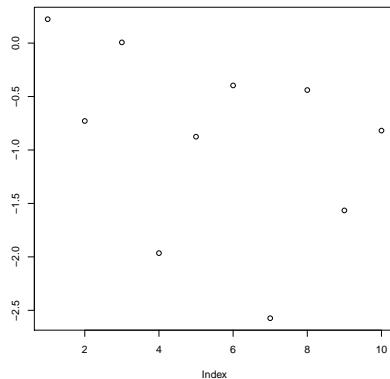
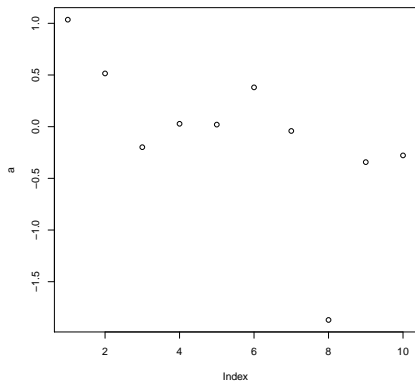
```
set.seed(3424)  
plot(rnorm(10))
```



# Graphiken

fig.show='hold'

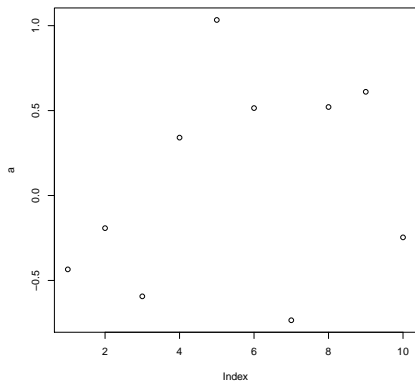
```
a <- rnorm(10)
plot(a)
b <- rnorm(10)
plot(b)
```



# Graphiken

fig.show='asis'

```
a <- rnorm(10)  
plot(a)
```

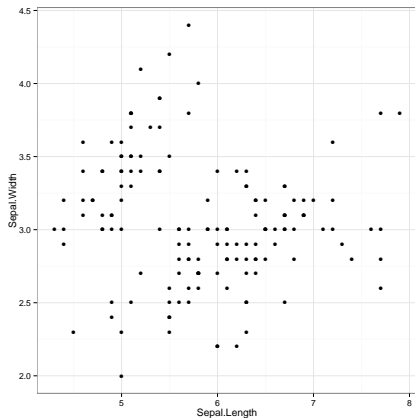


```
b <- rnorm(10)  
plot(b)
```

# Graphiken

fig.keep='last'

```
pl <- ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width))  
pl + geom_point()  
pl + theme_bw()
```



# Graphiken

fig.cap='Titel'

```
pl <- ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width))  
pl + geom_point() + theme_bw()
```

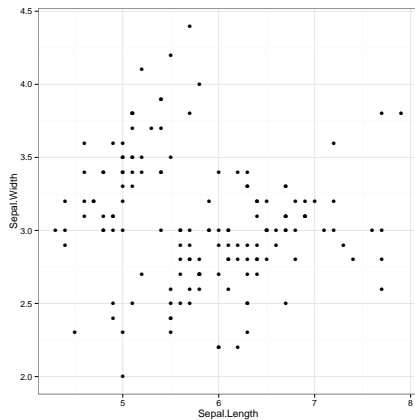
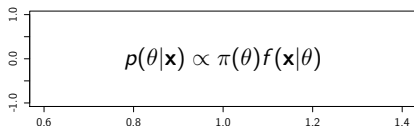


Abbildung : Titel

der tikz-Device hat im wesentlichen zwei Vorteile:

- 1  $\text{\LaTeX}$ -Code kann direkt in R geschrieben werden (expression wird überflüssig)
- 2 font styles der Plots stimmen mit denen des restlichen Dokuments überein

```
require(tikzDevice)
plot(0, type = "n", ann = FALSE)
text(0, paste("$p(\theta|\mathbf{x})$", "\\propto", "\\pi(\theta)f(\mathbf{x}|\theta)"),
      cex = 2)
```



# Graphiken

mehrere figure-Umgebungen mit `subfig`

Gelegentlich möchte man eine figure-Umgebung mit zwei Sub-Plots erzeugen. Hierzu dient die Option `fig.subcap`. Das  $\text{\LaTeX}$ -Paket `subfig` muss ebenfalls geladen werden.

Folgende Chunk-Optionen wurden für die beiden Plots verwendet:

- `out.width='0.49\\linewidth'`
- `fig.show='asis'`
- `fig.subcap=c('Untertitel 1','Untertitel 2')`
- `fig.cap='Haupttitel'`
- `fig.height=3`

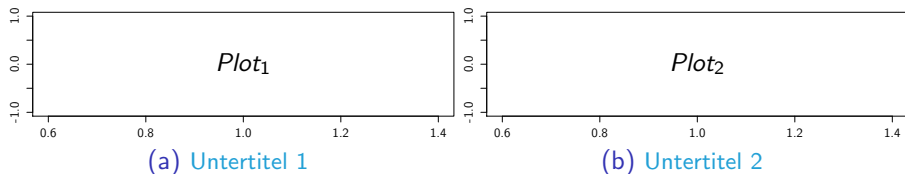


Abbildung : Haupttitel



# Setup

## Default-Optionen festlegen

Einige Chunk-Optionen sollen für alle Chunks gelten und man möchte sie nicht jedes Mal wieder angeben. Hierfür gibt es die Funktion `opts_chunk$set`. Der 'Setup-Chunk' sollte der erste Chunk im Dokument sein.

```
\documentclass{article}
\begin{document}
<<setup,include=FALSE>>=
opts_chunk$set(echo=FALSE,warning=FALSE,out.width='0.45\\textwidth
',fig.height=3)
@

<<plot8,fig.cap="Plot">>=
plot(0, type="n",ann=FALSE)
text(0,paste("Ein Plot"),cex=2)
@

\end{document}
```

# Setup

## Default-Optionen festlegen

Standardmäßig werden vier Nachkommastellen angezeigt, i.d.R. genügen auch zwei Stellen. In R kann dies über die Funktion `round` erreicht werden. Damit nicht in jedem Inline Code-Chunk `round(...)` stehen muss, gibt es den `options` command in `knitr`. Über `knit_theme$set` können verschiedene Themes ausgewählt werden.

```
\documentclass{article}
\begin{document}

<<setup, include=FALSE>>=
options(digits=2)
knit_theme$set("blacknblue")
@
```

Der Wert von  $\pi$  ist  $\pi$ .

```
<<lab, prompt=TRUE>>=
summary(diamonds$price)
@
```

```
\end{document}
```

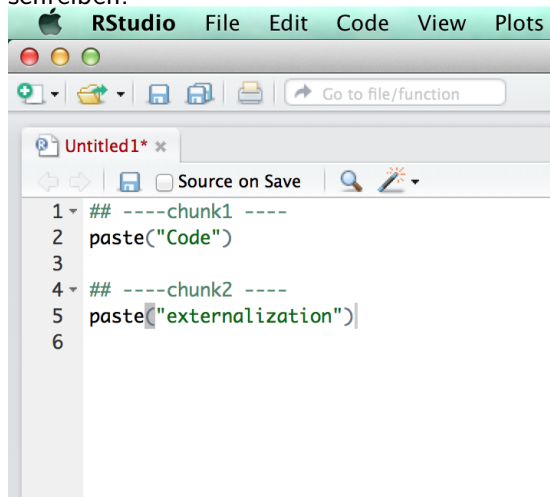
Der Wert von  $\pi$  ist 3.14.

```
> summary(diamonds$price)
```

##	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
##	326	950	2400	3930	5320	18800

# Code Externalization

Bei vielen Zeilen Code ist es übersichtlicher, den Code in ein R-Skript zu schreiben.



The screenshot shows the RStudio application window. The title bar includes the Apple logo, the text 'RStudio', and menu items 'File', 'Edit', 'Code', 'View', and 'Plots'. Below the title bar are window control buttons (red, yellow, green) and a toolbar with icons for file operations and a search bar labeled 'Go to file/function'. The main editor area shows a file named 'Untitled1\*' with a toolbar containing navigation arrows, a save icon, a checkbox labeled 'Source on Save', and search and edit icons. The code in the editor is as follows:

```
1 ## ----chunk1 ----
2 paste("Code")
3
4 ## ----chunk2 ----
5 paste("externalization")
6
```

# Code Externalization

```
\documentclass{article}
\begin{document}
<<setup2,include=FALSE>>=
read_chunk("file.R")
@

<<chunk1,echo=FALSE>>=
@

<<chunk2,echo=FALSE>>=
@

\end{document}
```

---

```
## [1] "Code"
```

```
## [1] "externalization"
```

Vielen Dank für die Aufmerksamkeit