

# EINFÜHRUNG IN R - HOUSEKEEPING

Jan-Philipp Kolb

13 Juni, 2019

# PAKETE AUTOMATISCH INSTALLIEREN

- Mit `installed.packages()` kann ich herausfinden, welche Pakete installiert sind.
- Bei mir sind es momentan 333 Pakete.

```
my_packages <- installed.packages()
mypack <- my_packages[, "Package"]
```

```
my_packages[, "Package"]
```

##	abind	acepack	AER
##	"abind"	"acepack"	"AER"
##	AmesHousing	antiword	aplpack
##	"AmesHousing"	"antiword"	"aplpack"
##	arm	askpass	assertthat
##	"arm"	"askpass"	"assertthat"
##	backports	base	base64enc
##	"backports"	"base"	"base64enc"
##	bayestestR	beanplot	BH
##	"bayestestR"	"beanplot"	"BH"
##	bindr	bindrcpp	bitops
##	"bindr"	"bindrcpp"	"bitops"

# FEHLENDE PAKETE INSTALLIEREN

- Zunächst wird eine Liste (bspw. `packlist`) mit den benötigten Paketen erstellt.
- Dann wird geprüft ob diese Pakete unter den installierten Paketen sind.
- Zuletzt werden Pakete installiert, die noch nicht vorhanden sind.

```
packlist <- c("ggplot2", "Rcpp")  
new.packages <- packlist[!(packlist %in% mypack)]  
if(length(new.packages)) install.packages(new.packages)
```



RStudio Support

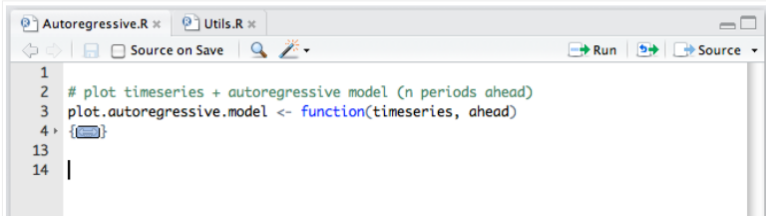
April 10, 2019 09:51

[Follow](#)

## Code Folding and Sections

### Code Folding

RStudio supports both automatic and user-defined folding for regions of code. Code folding allows you to easily show and hide blocks of code to make it easier to navigate your source file and focus on the coding task at hand. For example, in the following source file the body of the `plot.autoregressive.model` has been folded:



```
Autoregressive.R x  Utils.R x
Source on Save  Run  Source v
1
2 # plot timeseries + autoregressive model (n periods ahead)
3 plot.autoregressive.model <- function(timeseries, ahead)
4 {
13
14 |
```

- Wenn das Argument `echo=T` gesetzt ist, sieht man auch was innerhalb des Skripts passiert.
- Die Funktion eignet sich, bspw. wenn immer wieder die gleichen Arbeitsschritte durchgeführt werden, die selten verändert werden müssen.

```
source("../rcode/load_packages.R",echo=T)
```

# SNIPPETS - FÜR EINEN AUTOMATISCHEN HEADER

Tools > Global Options

## Snippets

☒ Enable code snippets

Edit Snippets...



```
16
17 head|
```

	header_script	{snippet}
	◆ head	{utils}
17:5	◆ head.matrix	{utils}

```
#####
## Project:
## Script purpose:
## Date:
## Author:
#####
```

Console Terminal x Jobs x

D:/github/ffm\_rintro/ ↻

[OffeneDaten.FRANKFURT.de](#)



[Start](#) [Datensätze](#) [Organisationen](#) [Gruppen](#) [Was sind Offene Daten?](#)

[/ Organisationen / Bürgeramt, Statistik und ... / Frankfurter ...](#)

## Frankfurter Stadtteilgrenzen für GIS-Systeme

0 Follower

[Datensatz](#) [Gruppen](#) [Aktivitätsanzeige](#)

## Frankfurter Stadtteilgrenzen für GIS-Systeme

Das ZIP-Paket enthält alle notwendigen Dateien zur Darstellung der Frankfurter Stadtteilgrenzen in GIS-Systemen. Die Grenzen liegen im ESRI® Shape Format vor und sind universell einsetzbar. Das zu Grunde liegende Koordinatensystem ist das europäische ETRS89/UTM.

# SHAPEFILES IMPORTIEREN

```
ffm_shp <- rgdal::readOGR(  
  "../data/Stadtteile_Frankfurt_am_Main.shp")
```

```
## OGR data source with driver: ESRI Shapefile  
## Source: "D:\github\ffm_rintro\data\Stadtteile_Frankfurt_am_Ma  
## with 46 features  
## It has 2 fields
```



```
head(ffm_shp@data)
```

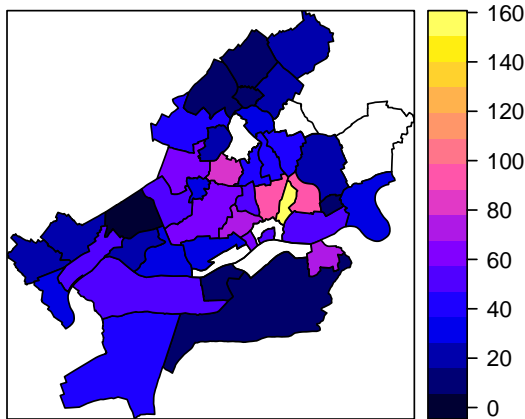
##	STTLNR	STTLNAME
## 0	1	Altstadt
## 1	2	Innenstadt
## 2	3	Bahnhofsviertel
## 3	4	Westend-Süd
## 4	5	Westend-Nord
## 5	6	Nordend-West

- Voraussetzung: in beiden Datensätzen befinden sich die Stadtteile in der gleichen Ordnung.

```
dat <- read.csv2("../data/bauenwohnen.csv")  
ffm_shp@data$Einwohnerdichte <-  
dat$Wohnumfeld...öffentlicher.Raum.Einwohnerdichte.je.ha.2012
```

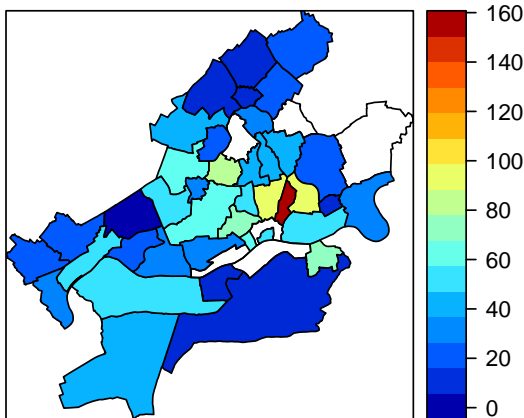
# EINE THEMATISCHE KARTE PLOTTEN

```
sp::spplot(ffm_shp, "Einwohnerdichte")
```



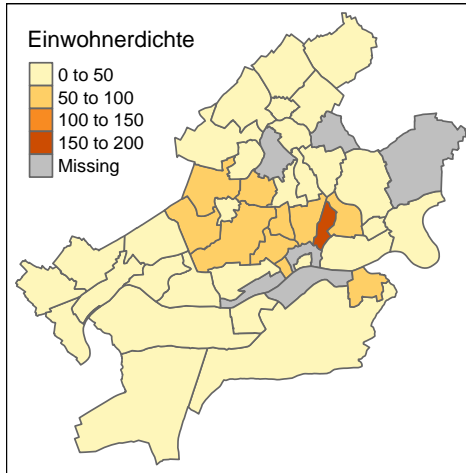
# EINE ANDERE FARBPALETTE

```
library(colorRamps)
sp::spplot(ffm_shp, "Einwohnerdichte", col.regions=matlab.like(100))
```



# THEMATISCHE KARTE MIT TMAP

```
tmap::qtm(ffm_shp, "Einwohnerdichte")
```



- Indikatoren matchen und hoffen, dass keine NA's auftauchen.

```
ind <- match(ffm_shp$STTLNAME, dat$Stadtteil)
ind
```

```
## [1]  1  2  3 NA  5  6  7  8  9 10 11 12 NA NA NA 15 16 17 18
## [24] 23 24 25 26 27 28 29 30 31 32 33 NA 35 36 37 38 39 40 41
```

# WO HAKT ES?

```
dat_ffm <- data.frame(ffm_shp$STTLNAME, dat$Stadtteil)
dat_ffm[is.na(ind),]
```

##	ffmpeg.STTLNAME	dat.Stadtteil
## 4	Westend-Süd	Westend-Süd
## 13	Sachsenhausen-N.	Sachsenhausen-Nord
## 14	Sachsenhausen-S.	Sachsenhausen-Süd
## 15	Flughafen	Oberrad
## 20	Riedelheim	Hausen
## 35	Häschst	Nied

```
ffm_shp$STTLNAME[which(is.na(ind))]
```

```
## [1] Westend-Süd Sachsenhausen-N. Sachsenhausen-S. Flughafen  
## [5] Riedelheim Hirschst  
## 46 Levels: Altstadt Bahnhofsviertel Bergen-Enkheim ... Zeilshausen
```

```
nas1 <- ffm_shp$STTLNAME[which(is.na(ind))][1]  
agrep(nas1, dat$Stadtteil)
```

```
## [1] 4
```

```
ind[which(is.na(ind))[1]] <- agrep(nas1, dat$Stadtteil)
```



```
data_path <- "D:/gitlab/IntroDataAnalysis/data/"
gpdat <- foreign::read.spss(paste0(data_path,
                                   "ZA5666_v1-0-0.sav"), to.data.frame=TRUE)
att_dat <- attributes(gpdat)
names(att_dat)
```

## "names"

"class"

"row.names"

"variab

```
att_dat$variable.labels
```

```
"Großstadtnähe Wohngegend"
bczd002a
"Beeinträchtigung Umwelteinflüsse: Lärmbelästigung"
bczd003a
"Beeinträchtigung Umwelteinflüsse: Luftverschmutzung"
bczd004a
"Beeinträchtigung Umwelteinflüsse: Fehlende Grünflächen"
bczd005a
"NEP-Skala: Nähern uns Höchstzahl an Menschen"
bczd006a
"NEP-Skala: Recht Umwelt an Bedürfnisse anzupassen"
bczd007a
"NEP-Skala: Folgen von menschlichem Eingriff"
bczd008a
```

- Für die Nutzung der API ist vermutlich das `httr` Paket hilfreich

```
install.packages("httr")
```

```
devtools::install_github("cutterkom/destatiscleanr")
```

```
library(destatiscleanr)
```

```
$codepage  
[1] 65001
```

```
> names(att_dat)
```

```
[1] "names"
```

```
> destatiscleanr::
```

◆ clean_header	{destatiscleanr}
◆ convert_columns_to_numeric	{destatiscleanr}
◆ delete_copyright	{destatiscleanr}
◆ destatiscleanr	{destatiscleanr}
◆ read_file	{destatiscleanr}

```
read_file(file)
```

This functions reads the csv file by using German decimal marks

Press F1 for additional help

```
e.labels" "codepage"
```