

recexcavAAR and far beyond...

Anregungen rund um R, Data Science und Archäologie

Clemens Schmid

17. Mai 2017

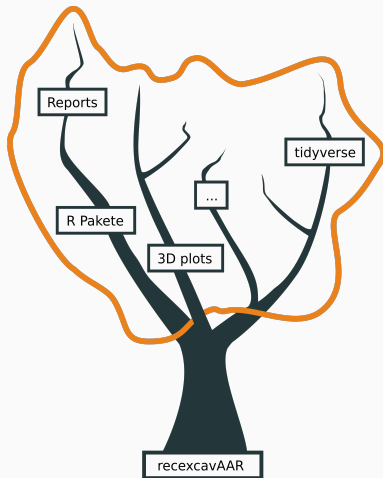



Figure 1: Inhaltsbaum

- R Pakete 
- 3D plots in R
- Reports
- recexcavAAR
- tidyverse

*A step to help you slow down -
put your fork/spoon down
between bites and take a sip
of water before picking them
up for the next bite.*
– reddit user **angelforhirex** in
Thread "I want to learn how
to eat smaller portions"

R Pakete

R Development Core Team – Writing R Extensions

<https://cran.r-project.org/doc/manuals/r-release/R-exts.pdf>

Hadley Wickham – R packages

<http://r-pkgs.had.co.nz>

*Packages are the fundamental units of reproducible R code. They include reusable **R functions**, the **documentation** that describes how to use them, and **sample data**.*

– Hadley Wickham

Windows ohne Rtools

1. RStudio: Ein neues Projekt `~/expack` anlegen

2. Windows Binaries für Paket `expack` herunterladen

https://github.com/nevrome/tutorial_cologne_may2017/tree/master/expack

3. RStudio: `expack` installieren
(Packages > Install > Package Archive File)

4. Paket laden

```
library(expack)
```

Linux oder Windows mit Rtools

1. Paket anlegen

```
install.packages("devtools")  
devtools::create("~/expack")
```

2. RStudio: In das entstandene Projekt wechseln

3. Beispieldaten schaufeln

https://github.com/nevrome/tutorial_cologne_may2017/tree/master/data
📁 `~/expack/data`

4. Paket bauen, installieren und laden:

`ctrl` + `↑` + `B`

```
#devtools::build()  
#devtools::install()  
library(expack)
```

5. Fleischbeschau!

```
help(package = "expack")
```

6. Beispieldaten ansehen

```
trench
```

7. On a completely different note... Pakete installieren

```
install.packages(c("rgl", "recexcavAAR", "magrittr", "tidyverse"))
```

3D plots in R

3D plots in R

JavaScript APIs: **plotly** & **rthreejs**

<https://plot.ly/r/#3d-charts>

<https://github.com/bwlewis/rthreejs>

OpenGL API: **rgl**

<https://cran.r-project.org/web/packages/rgl/vignettes/rgl.html>

Für echte 3D plots gibt es in R im Augenblick zwei Optionen:

APIs zu JavaScript Rendering Bibliotheken und GPU APIs.

Nur **rgl** besitzt einen **großen Funktionsumfang**, ist wegen der direkten GPU Nutzung über OpenGL lokal **performant** und kann gleichermaßen automatisch embedded JavaScript für **Web-Anwendungen** generieren.

JavaScript Visualisierung mit R

<http://www.htmlwidgets.org>

3D plots in R

1. neues R-Skript `~/expack/R/dreiD.R` anlegen
2. coole Funktion schreiben

```
coolplot <- function(points_df, type = "p", ...){  
  rgl::plot3d(  
    points_df[, 1], points_df[, 2], points_df[, 3],  
    type = type,  
    xlab = "x", ylab = "y", zlab = "z",  
    ...  
  )  
}
```

3. Namen der Funktion coolplot exportieren, indem man `~/expack/NAMESPACE` editiert

```
export(coolplot)
```

4. Paket **rgl** als Voraussetzung des Pakets definieren, indem man */expack/DESCRIPTION* editiert

```
...  
Description: What the package does (one paragraph).  
Imports:  
  rgl (>= 0.98.1)  
Depends: R (>= 3.3.3)  
...
```

5. Paket neu bauen: `ctrl` + `↑` + `B`

6. Ausprobieren!

```
coolplot(trench, type = "l")  
coolplot(sherds, type = "p", add = TRUE)
```

7. noch eine coole Funktion schreiben

```
coolpicture <- function(image, position) {  
  rgl::show2d( { graphics::par(mar = rep(0, 4))  
    graphics::plot( 0:1, 0:1, type = "n", ann = FALSE,  
                    axes = FALSE, xaxs = "i", yaxs = "i" )  
    graphics::rasterImage(image, 0, 0, 1, 1)  
  },  
  x = position[, 1], y = position[, 2], z = position[, 3]  
)  
}
```

8. Namen der neuen Funktion exportieren und Paket neu bauen (`(ctrl) + (↑) + (B)`)

```
export(coolpicture)
```

9. Ausprobieren!

```
coolplot(trench, type = "l")  
coolpicture(Nprofile, Nprofile_corners)
```

Reports

R Markdown

<http://rmarkdown.rstudio.com/>

Markdown Cheatsheet

<http://commonmark.org/help/>

R Markdown ist ein Framework zur Erstellung von **daten- und analyseorientierten Texten, Websites und Präsentationen**

- Sehr einfache Auszeichnungssprache **Markdown**
- Verarbeitung von **R, Python und SQL-Code**
- Integration von **Textverarbeitungswerkzeugen** (z.B. Referenzmanagement)
- Output über vorimplementierte Parsingworkflows:
PDF, HTML, Word-Dokumente, etc...

1. RStudio: Ein neues R Markdown Dokument anlegen
(File > New File > R Markdown > Document > HTML)

1. Eine neue Vignette anlegen und das Paket dafür vorbereiten

```
devtools::use_vignette("recex")
```

2. Dokument testweise rendern: `ctrl` + `↑` + `K`

```
rmarkdown::render("~/expack/.../recex.Rmd")
```

3. Inhalt abgesehen vom Header (zwischen ---) löschen

4. Überschrift und Text anlegen (△ Leerzeilen!)

```
# Space archaeology
```

```
The Kingdom of the Crystal Skull was a good movie.
```

```
> "RRRAARRWHHGWR."
```

```
> -- Chewbacca in *The Empire Strikes Back*
```

```
![] (https://goo.gl/X0F2ce)
```

5. R code chunks anlegen: ````{r} ... ````

6. Chunks mit Leben füllen: Bibliotheken laden

```
```{r}
```

```
library(expack)
```

```
library(rgl)
```

```
library(recexcavAAR)
```

```
```
```

7. 2D Plot (und anschließend rendern: `ctrl` + `↑` + `K`)

```
```{r}
```

```
plot(sherds$x, sherds$z)
```

```
```
```

8. 3D Plot (`ctrl` + `↑` + `K`)

```
```{r, echo=FALSE, results="hide"}  
avoid plotting in X11 window
open3d(useNULL = TRUE)

...

```{r}  
  
coolplot(trench, type = "l")  
coolpicture(Nprofile, Nprofile_corners)  
coolplot(sherds, type = "p", add = TRUE, col = "red", size = 10)  
  
...  
  
```{r, echo=FALSE}  

rglwidget()

...`
```



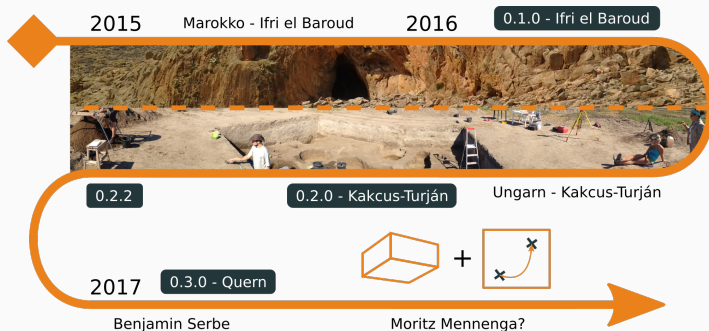
**recexcavAAR**

---

...

*Description: A toolset for 3D reconstruction and analysis of excavations. It provides methods to reconstruct natural and artificial surfaces based on field measurements. This allows to **spatially contextualize documented subunits and features**. Intended to be part of a 3D visualization workflow.*

...



**Figure 2:** Timeline recexcavAAR

recexcavAAR auf Github – Entwicklungsumgebung




<https://github.com/ISAACKiel/recexcavAAR>

recexcavAAR auf CRAN

<https://CRAN.R-project.org/package=recexcavAAR>

---

## Vignettes (Mai 2017)

-  Semiautomatic spit attribution
-  Trench visualisation
-  Transforming coordinates

## Meine Lieblingsfunktionen

- `fillhexa`: Fills hexahedrons with a regular point raster (3D)
- `posdeclist`: Multiple point position decision in relation to a set of stacked surfaces (3D)

## 1. Paket **recexcavAAR** als Voraussetzung des Pakets definieren

(...  +  + )

Imports:

```
rgl (>= 0.98.1),
```

Suggests:

```
recexcavAAR (>= 0.3.0)
```

## 2. Kriging und Positionsbestimmung im R Markdown Dokument hinzufügen

```
```{r}
```

```
surf <- recexcavAAR::kriglist(
  list(nivellement1, nivellement2), lags = 2
)
sherds_pos <- recexcavAAR::posdec(sherds[1:3], surf)
```

```
```
```

### 3. 3D Plot (... und + + )

```
```{r}

coolplot(trench, type = "l")
coolpicture(Nprofile, Nprofile_corners)
coolplot(
  sherds_pos, type = "p", add = TRUE,
  col = sherds_pos$pos + 1, size = 10
)
coolplot(surf[[1]], type = "p", add = TRUE)
coolplot(surf[[2]], type = "p", add = TRUE)

...

```{r, echo=FALSE}

rglwidget()

...

```

**tidyverse**

---

Hadley Wickham – R for Data Science

<http://r4ds.had.co.nz>

Winston Chang – R Graphics Cookbook

<http://www.cookbook-r.com/Graphs>

---

Das tidyverse ist eine **Sammlung von auf einander angepassten Paketen** für **Datenverarbeitung** und **Modellierung**, die zum **digitalen Standardwerkzeug** eines Archäologen gehören sollte.

---

<b>ggplot2:</b>	Daten visualisieren (plot 2.0)
<b>tibble:</b>	Daten verwalten (data.frame 2.0 ?)
<b>tidyr:</b>	Datenlayout bereinigen
<b>readr:</b>	Daten einlesen (read.table 2.0 ?)
<b>dplyr:</b>	Daten verarbeiten -> Perspektive data.frame
<b>purrr:</b>	Daten verarbeiten -> Perspektive list + vector

---

Speziellere Pakete, die der selben Philosophie folgen: forcats, stringr, readxl, ...

**magrittr** dient dabei dazu, Funktionen systematisch aneinander zu reihen.

1. neues R Skript `~/expack/playground/sherds.R` zum Spielen anlegen
2. Skript der `.Rbuildignore` hinzufügen, um es vom Buildprozess des Pakets auszuklammern

```
devtools::use_build_ignore("playground/sherds.R")
```

3. Bereits bekannten Code kopieren

```
library(expack)
library(recexcavAAR)
library(magrittr)
library(tidyverse)
```

```
surf <- recexcavAAR::kriglist(
 list(nivellement1, nivellement2), lags = 2
)
sherds_pos <- recexcavAAR::posdec(sherds[1:3], surf)
```

4. Let's play together: R.



```
sherds_main <- sherds %>%
 dplyr::mutate(pos = sherds_pos$pos) %>%
 dplyr::mutate(
 pos = replace(pos, pos == 0, "couche_rouge"),
 pos = replace(pos, pos == 1, "sec._loess"),
 pos = replace(pos, pos == 2, "plough_horizon")
)

stratigraphy <- c("plough_horizon", "sec._loess", "couche_rouge")
sherds_main$pos <- factor(sherds_main$pos, stratigraphy)

sherds_main %>% ggplot() +
 geom_bar(aes(x = pos))

sherds_main %>%
 dplyr::group_by(pos) %>%
 dplyr::summarise(count = n()) %>%
 dplyr::mutate(percentage = (count / sum(count) * 100) %>% round(0))
```

```
sherds_main %>% ggplot() +
 geom_bar(aes(x = decoration)) +
 facet_grid(~pos) +
 theme(axis.text.x = element_text(
 angle = 30, hjust = 1, vjust = 1)
)
```

```
sherds_main <- sherds_main %>%
 tidyr::separate(
 decoration,
 into = c("deco", "deco_comment"), sep = ";",
 remove = FALSE
)
```

```
sherds_main %>% ggplot() +
 geom_bar(aes(x = deco)) +
 facet_grid(~pos)
```

```
g <- sherds_main %>% ggplot() +
 geom_point(aes(x = size, y = w_thick)) +
 facet_grid(~pos)

models <- sherds_main %>%
 base::split(.$pos) %>%
 purrr::map(~lm(w_thick ~ size, data = .)) %>%
 purrr::map(~coef(.)) %>%
 as.data.frame %>% t %>% data.frame %>%
 dplyr::mutate(pos = factor(rownames(.), stratigraphy))

g + geom_abline(
 data = models,
 aes(intercept = X.Intercept., slope = size)
)
```

## **Abschließende Gedanken**

---

- ⚠ Das ist **kein** produktionsfertiges R-Paket. Wir haben einige essenzielle Aspekte ausgelassen: `ctrl` + `↑` + `E`
  - Dokumentation! (z.B. mit roxygen2)
  - Versionskontrolle (z.B. mit git)
  - korrekte Datenimplementierung
  - Tests
  - ...
- **Aber:** Wir haben einige coole Zusammenhänge berührt!
  - Paketentwicklung
  - 3D plots
  - R Markdown
  - recexcavAAR
  - **tidyverse**
- recexcavAAR, mortAAR, quantAAR... **ISAAK** ist offen für externe Mitarbeit!

<https://isaakiel.github.io>