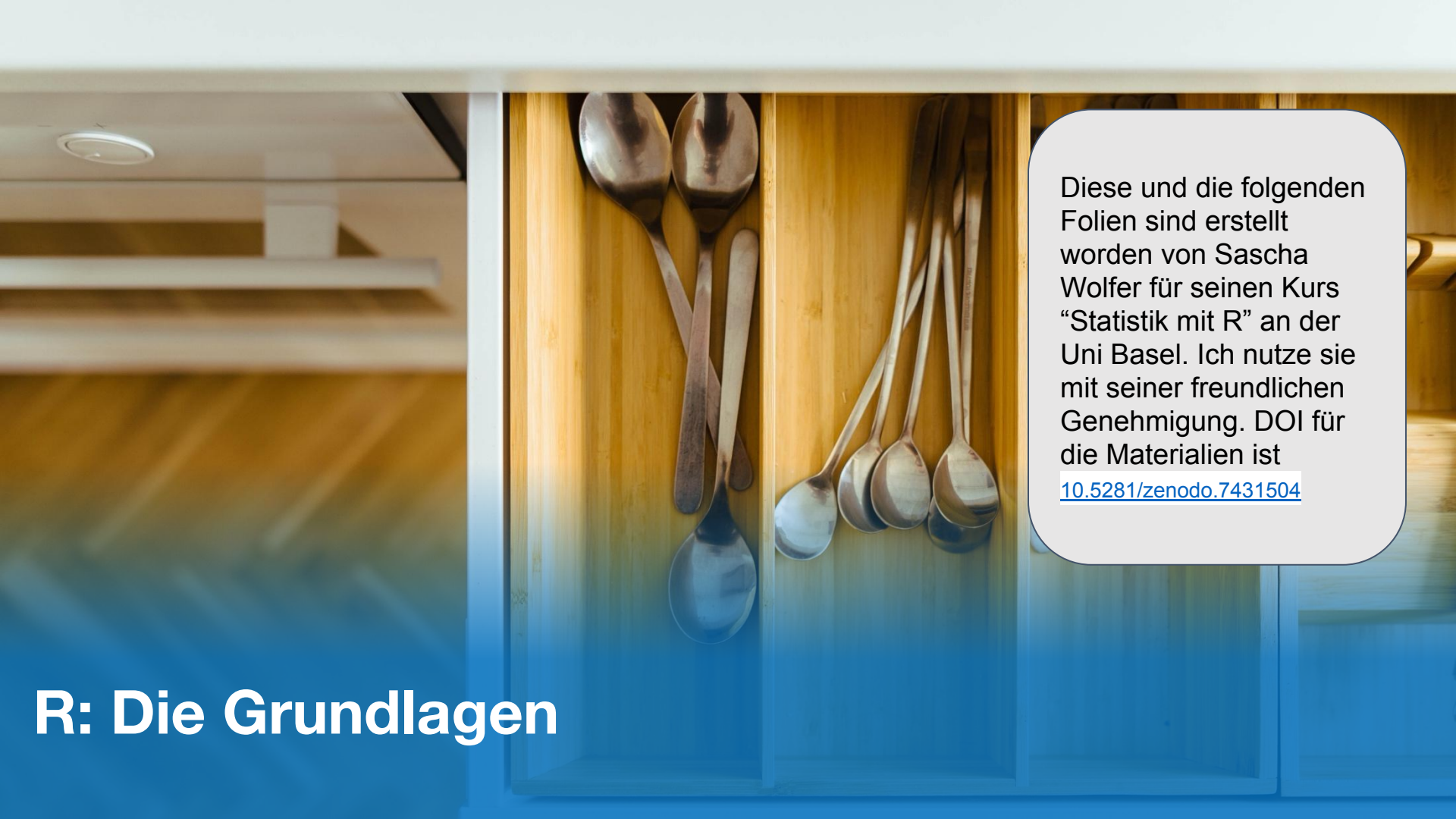
Einführung in die Statistik

Praktische Übung – Jürgen Hermes – IDH – SoSe 2023

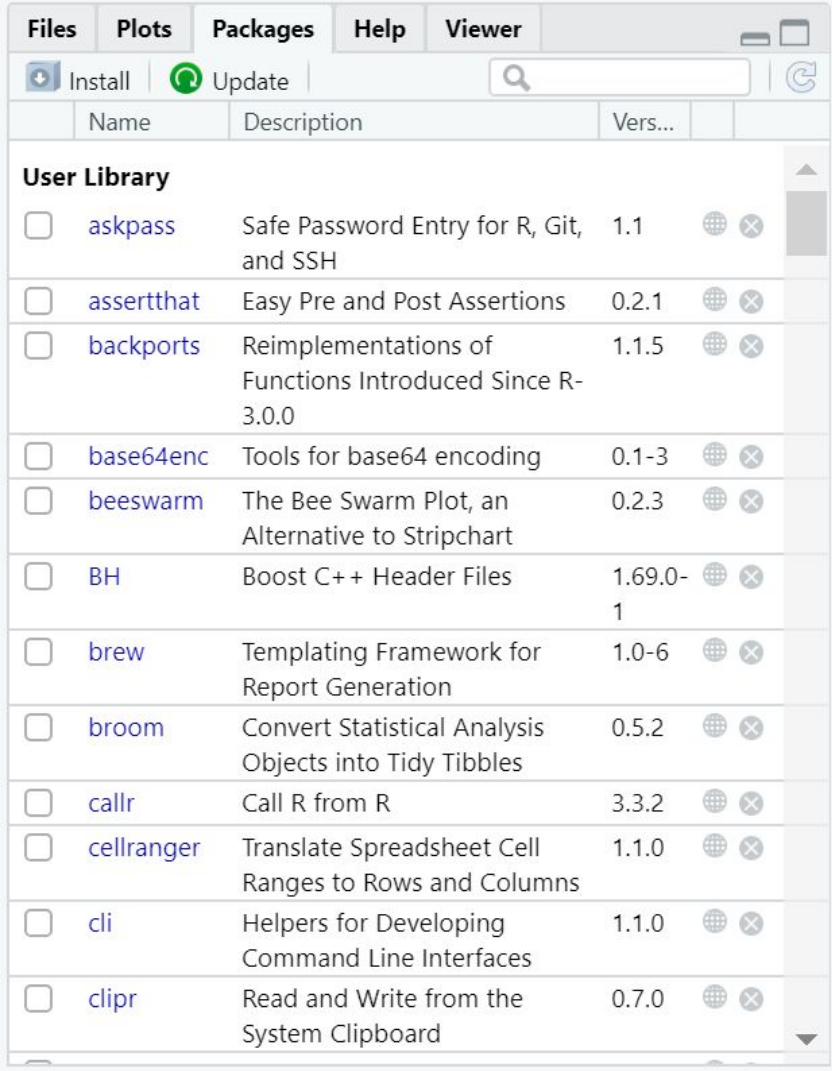


R: Die Grundlagen

Diese und die folgenden Folien sind erstellt worden von Sascha Wolfer für seinen Kurs “Statistik mit R” an der Uni Basel. Ich nutze sie mit seiner freundlichen Genehmigung. DOI für die Materialien ist 10.5281/zenodo.7431504

Zusatzpakete

- R kann mit (sehr vielen) Zusatzpaketen erweitert werden.
- Befehl: `install.packages("<Paketname>")`
- "Packages" pane in RStudio
- Packages werden häufig mit geschweiften Klammern genannt: {beeswarm}
- `data.table::fwrite()` bedeutet: Die Funktion `fwrite()` aus dem Paket {data.table}
- Installierte Pakete laden mit `library(<Paketname>)`



Übung

- Installieren Sie das Paket `{openintro}`.
- Schauen Sie sich die ersten 6 Zeilen des Dataframes `cia_factbook` an.
- Lassen Sie sich die Zeile für Deutschland ausgeben.
- Lassen Sie sich die Zeilen für die Schweiz, Österreich und Deutschland ausgeben.
- Berechnen Sie den Anteil von Internetbenutzer*innen für jedes Land, speichern Sie diesen Wert in der Spalte `www_percent`.
- Welches Land hat die größte Fläche im Datensatz?
    - Hier taucht ein Problem mit nicht definierten Werten (NA) auf, das man mit is.na() behandeln kann

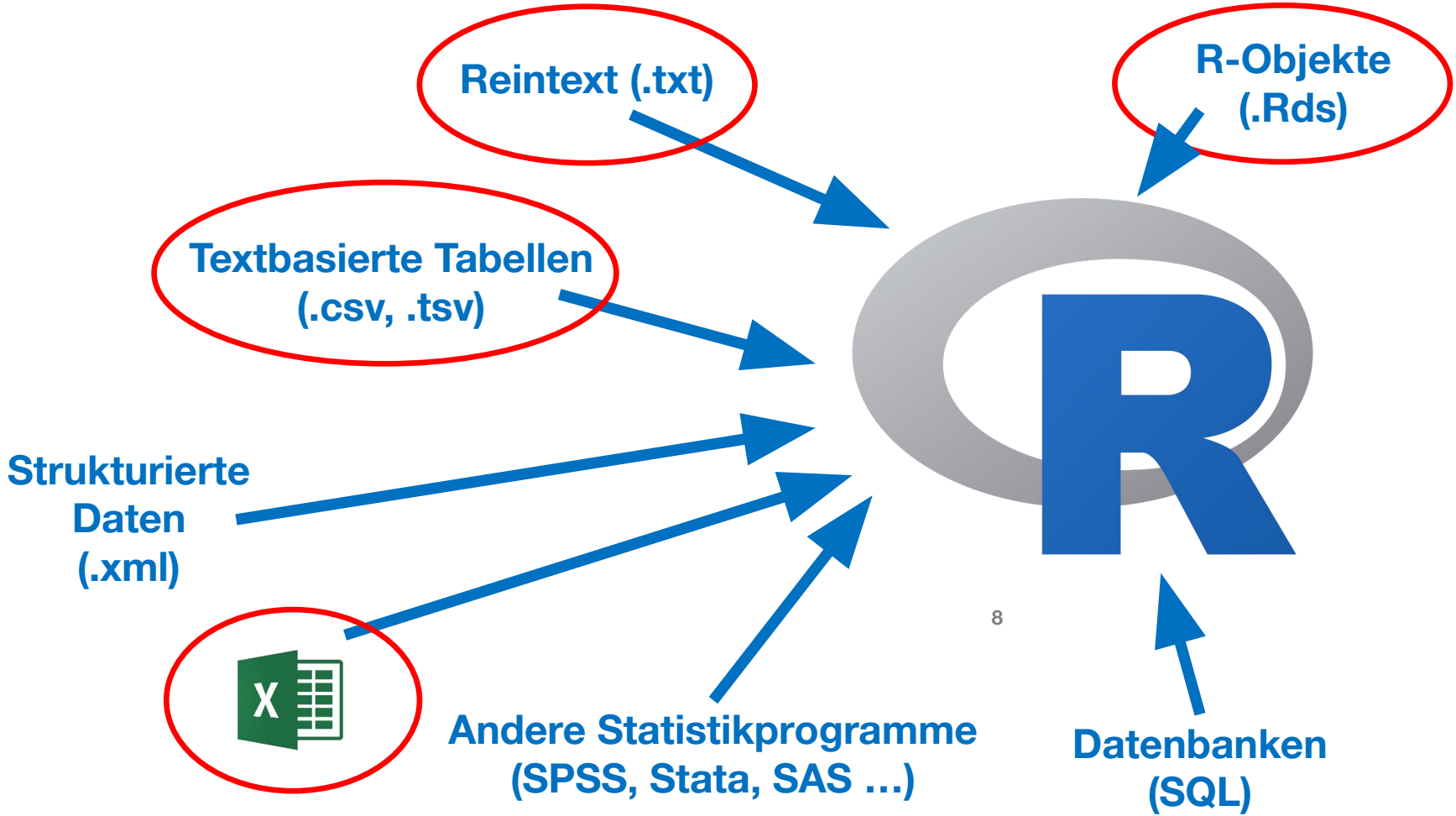
Daten einlesen

R aktualisieren

- Sollten Sie Probleme beim Installieren von Paketen haben, versuchen Sie bitte zunächst, R zu aktualisieren.
  - https://cloud.r-project.org
  - Nicht RStudio, sondern R!
- Die aktuelle R-Version (03.05.2023) ist **R 4.3.0** "Already Tomorrow"
- Ihre R-Version erscheint beim Startup. Sie können sie auch abrufen mit `R.Version()` – dort unter `"version.string"`

Bisher...

- ... haben wir mit `c()` oder `data.frame()` Datensätze selbst erstellt oder Datensätze aus Paketen verwendet.
- Meistens kommen Daten aber "von außen". Welche Daten können das sein?
  - "Rohe" Sprachdaten (z. B. Textdateien)
  - Ausgaben aus Korpusplattformen (z. B. Frequenzlisten)
  - Experimentaldaten (Reaktionszeiten, Antworten, ...)
  - Selbsterstellte Tabellen (z. B. händische Annotationen)
  - Webseiten ("Scraping")
  - ...


Reintext (.txt)
R-Objekte (.Rds)
Textbasierte Tabellen (.csv, .tsv)
Strukturierte Daten (.xml)
Andere Statistikprogramme (SPSS, Stata, SAS …)
Datenbanken (SQL)

Reintext einlesen mit `scan()`

- Liest Dateien ein und speichert das Ergebnis in einem Vektor.
- Wir wollen **Text** einlesen, dann lautet die Syntax:
  - `scan(<Datei>, what = "character")`
- `scan()` kann auch Textdateien direkt aus dem Internet lesen:

```
udhr <-
scan("http://research.ics.aalto.fi/cog/data/udhr/txt/ger.txt",
                what = "character", fileEncoding = "UTF-8")
```

Exkurs: Häufigkeitstabellen mit `table()`

- `table()` erstellt aus einem Vektor eine Häufigkeitstabelle.

```
table(c("B", "A", "B"))
```

```
A B
1 2
```

```
udhr <-
scan("http://research.ics.aalto.fi/cog/data/udhr/txt/ger.txt",
                what = "character", fileEncoding = "UTF-8")

tab <- table(udhr)

tab

sort(tab, decreasing = T)[1:10]
```

Exkurs: Häufigkeitstabellen

In korpuslinguistischen Untersuchungen werden Wörter oft in Kleinschreibung überführt.
In R: `tolower(<Vektor>)`

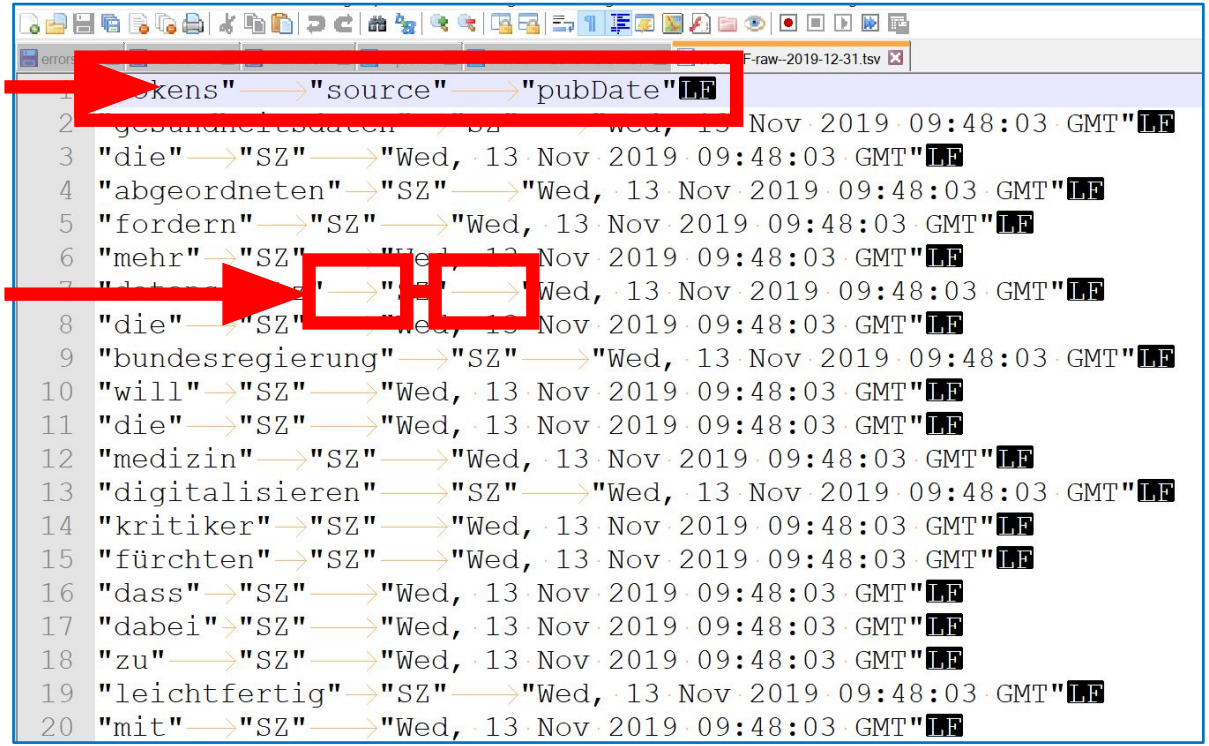
- Später werden wir noch lernen, wie wir die Häufigkeitstabelle für Wörter noch etwas "sauberer" hinbekommen.
  - z. B. keine Satzzeichen, die an Wörtern hängen ("ging," vs. "ging")
- Tables ≠ Dataframes
- Vektoren mit benannten Elementen
- Deshalb auch indizierbar über die Namen der Elemente
- Wie oft kommt das Wort "Generalversammlung" vor?

```
tab["Generalversammlung"]
```

Textbasierte tabellarische Daten

Header/ Variablennamen

Separator/ Spaltentrenner



Textbasierte tabellarische Daten

- "comma-separated values" (CSV)
- Solche tabellarisch aufgebauten Daten lassen sich u.a. mit der Funktion `vroom()` aus dem Package `{vroom}` einlesen.

  ```
  dat <- vroom(<Datei>)
  ```

- Die Funktion "rät" ziemlich gut selbst, wie das genaue Format der Datei ist (z. B. welcher Separator verwendet wird).
  - Falls es mal schief geht: `?vroom`
- `vroom()` gibt einen "Tibble" aus.
  - Tibbles sind (fast) wie Dataframes, die Ausgabe ist nur etwas komfortabler. Falls Sie lieber mit einem Dataframe arbeiten wollen:
    ```
    dat <- as.data.frame(vroom(<Datei>))
    ```

Übung

- Laden Sie die Datei `conll.tsv` herunter (ILIAS)
- Lesen Sie die Datei in die Variable `data` ein. Schauen Sie sich die ersten 6 Zeilen an.
- Wie viele Zeilen hat die Datei ingesamt?
- Erstellen Sie eine sortierte Häufigkeitstabelle der Spalte `POS`. Welches POS kommt am häufigsten in `conll.tsv` vor?

Excel-Dateien einlesen

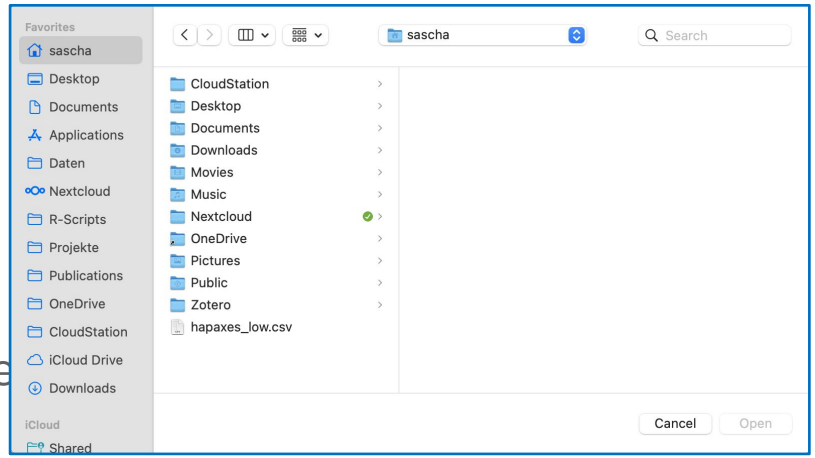
- Paket: `{readxl}`, Funktion: `read_excel()` – Argumente:
  - `path`: (chr) Datei, die eingelesen werden soll (`.xls`/`.xlsx`)
  - [`sheet`: (chr/num) Arbeitsblatt, das eingelesen werden soll]
  - [`range`: (chr) Zellenausschnitt, der gelesen werden soll]
    - Zellausschnitt wird angegeben wie in Excel (z. B. `"B4:E20"`)
  - [`col_names`: (log) Erste Zeile Spaltenüberschriften/Variablennamen?]
  - weitere Argumente: `?read_excel`
- `read_excel()` gibt einen Tibble zurück, `as.data.frame()` um diesen in einen Dataframe umwandeln.

Dateiauswahl-Dialog

- `file.choose()` startet einen Dateiauswahl-Dialog.
- Damit können Sie eine Datei zum Einlesen auswählen.

```
vroom(file.choose())
read_excel(file.choose())
```

- Nachteil: Wenn Sie ein Script nochmal laufe startet jedes Mal der Dateiauswahl-Dialog.
  - Deshalb ist es eigentlich besser, den Pfad der Datei in das Script zu schreiben.



R-Objekte einlesen

- R-Objekte (Variablen) kann man in Dateien speichern. Diese Dateien haben die Endung `.Rds`.

  ```
  dat <- readRDS(<Datei>)
  ```

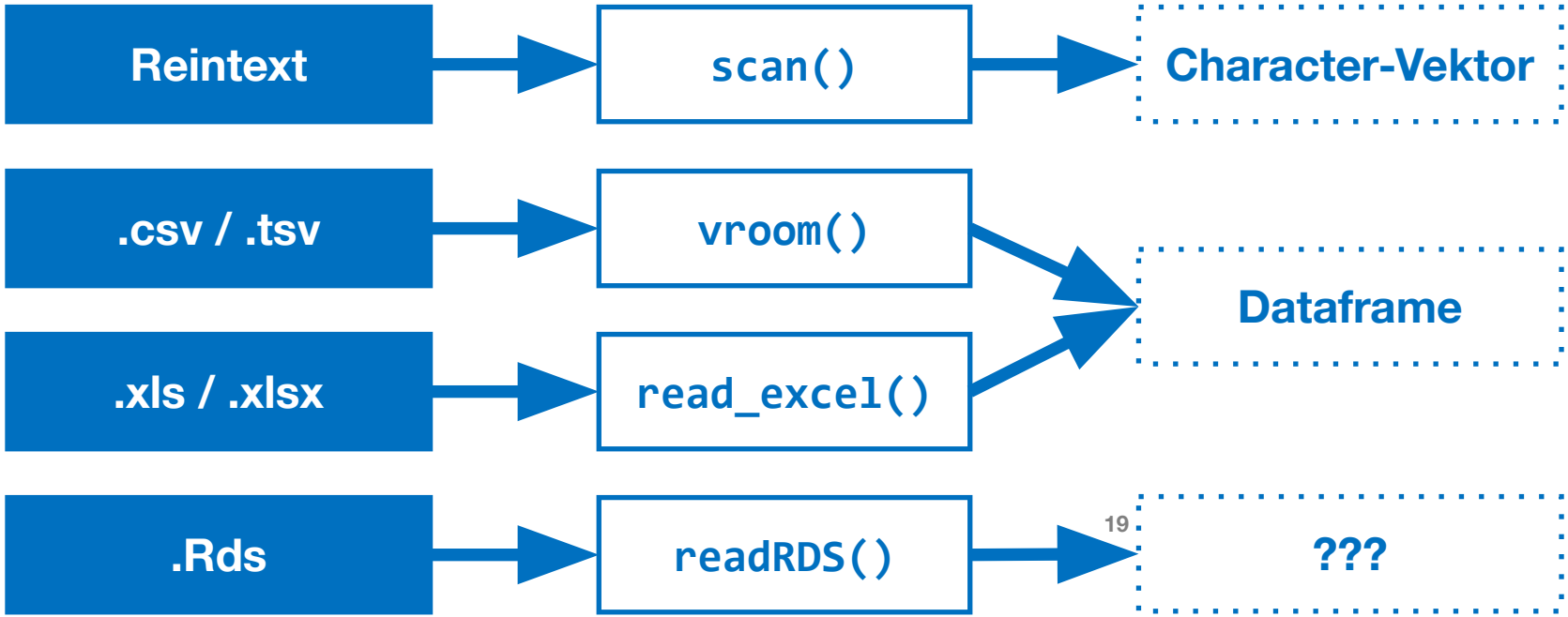
- Diese Dateien können nur mit R eingelesen werden, eignen sich also nicht zum Austausch mit anderen Programmen.
- Dafür kann relativ wenig schiefgehen.
- Man kann alle Datentypen in Rds-Dateien abspeichern.
  - Nicht nur Dataframes, sondern auch Vektoren, Listen, Matrizen usw.

Übung

- Laden Sie die Datei `cit.Rds` (ILIAS) herunter und lesen Sie die Datei mit `readRDS()` ein.
  - Speichern Sie das Ergebnis in der Variable `cit`.
- Geben Sie ein:

```
barplot(cit$einw, names.arg = cit$name)
```

Einlesen: Zusammenfassung



Daten ausgeben

Daten ausgeben: Funktionen

Reintext	`cat()`
.csv / .tsv	`vroom::vroom_write()`
.xls / .xlsx	`WriteXLS::WriteXLS()`
.Rds	`saveRDS()`

Reinen Text ausgeben mit `cat()`

- Es kommt nicht sonderlich oft vor, dass Sie aus R laufenden Text ausgeben müssen.
- Wenn, dann am einfachsten mit

```
cat(<Vektor>, <Datei>, sep = <Separator>)
```

- Ausgeben der Buchstaben a bis z in einzelnen Zeilen der Datei `buchstaben.txt`:

```
cat(letters, file = "buchstaben.txt", sep = "\n")
```

CSV-Dateien schreiben mit `vroom_write()`

- `vroom_write()` schreibt Dataframes in CSV-Dateien.

  ```
  vroom_write(<Dataframe>, <Dateiname>, delim =
  <Separator>)
  ```

  - Weitere Argumente: `?vroom_write`

R-Objekte schreiben mit `saveRDS()`

- Wenn Sie ein beliebiges R-Objekt in einer .Rds-Datei speichern möchten, benutzen Sie

  `saveRDS(<Variable>, <Datei>)`

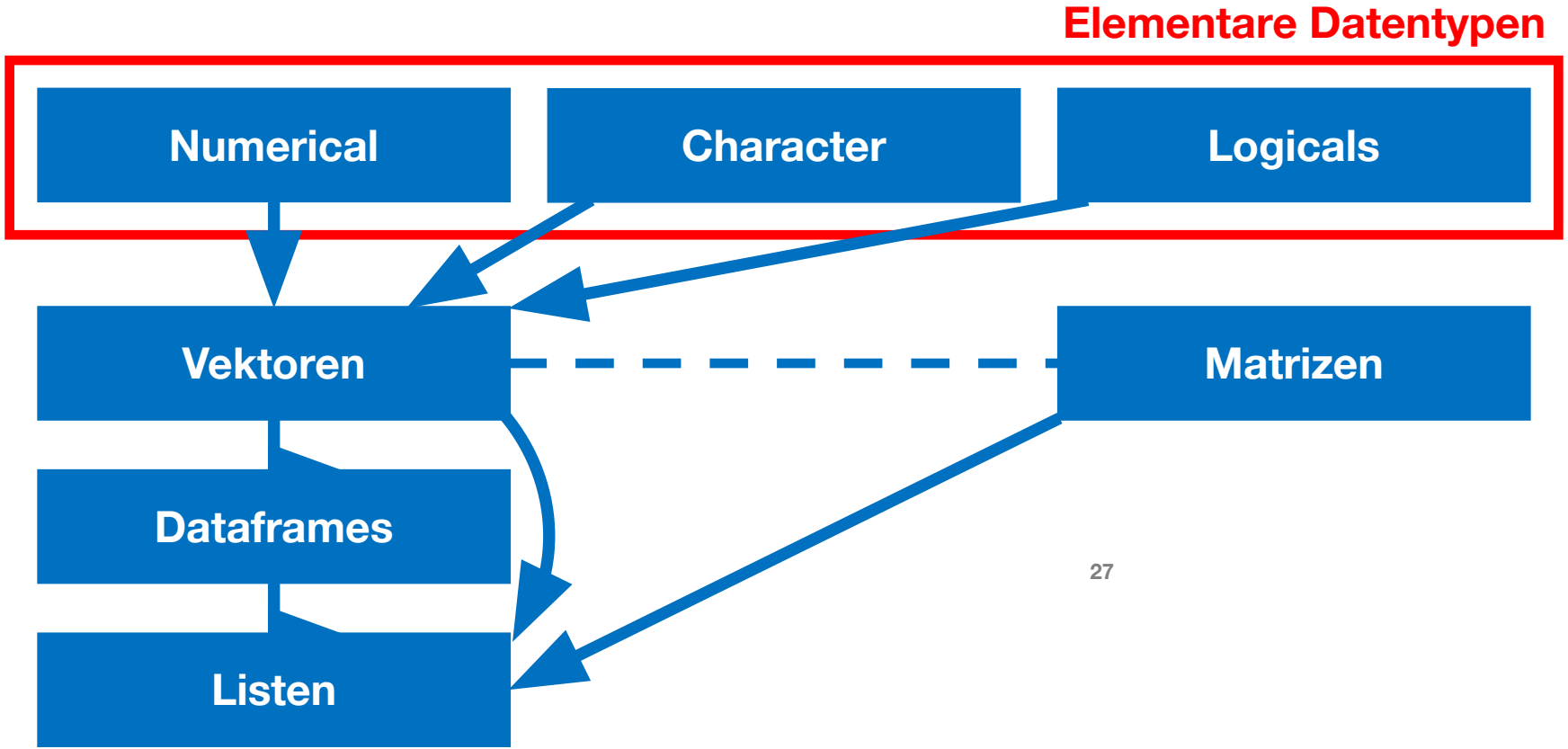
- Diese Dateien können nur von R wieder eingelesen werden.
  - Erinnerung: Einlesen mit `readRDS()`

R-Einführung: Zusammenfassung

Oberfläche

- Befehle sammeln im Script-Editor, **jetzt** mit # kommentieren und sich später freuen!
- Pakete installieren mit `install.packages()`
- Hilfe abrufen mit `?<Funktion>` oder Hilfe durchsuchen mit `??<Suchterm>`

Datentypen


Elementare Datentypen
Numerical
Character
Logicals
Vektoren
Matrizen
Dataframes
Listen

Vektoren ...

- ... bestehen immer aus *einem* elementaren Datentyp.
- ... werden automatisch **recyclet**.
- ... können **Faktor**vektoren sein, um nominal-/ordinalskalierte Daten zu repräsentieren.

Indizierung

- Zugreifen auf/Selektieren von Elemente(n) in komplexen Datentypen.

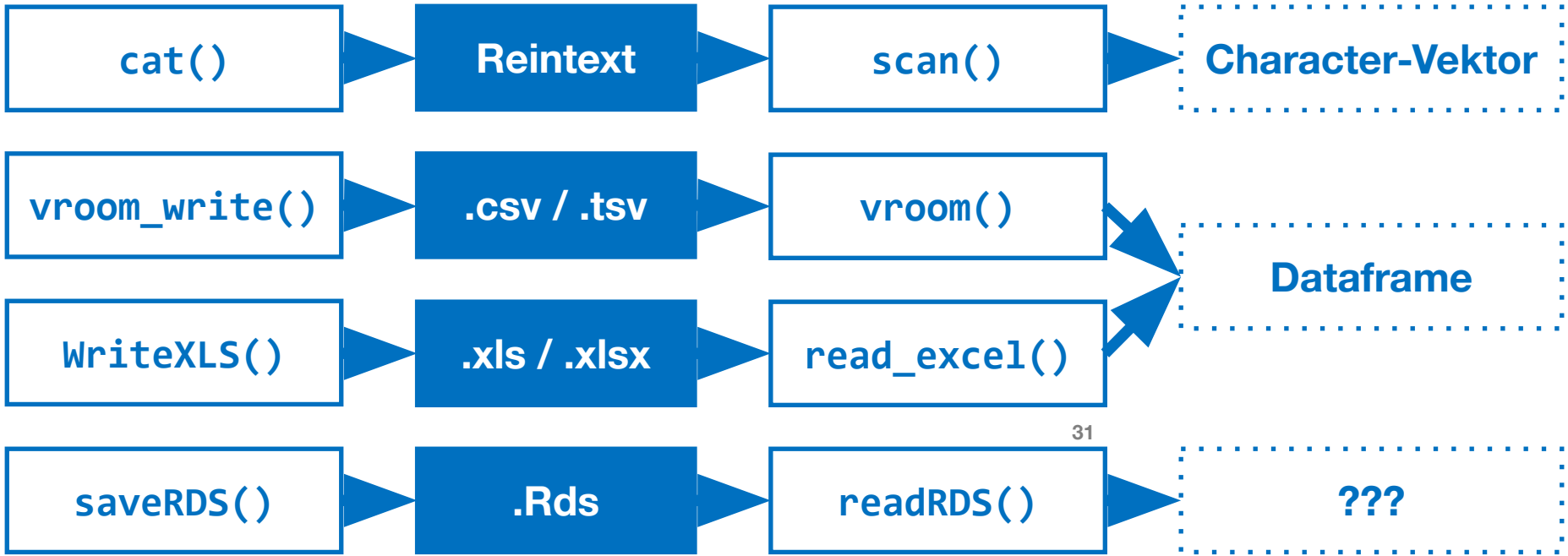
`[]` `[,]` `$` `[[]]`

Wir können über Zahlen, Namen und Wahrheitswerte indizieren.

Funktionen

- Funktionen haben **Argumente**.
- Funktionen können **verschachtelt** werden.
- **Prädikate** sind spezielle Funktionen, die immer einen Wahrheitswert zurückgeben.

Einlesen & Ausgeben



> If you're not failing 90% of the time, then you're probably not working on sufficiently challenging problems.
>
> - Alan Kay

Hausaufgabe

- Abgabemodalitäten finden Sie auf ILIAS
- Ist relativ ähnlich der Übung P06F04 gestaltet
- Sie benötigen das gleiche Paket {openintro}.
- Speichern Sie den data.frame “fastfood” auf einer eigenen Variable.
- Geben Sie alle Produkte (Items) des Restaurants “Burger King” aus, die mehr als 1000 Kalorien enthalten.
- Geben Sie die Items mit den höchsten Werten für Vitamin C aus. Nutzen Sie dafür die Funktionen “order()” und “head()”
- Berechnen Sie für alle Items den Quotienten aus protein und calorien. Speichern Sie diese in einer Spalte $protPerCal
- Ermitteln Sie für jedes Restaurant, wie viele Items es anbietet. Geben Sie für diese Werte einen Barplot aus. Nutzen Sie dafür die Funktionen table() und barplot()
- Speichern Sie ihr data.frame als R-Objekt und als tsv-Datei.