# Anmerkungen zum bisherigen Code (und allgemeiner Teamarbeit)

## Datenstruktur und -format

- darf in keinem Fall abgeändert werden, ohne dass alle Beteiligten informiert sind
- Analoges gilt für die Funktionsdefinitionen und deren Schnittstellen

## R-Verzeichnis

- exakt vorgegebene Struktur
- `./R`: nur Dateien mit `R`-Code und Endung `.R` oder `.r`
- `./man`: nur manual-Dateien mit Endung `.Rd`
- `./data`: Datensätze für den user, die er mit `data(...)` zuladen kann. Sie müssen auf den man-Seiten ausführlich beschrieben sein.
- `./inst` enthält spezielle Dateien, die besonders geladen werden
- `.Rbuildignore` enthält (Dateien und) Verzeichnisse, die nicht eingebunden werden sollen

## Boolesche Logik

Sei $A$ ein boolscher Wert ($T$, $F$)

- $A \wedge T$, $\{A = T\}$, $\{\text{if } A \text{ then } T \text{ else } F\}$ haben alle den Wert $A$
- $\{A = F\}$ hat den gleichen Wert wie $\neg A$
- *tertium non datur* (außer `NA`)

## `if`-Anweisung

- `if`-Anweisung hat in R einen Rückgabewert
- Unterscheide: `if` (Boolesche Aussage) und `ifelse` (Operator auf einem booleschen Vektor)

## User vs. Programmierer

- Verwende nie als Programmierer `T` für `TRUE` oder `F` für `FALSE`
- `data.frame` ist was für Nutzer
- Argumentnamen müssen von Programmierern voll angegeben werden (von allen gemacht!)

## Programmierstil, Teil 1

- Code nie doppeln. (Selbst in `*.Rd`-Dateien nicht.)
- `eval(parse(...))` ist *ultima ratio*
- aufwendigere Resultate nicht mehrfach berechnen
- statt `tryCatch` ist `try` u.U. adäquater
- unterscheide `is.null(arg)` und `length(arg) == 0`
- Konstanten nicht mitten im Code definieren, sondern als Argument mit default oder am Anfang der Funktion.
- Konstanten nicht neu definieren, wenn sie sich aus vorhandenen Konstanten berechnen lassen (ausser es ist sehr aufwändig; dann sind aber auch Kommentare zwingend)

## Programmierstil, Teil 2

- jegliche Zeile hat nur 80 Zeichen
- wenn Länge eines Ergebnisvektors bekannt ist, im Vorab so definieren, z. B. `ans <- vector('list', length(x))`
- Verschachtelungen und `if`-Ketten vermeiden versuchen
- '.' und '_' sind am Wortanfang und -ende tabu
- Sämtliche Ausgaben muss der Nutzer abstellen können (unterscheide Verwendung von `print`, `cat`, `message`)
  Ausnahme: Ausgaben.
  Achtung! `summary` berechnet nur die Ausgabe, gibt aber nichts aus.
- wenn Argumente bekannt (und wenig) sind, diese nicht durch '...' abfangen/abfragen