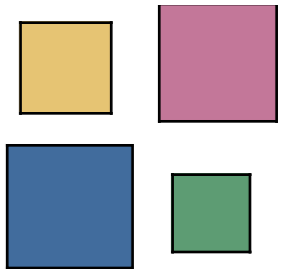


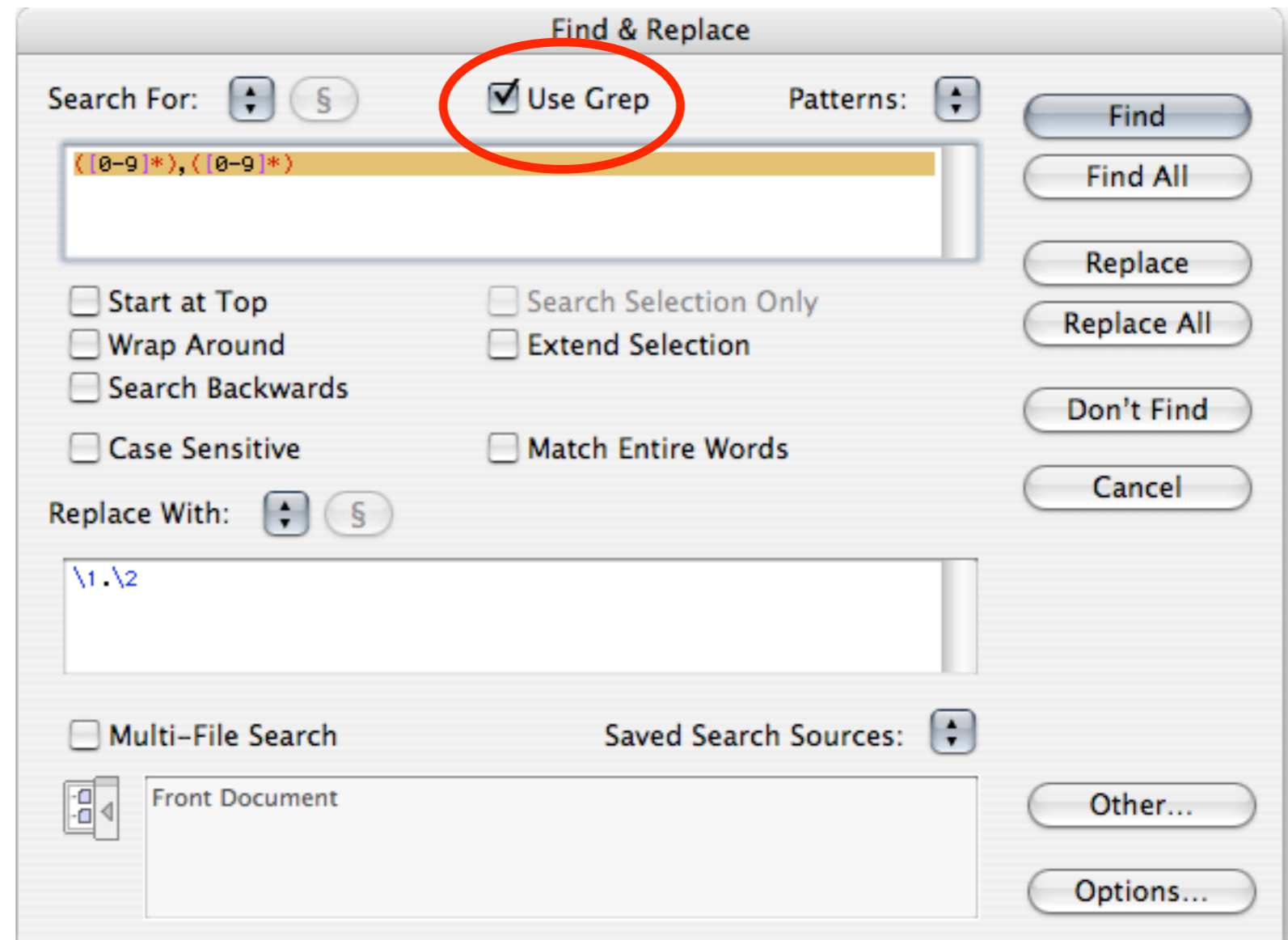
ASCII Daten: Reguläre Ausdrücke

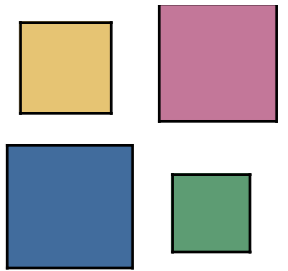
- “Normales” Suchen und Ersetzen kann nur mit **Konstanten** arbeiten ...
- ... variable Treffer, Sonderzeichen oder Positionen in Zeilen können nicht spezifiziert werden.
- Reguläre Ausdrücke erweitern die möglichen Suchbegriffe.
- Reguläre Ausdrücke sind bei den meisten UNIX Applikationen Standard – einen Standard jedoch gibt es leider **nicht!**
- Reguläre Ausdrücke sind nicht mit “Wildcards” im Dateisystem zu verwechseln ...
(... auch wenn sie eine gewisse Ähnlichkeit haben)
- Reguläre Ausdrücke werden von einigen Editoren (auch unter Windows und Mac) unterstützt.



Reguläre Ausdrücke: Software

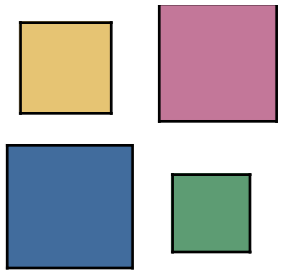
- UNIX
 - vi
 - ed
 - ...
- Windows
 - EditPad Pro
 - ...
- Mac
 - Text Wrangler
 - ...
- Alle Systeme
 - perl
 - Java
 - R
 - ...





Reguläre Ausdrücke

- Reguläre Ausdrücke bestehen aus 3 wichtigen Komponenten:
 - **Anchors** (Anker)
Anker spezifizieren eine Position in einer Zeile (Anfang, oder Ende) auf die sich der Suchausdruck bezieht.
Beispiel: `^A` findet alle Zeilen die mit “A” beginnen
 - **Character Sets** (Alphabete)
Alphabete definieren, welches Zeichen gefunden werden kann.
Beispiel: `[0-9]` findet jegliche Ziffer
 - **Modifier**
Modifier geben an, wie oft ein bestimmter Ausdruck gefunden werden soll.
Beispiel: `[0-9]*` findet keine, oder mehrere Ziffern

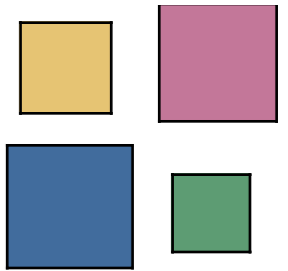


Reguläre Ausdrücke: Anker

- Anker dienen zur Positionierung eines Suchbegriffs:
 - `^` entspricht dem Zeilenanfang
 - `$` dem Zeilenende
 - Neuere Implementationen erlauben weiter:
 - `\A` Anfang des Textes
 - `\Z` Ende des Textes
 - `\b` eine Wortgrenze
 - `\B` Komplement von `\b`, d.h. alles außer einer Wortgrenze
- Bemerkung

Sollen `^` oder `$` gefunden werden, so müssen sie mit dem Escape Character `\` geschützt werden.

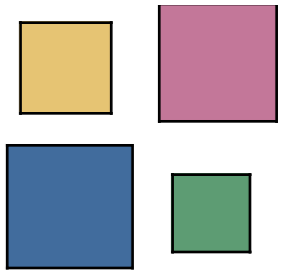
Nach `\` kann nur mittels `\\` gesucht werden.



Reguläre Ausdrücke: Anker

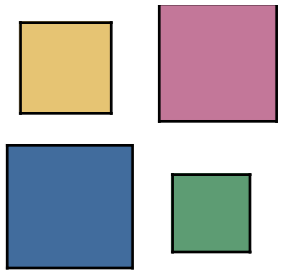
- Beispiele

- `^A` findet alle Zeilen die mit "A" beginnen
- `A$` findet alle Zeilen die mit "A" ende
- `A^` findet alle Vorkommnisse von `A^`
- `$A` findet alle Vorkommnisse von `$A`
- `^\^` findet ein `^` am Anfang der Zeile
- `^^` dito
- `\$$` findet ein `$` am Ende der Zeile
- `$$` dito
- `\AIm Anfang` findet `Im Anfang` nur am Anfang des Textes
- `\bder\b` findet alle Stellen wo das Wort `der` steht, aber z.B. nicht `der` im Wort `Wanderer`



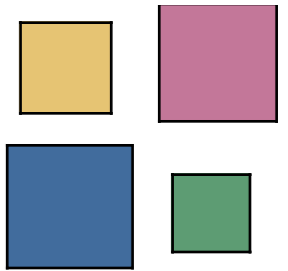
Reguläre Ausdrücke: Alphabete

- Alphabete definieren, welches Zeichen gefunden werden kann
- Ein beliebiges Zeichen wird mit `.` gefunden
- Die Zeichen eines Alphabets müssen in eckigen Klammern stehen.
- Ausnahmen:
 - beginnt ein Alphabet mit `[^` dann wird alles **außer** die spezifizierten Zeichen gefunden.
 - `-` wird zur Bereichsdefinition benutzt
- Beispiele:
 - `[0-9]` findet **eine** Ziffer
 - `[^0-9]` findet alle Zeichen außer Ziffern
 - `[-0-9]` eine Ziffer oder ein `-`
 - `[0-9A-z]` eine Ziffer oder ein Buchstabe
 - `[]a-z]` ein Kleinbuchstabe oder eine `]`



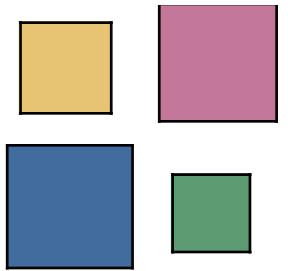
Reguläre Ausdrücke: Vor definierte Werte

- Bestimmte, häufig benutzte, Alphabete sind schon vor definiert:
 - `\s` irgendein Whitespace
 - `\S` Komplement von `\s`, d.h. alles außer einem Whitespace
 - `\w` irgendein “Wort-Zeichen” (`a-z`, `A-Z`, `0-9`, `_`, ...)
 - `\W` Komplement von `\w`
 - `\d` eine Ziffer
 - `\D` Komplement von `\d`, d.h. alles außer einer Ziffer
- Beispiele:
 - `\w+` findet einzelne Wörter,
wäre sonst: `\b[A-z_]+?\b`
 - `\d+` findet einzelne ganze Zahlen



Reguläre Ausdrücke: Modifizier

- Modifizier geben an, wie oft ein Ausdruck gefunden werden soll
- * besagt, dass der **davor** stehende Ausdruck nicht, oder beliebig oft gefunden werden soll.
- + der davor stehende Ausdruck wird **mindestens** einmal gefunden
- ? der davor stehende Ausdruck wird maximal einmal gefunden
- Eine genaue Trefferanzahl von **n** Treffern wird in manchen Programmen mit $\{n,m\}$ oder $\{n,m\}$ angegeben.
- Beispiele:
 - $[0-9][0-9]^*$ findet jede positive ganze Zahl
 - $^{\#}$ findet jede Zeile
 - $[0-9]\{3,5\}$ 3, 4 oder 5 Ziffern in Folge
 - * jede Zeile, die ein * beinhaltet

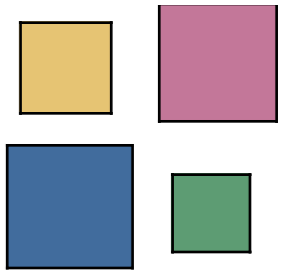


Reguläre Ausdrücke: Sonderzeichen etc.

- Bestimmte Zeichen werden mittels des Escape Characters dargestellt:
 - `\t` Tabulator
 - `\n` Newline (UNIX Zeilenende)
 - `\r` Carriage Return (Mac Zeilenende)
 - `\f` Seitenende
- Nicht zu verwechseln mit der Verwendung von `\` um reservierte Zeichen zu benutzen.
- Klammern haben ihr “normale” Bedeutung zur Gruppierung
- `|` unterscheidet zwischen zwei Alternativen

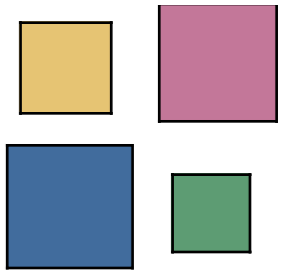
Beispiel:

- `d(er|ie|as)` findet einen Artikel



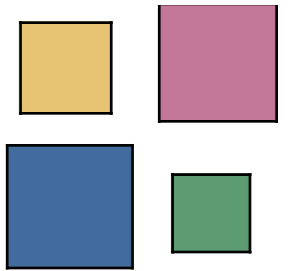
Reguläre Ausdrücke: Treffer wiederverwerten

- Für “Suchen und Ersetzen” ist entscheidend, dass man die variablen Suchergebnisse auch wieder verwenden kann.
- Um sich beim Ersetzen auf schon gefundene Komponenten zu beziehen, müssen diese in `\ (` und `\)` eingeschlossen sein.
(Oft auch nur `(` und `)`)
- Mittels `\n` kann dann auf die `n`-te so spezifizierte Komponenten zugegriffen werden.
`&` steht für den gesamten Ausdruck.
- Beispiele:
 - `\([aeiou])\1` findet alle doppelten Vokale
 - `([-0-9]*) , ([0-9]*)` wird gesucht
`\1.\2` wird ersetzt



Reguläre Ausdrücke: Non-Greedy Modifiers

- Häufigstes Problem bei regulären Ausdrücken:
Es wird **zu viel** gefunden
- **Regel**
Es ist sicherer zu überlegen, was man **nicht** finden will,
als was man finden will.
- **Problem:** “Longest Match”
z.B. in “**AAA AAA AAA**” findet “.+” “**AAA AAA**” nicht aber “**AAA**”
- **Lösung:** “Faule Auswertung”
Wenn Modifier bzw. Ausdrücke von einem ? gefolgt werden, dann
“begnügt” sich der Ausdruck mit dem ersten erfüllenden Treffer,
d.h.
“**AAA AAA AAA**” findet “.+?” nun nur “**AAA**”



Reguläre Ausdrücke: Beispiele I

- Formatieren einer Namensliste

Das Format “**Hans Mustermann**” soll zu “**Mustermann, Hans**” geändert werden.

suchen nach $^(.*) ([^]+)$$
ersetzen mit $\2, \1$

- Finden einer Fließkommazahl

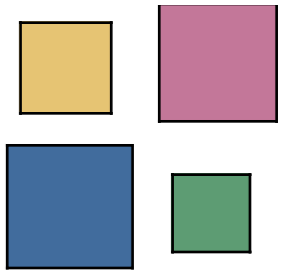
Erster Ansatz: $[-+]?[0-9]*\.[0-9]*$

Problem: Alles ist optional

Besser: $[-+]?([0-9]*\.[0-9]+|[0-9]+)$

Optimiert: $[-+]?([0-9]*\.)?[0-9]+$ bzw. $[-+]?(\d*\.)?\d+$

Nicht gelöst: Wortgrenzen!



Reguläre Ausdrücke: Beispiele II

- Aufteilung einer IP-Adresse

In einer Spalte steht eine IP-Adresse wie **192.168.1.100**, die in 4 Spalten, durch Tabulatoren getrennt, aufgeteilt werden soll.

suchen nach: `\b(\d{1,3})\.\d{1,3})\.\d{1,3})\.\d{1,3})\b`

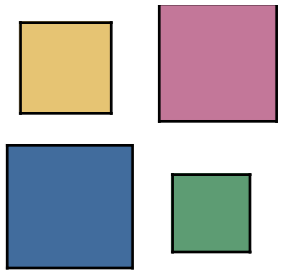
ersetzen mit: `\1\t2\t3\t4`

- Löschen einer Spalte

In einem Datensatz soll die 3. Spalte gelöscht werden:

suchen nach: `^(.+?\t){2}(.+?\t)`

ersetzen mit: `\1`



Reguläre Ausdrücke in R

- `grep(...)`

`grep(pattern, x, ...)` sucht im Text `x` nach `pattern`

Beispiel:

```
txt <- c("arm", "foot", "lefroo", "bafoobar")
if(any(i <- grep("foo", txt)))
  cat("'foo' appears at least once in\n\t", txt, "\n")
i # 2 and 4
txt[i]
```

- `(g)sub(...)`

`sub(pattern, replacement, x, ...)` ersetzt im Text `x` das `pattern` durch `replacement`

Beispiel:

```
str = 'Now is the time      '
sub(' +$', '', str) ## spaces only
```

`(gsub(...))` ersetzt **alle** Vorkommen des gefundenen Musters)