20.04.'17

CO-VL

Richtung:
Man erzählt was zur Projektaufgabe. Ca. zur Mitte des Semesters legt man sich fest.
Dazu erzählen und zur Beziehung zur VL setzen!

#3 Python + pandas → man kann so ähnlich wie in R arbeiten (Tabelle einer DB)

Exercises:

1. 4^n

2. AA/TT, AC/TG, AG/TC, **AT/TA**; CA/GT, **CG/GC**, CC/GG; ~~TC/AG~~, **TA/AT**, **AC/CG**

$\frac{12}{2} + 4 = 10$

bei ungerade $\frac{\#RNA}{2}$

bei gerade komplizierter

⇒ deswegen arbeitet man eigtl. nur mit ungerade Nukleotid-Anzahl

Bei Genom-Assemblierer kann man nur ungerades n angeben!

A Allgemeine Formel für gerade aufschreiben

$$\frac{4^n - 4^{\frac{n}{2}}}{2} + 4^{n/2}$$

$n/2$ geht auch, da n gerade sein muss.

bei n gerade

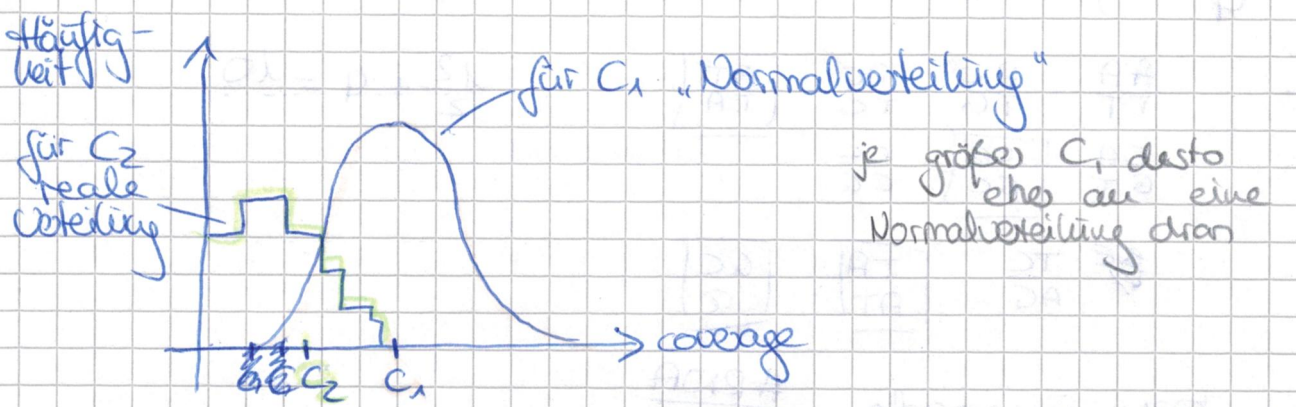
$$4^{\frac{n}{2}}$$

bei n ungerade

Kleine sequenzierte Fragmente DNA

N: # Fragmente
F: Fragmentlänge (100 - 1000)
G: Genomlänge ($10^6 - 10^9$)
Säugetiere, Pflanzen
[Viren: $10^3 - 10^4$]

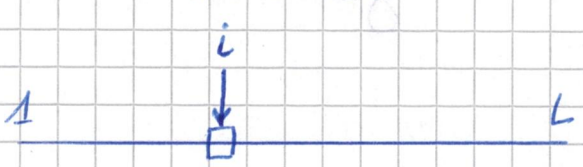
$$C = \frac{N \cdot F}{L}$$ durchschnittliche Coverage



Normalverteilung stimmt nicht ganz
- durch 0 beschränkt
- in Realität nur ganze Zahlen

Idee:

geworfene Bälle
NF Bälle zufällig auf L Positionen



Jeder Ball landet mit Wkt ½ bei i.
Wenn die Bälle unabhängig sind: (Was nicht ist, aber das ignorieren wir.)

Binomial-Verteilung!

Binomial-Verteilung

X: Anzahl Bälle bei i

$$P(X=k) = \binom{NF}{k} \left(\frac{1}{L}\right)^k \left(1-\frac{1}{L}\right)^{NF-k} \qquad k = 0, 1, \ldots, NF$$

Allg.: $B_{n,p}(k) = \binom{n}{k} p^k (1-p)^{n-k}$

$n \to \infty$
$p \to 0$
$np \to c$
wie wir uns das angücken

$NF = 10^{11}$
$k = 30$
$L = 3 \cdot 3 \cdot 10^9$

Zahlen sind sehr groß bzw. sehr klein.
⇒ beliebig viele Probleme beim numerischen Rechnen
⇒ approximieren

$$\binom{n}{k} = \frac{n!}{k!\,(n-k)!} = \frac{n^{\underline{k}}}{k!} \leftarrow \text{„n hoch k fallend“}$$

$$= \frac{n(n-1)(n-2)\ldots \text{[k Faktoren]}}{k!}$$

n sehr groß (10^{11}) $n = NF$
p sehr klein ($1/(3 \cdot 3 \cdot 10^9)$) $p = 1/L$
$np = C = z.B.\ 30$
↑ weder sehr groß, noch sehr klein, sondern ein vernünftiger Wert

Approximation:

$$\log\left[\frac{n^{\underline{k}}}{k!}\, p^k (1-p)^{n-k}\right]$$

Logarithmus macht aus Produkten Summen

$$= \log n^{\underline{k}} - \log k! + k \cdot \log p + (n-k) \cdot \log \cdot (1-p)$$

Was darf man approximieren und was nicht?
⇒ nicht „zu falsche“ Approximation, bzw. nicht zu große Fehler
z.B. $n^{\underline{k}} \approx n^k$ Unterschied nicht schlimm, wenn n sehr groß und k eher klein!
n^k $\quad n \cdot (-p)$

$$\approx k \log n - \log k! + k \log p - np$$

$$= k \log (np) - \log k! - np$$

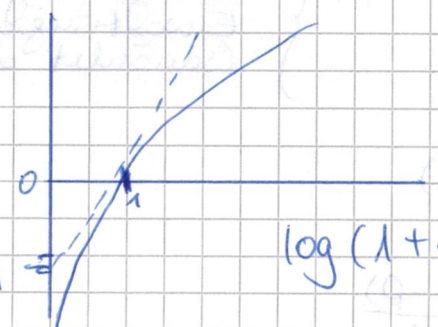
np ist endlich und eine konst./konkrete? Zahl!

$$= k \log C - \log k! - C$$

$$\approx (k \log k)$$

es gibt genauere Darstellungen von $\log k!$

„fast unendlich mal fast Null“



$\log(1+\varepsilon) \approx \varepsilon$

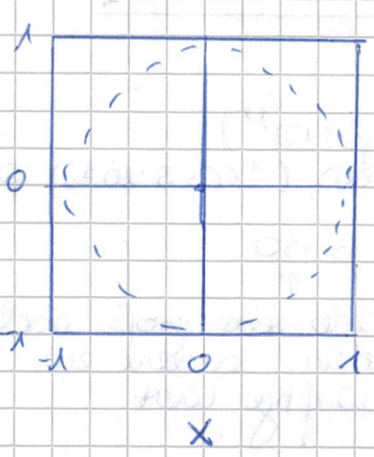
$$\exp[\cdot] = \frac{c^k}{k!} \cdot \exp(-c)$$

$$P(X=k) = \binom{NF}{k} \left(\frac{1}{L}\right)^k \left(1 - \frac{i}{L}\right)^{NF-k}$$

$$= \boxed{e^{-c} \cdot \frac{c^k}{k!}}$$ Poisson-Verteilung mit Erwartungswert c

(Approximation der Binomial-Verteilung mit entspr. Werten)

Wo kommt Poisson-Verteilung noch vor?



Kreis Fläche $\pi \approx 3.141$

Quadrat Fläche 4

Zufällig Punkte „reinschießen"

↳ mit gleicher Wkt 2 Koordinaten (je x, y ziehen)

$x^2 + y^2 \leq 1^2$ für jeden Punkt testen, ob im Kreis oder nicht

$p = \frac{\pi}{4}$

X: Punkte im Kreis

$EX: np \approx X$ Erwartungswert x

(np ≈ X: ungefähr Beobachtung X)

$\pi = 4p \approx \frac{4x}{n}$ genauer je größer n ist

mit Poisson-Verteilung approximieren!

x Poisson (λ)

$\mathbb{E}X = \lambda$

$\mathbb{V}ar X = \lambda$

} wichtige Eigenschaft: Erwartungswert und Varianz gleich (schränkt aber auch ein)

x Bin(n,p)

$\mathbb{E}X = np$

$\mathbb{V}ar X = np(1-p)$

„Gefühltes Praktikum"

EnsemblGenomes

> Ensemble Bacteria

E.coli

> Download genes ...

> DNA .fa.gz mit GZip FASTA-Datei

sm = softmasks → repeats mit Kleinbuchstaben
rm → +/- ersetzt mit x o.ä. Platzhalter Buchstaben
sonst → normal

toplevel → alles
nonchromosomal → nicht auf Chromosomen

⇒ Softmask am angenehmsten, da man noch selber entscheiden kann

Skript: → nur privat von Sven, stattdessen dinopy (offiziell)

```
from dnaio import fasta_read
with open
for entry in fasta_reads(fname):
    (header, seq)
        (:)
```

S. Elias

⇒ C+G und A+T sind „gleicher" zueinander als zu den anderen von der Anzahl her.

CO-VL

(...)

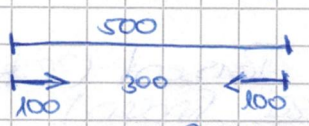
- DNA-Sequenzierung
  - pH-Wet → sehr ungenau
  - Illumina → viel teurer und viel genauer als mit pH i.H. ca. 80-90% aller Sequenzierungen (Fehlerrate < 1%)
  - durch Nano-Pore → neue Technik (Einzelmolekültechnik) (höhere Fehlerrate)

  => die meisten können mit einem Molekül nichts anfangen

  → PCR-Amplifizierung, um ausreichende Signalstärke zu bekommen

- PCR
  - zum Glück recht geringe Fehlerrate
  - Fehler am Anfang schwierig zu unterscheiden

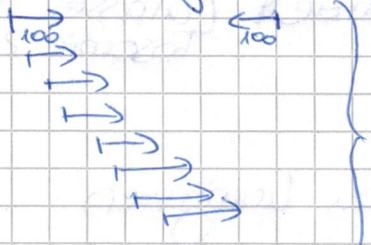
- Paired-end-sequencing



Stücke der DNA von beiden Enden

=> 2×100bp Stücke sequenziert

Man hat gewisse Fehlerraten (PCR, Sequ., ...)



durch Überdecken kann man Fehler korrigieren

Fragmente: 100-200bp

↑ mehrere Millionen davon

=> meist mit Qualitätsmerkmalen (Stichwort: FASTQ)

ERROR CORRECTION PROBLEM (#3)

- DE-DUPLICATION PROBLEM (#4)

  Ziel: unabhängige Sequenzen bekommen

  (PCR-Duplikate rausfiltern)

(#5) k-mer ≡ q-gram

- es ist in der Tat schwierig
- immer noch aktuelles Thema (wird immer noch viel publiziert)

- z.B. 16 Byte (oder mehr) pro k-mer und Zahl und man will schnellen Zugriff haben
  ⇒ so lange klein und ins RAM möglich, sonst schwierig

- was macht man?
  - k-mer als Integer (128 Bit ≙ ~~64~~ 63-mere)
    ↑ wir betrachten nur ungerade
    * k bis 31 gehe, da es in 64 Bit passt
    * jedes komplementäre Molekül! → 2 versch. Integer
      ↳ es wird nur das „kanonische Integer" betrachtet
      ⇒ man wählt den kleineren Integer-Wert
  - Datenstruktur
    * kleines k → Integer-Array
    * bei großen k geht das nicht 2^{62} (mehrere Exa-Byte)
      ⇒ k-mer codes hashen
    * Bloom-Filter
  - k-mere Poisson verteilt (mit Mittelwert x)

    solide vs. schwache k-mere
    ↑ vermutlich vorhanden ↑ wenn in PCR Fehler o.ä.

    ⇒ guter Schwellenwert nötig (hilft Poisson-Verteilung zu finden)

- biologische Effekte:
  - lokale Abschnitte sind doppliziert (krasser Fall: Trisomie 21)
- sequenzielle Effekte

⇒ ggf. nicht wegschmeißen, sondern korrigieren

schwieriger einen guten Schwellenwert zu finden

(#7) Methode für Pilen ggf. in Ordnung („nicht paradigmal „wegschmeißen")
⇒ dann den einfachsten

(#8) versch. Philosophien: Hash-Tabellen sollten zw. 50% und 70% gefüllt werden
(Hashing ist sehr weites und sehr interessantes Gebiet)

- Strategien um Kollisionen aufzulösen (z.B. Linear Probing)

Weiterverfachen Array ist die Adresse des Arrays

▷ Bloom-Filter (#9-11)

Schlüssel drin? nein → bin sicher das nicht drin
ja → könnte drin sein, muss aber nicht

↓

Wkt f. falsch positive ist bekannt

Schlüssel können aus Bloom-Filtern nicht wieder rausgenommen werden.

→ man kann bis 1 zählen

→ Es gibt Varianten von Bloom-Filtern, mit denen man auch weiter zählen kann

- Bloom-Filter ist Bit-Array

Bloom Filter B

| 0 | 0 | 0 | 0 | 1 | ... | 1 | 1 | 0 | 1 |

m Bits

Universum X: "beliebig" groß, alle Schlüssel,
z.B. $|X| = 4^{71}$

k Hashfkt: $h_i : X \to \{0, \dots, m-1\}$

B.add(x): Setze $B[h_i(x)] := 1 \quad \forall i = 1, \dots, k$

x in B? $\bigwedge_{i=1..k} B[h_i(x)]$

Was kann passieren?

Man muss versuchen, dass die Wkt, dass man zufällig nur 1en trifft sehr gering ist

=> größeres k => mehr Nullen

=> Nachteil: dauert länger, mehr Speicherplatz

Man kann das ausrechnen!

(Vgl. „Magie" in AIDS)

S. #11

- wenn man zu viele Werte einfügt, sind alle Werte auf 1
  ⇒ abschätzen wie viele Werte man einfügt
- false-positiv-Rate auf ε beschränkt
  ↳ eigentliche ε

Design-Aufgabe:
Wähle (m, k) so, dass FPR $\leq \varepsilon$, wenn $\leq n$ Schlüssel eingefügt werden.

Man rechnet approximativ unter Annahmen die nicht wirklich so eintreffen

n Elemente rein, ich gucke mir ein zufälliges Bit an. Wie groß wäre das es gleich 0 ist

Wie ich hier (#11) rechne ist falsch. Aber ~~viel~~ es ist ähnlich wie in Approximation ~~der~~ Poisson-Verteilung. Es ist nur „ein bisschen falsch", aber nicht „schlimm falsch"

bestimmte Anzahl 1-Bits.

Anteil 1-Bits: $\approx \frac{kn}{m}$

Anteil 0-Bits: $\approx 1 - \frac{kn}{m}$

kn klein gegenüber m ist $1 - \frac{kn}{m}$ eher richtig?

wenn kn größer wird: $1 - \frac{kn}{m} + \frac{(\frac{kn}{m})^2}{2!} \pm \ldots$

$$\approx \exp\left(-\frac{kn}{m}\right)$$

Auf Folie ist nur Näherungsweise. So stimmen würde es dann, wenn alles unabhängig ist.
↳ in Praxis unwahrscheinlich

Approximation ähnlich wie letztes Mal mit Log.

Warum ~~gibt~~ gibt es ein k das minimiert? Warum wird es nicht immer besser, je mehr Hash-Fkt. ich nehme?
→ es gibt eine feste Größe und dann würde es mehr 1en geben

Bloom-Filter sind eine schöne Sache, aber man kann z.B. nicht zählen.

Aufgabe: Paper "Lighter" lesen → muss nicht ganz gelesen werden, aber „Trick finden" wie man k-mere trennen kann ohne echt zu zählen!

Ausblick Übung:
- Aktueller Stand Hashing
- Bloom-Filter

27.04.17

co-ü

Paketmanager Conda

Ki Arbeitsgruppe Paketmanager Conda
hier: Miniconda
https://conda.io/miniconda.html

```
conda create --name omics python=3 jupyter numpy scipy matplotlib seaborn pandas h5py numba
```

(Umgebungsname)

managt Paketabhängigkeiten
Mit Conda kann ich beliebig viele von solchen Umgebungen anlegen mit speziellen Konstellationen
⇒ beeinflusst nicht normales Systempython!

Wie verwende ich das?

```
source activate omics
(bis source deactivate (ggf o Pfade ändern))
jupyter notebook
> Python 3 auswählen
```

(zeigt Wert letzten Ausdrucks mit Shift + Return)

▶ Bloomfilter:
Fehlerrate ist e in Abhängigkeit von k m und n und ich kann nach k optimieren
10^{-6} ist kleine Falsch-Positiv-Rate

Naiv überlegt:

$|X| = 4^{71}$, davon kommen 4^{16} vor

4^{16} versch. Objekte repräsentieren

$\Rightarrow$ wir benötigen Vielfaches von 4^{16}

$\rightsquigarrow$ für solche Spielereien Notebooks benützen.

Da kann ich einfach HTML-Seite machen und verschicken.

[Notebook s. Elias]

$\Rightarrow$ x, y auf log. Skala $\Rightarrow$ eine Gerade

ganz am Anfang gegen 1, aber ab da eine Gerade

Bsp: mit 10^{-9} Fehlerrate $\rightarrow x = 40$ $\Rightarrow$ 5 Byte
$\Rightarrow$ 40 GB Platz f. Bloom-Filter

Plotten:

```
%matplotlib inline   <- „magic" komando f. Notebook
plt.plot(x-Achse, y-Achse)
plt.yscale('log')
plt.show()
```

▷ Fehlerrate von 10^{-6} oder 10^{-9} klingt erst mal gut

4^{16} Schlüssel

$(4^{71} - 4^{16}) \cdot 10^{-9}$

$= 4^{71} \cdot 4^{-15}$ $(\approx 4^{-15})$

$= 4^{56}$

$\Rightarrow$ Man bekommt die richtigen Antworten und einen kleinen Teil des Restes falsche Antworten.
Da aber der Rest so groß ist, bekommt man entsprechend viele falsche Antworten (hier 4^{56})

schon bei 3 Ebenen relativ viel es sehr

Bloomfilter
Ausnahmemenge
Bloomfilter
Ausnahmemenge
etc.

Alles außerhalb davon kann ich ignorieren

⇒ sehr aufwändig / rechenintensiv!

Beim Erstellen gucke ich noch mal alle Mengen durch!

Wie kann ich den Bloomfilter exakt (exakter?) machen, ohne dass ich meine Fehlerrate so extrem niedrig zu setzen.

⇒ clevere Methode

Ausnahmemenge komplett zu speichern ist Quatsch. Die Ausnahmenmenge besser so speichern!

Dadurch kann man wieder mit kleineren Fehlerraten (z.B. 10^{-3}) arbeiten.

Aufg.: mit Bloomfilter rumspielen

Wie groß werden Ausnahmemengen?
Lohnt sich dann noch ein Bloomfilter?

Mit Bloomfilter kann ich mir bis 1 zählen. Wie kann ich das trotzdem lösen (k-mere von häufig und selten unterscheiden)?

Mehr Listen brauchen mehr Speicher. Geht es auch mit nur einem Bloom Filter?

⇒ Sampeln / Ausdünnen der Menge
↳ nur ein Teil nehmen (5% → bei [redacted] (=30))
damit Bloomfilter
⇒ Rest überprüfen und ausdünnen

⇒ Mit Formeln spielen

Auf Webseite wird Quellcode und weitere Fragen eingestellt.

CO-VL Lighter-Paper

→ da keiner gelesen hat → keine Besprechung

Genomassemblierung

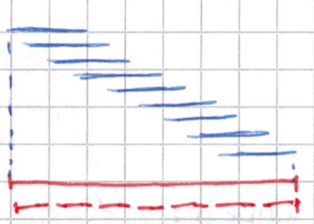
Geg.: viele DNA Reads (Länge 100–500 bp) (oder 1000)

Ges: Genom, aus dem die Reads stammen

(Genom: mehrere Chromosomen, linear oder zirkulär)

Bsp.: Mensch 1 2 ⋮ 22 X Y — E.coli ○

Idee: hohe Abdeckung (coverage), überlappende reads



bzw. bei Chrom.:

Zahlen bsp für Mensch

Reads sind in der Datei natürlich nicht sortiert!

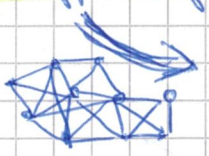
Viele reads sind lang und haben Fehler.

Heute kein Problem was Speicherplatz etc. angeht. Humangenom ca. 1998–2000 assembliert. Zwar waren die reads länger (s.o.) aber keine so große Rechner, Festplatten etc. Damals Riesen-Problem.

Ideensammlung:

Klassisch 90'er

Überlapp-Graph (Knoten: Reads, Kanten: weit überlappende Reads)



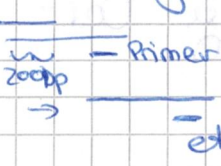
~100 Mio. Knoten

~ vielfaches an Kanten (>1000x)

100 Mrd Kanten

~ (100 Mio)² / 2 Überlapp-Tests (naiv)

Um Lücken zu füllen: Primer designen



⇒ früher wenig Sequenzen, die lange waren, darum geht das

"modern" 2000es

De Bruijn Graph (Knoten: k-mere, Kanten: $(k+1)$-mere) → solide → 5. Lichte-Paper

AGTTC GTTCT

AGTTCT

~3 Mrd. Knoten
Kanten implizit

in Vorlesung näher betrachten

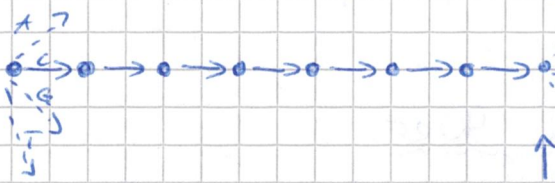
neue Ideen 2015+

Überlapp mit Hashing

Ähnlichkeitserhaltendes Hashing
(engl. locality sensitive hashing (LSH))

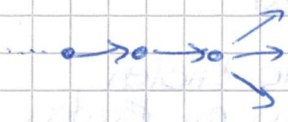
Datenstruktur für k-mere:

- sortierte Liste (binäre Suche)
- Hashtabelle (mehr Platz, viel Speicher, ggf. schneller Zugriff) ← wenn man es wirklich gut macht
- Bloom Filter (Zugriff da, wenig Speicher)
  + Ausnahmeliste (Bloom Filter...)
    ↳ es gibt Bloom Filter, die cachefreundlicher sind; im Lichte-Paper auch supplement
    Varianten: Blockweise Bloom Filter (schneller, aber größer)



Idealfall, es gibt nur einen Weg (die Vorl. das das Chromosom zu Ende ist „vergessen" wir jetzt einfach)

kein Nachfolger → Lücke in Sequenzdaten oder Chromosom zu Ende (lokal gesehen zu Ende)



vermutlich mehrfach im Genom mit versch. Nachfolgern

im 1. Schritt an allen Verzweigungspunkten aufhören

3 Schritte: → v. contigous (?) ≙ zusammenhängend

1. Contig-Assemblierung
2. Scaffolding (Verbinden von Contigs)
3. Gap filling

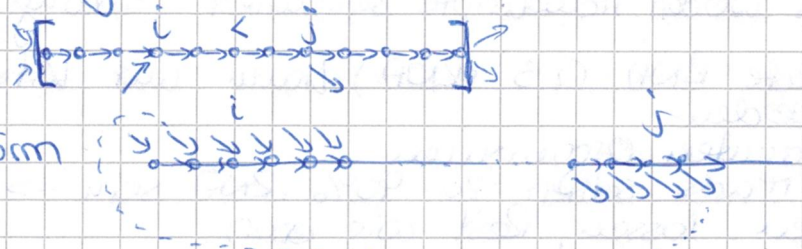
(1) Contig: Sequenz, die sicher im Genom ≥ 1× vorkommt

(Es gibt versch. Klassen von Contigs. Hier werden 2 erwähnt.)

? Im Moment betrachten wir Strings, nicht DNA (Komplementär, etc.)

a) Unitigs: maximale Pfade mit inneren Knoten mit Ein- und Ausgrad 1. (einer Kante)

b) Omnitigs

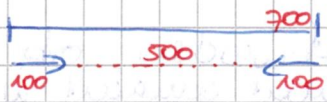


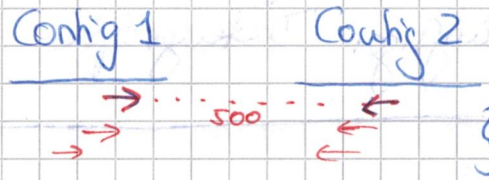
maximal lang, bis es $i \leq j$,

i eingehend verzweigt
j ausgehend verzweigt gibt,
mit Pfad $j \rightsquigarrow i$

Omnitigs sind länger als Unitigs

(2) Fragmente = Paare von Reads





} nicht nur ein Read-Paar, sondern ganz viele, die es konsistent halten

Contigs dürfen mehrfach verwendet werden!

⇒ Daraus sog. Scaffolds bauen

„Puzzlespiel" auf höherer Ebene, wobei die Lage-information genutzt wird.
Es können Lücken bleiben!

Ausgabe: Scaffolds (mit Glück ganze Chromosomen, aber eher kleiner)

Co-Ü

GitLab

→ https stickeren, ssh-Link klappt nicht
↑ man braucht ssh-Key

Email mit GitLab-Benützername schicken

Review-Artikel nicht frei zugänglich → v.a. Bibel
⇒ muss man nicht lesen; gibt einen guten Eindruck

Vorschlag für Projekt: RNA-Assemblierung

- nach dem Splicing meist 3-10 reife Teile (aus Exons)
- mature mRNA → cDNA → doppelsträngige DNA
  ⇒ Sequenzierung
  ↳ es werden Transkripte sequenziert (abhängig v. Anzahl)
- unerwünschte RNA (z.B. rRNA) kann mit Kits reduziert gefiltert werden.
- bei unbekannten Organismen
  → solide Reads sollten zu 90% rRNA sein → damit Kit bauen lassen, Rest wie eben

[~60.000 Gene ges.
25.000 davon Protein kodierend → führt zu mehreren Proteinen

Alles RNA Omnitigs machen. (Kann man mit „Hausmittelchen" machen. Man braucht keine fancy Datenstrukturen.

→ Datensatz v. Pilz o. Weißbüschelaffen (rel. groß) können zur Verfügung gestellt werden.

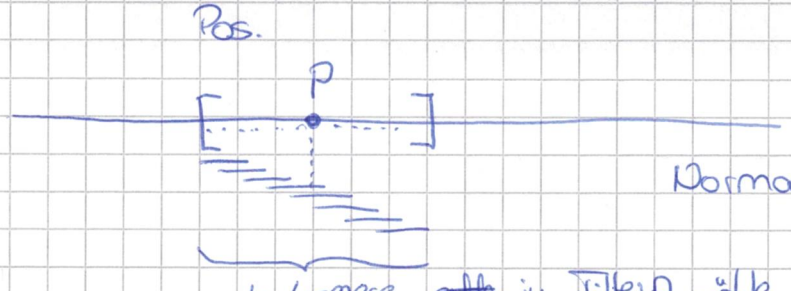
Jedes Contig müsste untersucht werden, ob ähnlich zu bekannten. (Müsste hier nicht gemacht werden.)

⇒ Sinnvolles Projekt mit vernünftiger Datengröße

Giguter-Paper

Schritt 1: k-mere zählen

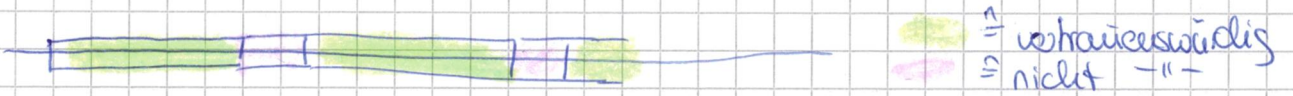
Schritt 2: Positionen im Read zählen



Normalfall: wenn in Umgebung alles i.O. ist, sollten alle k-mere im Bloomfilter A sein

k k-mere ~~alle~~ davon t in Filter A, wähle z.B. $t = \lceil \frac{k}{2} \rceil$

→ p „vertrauenswürdig", sonst „nicht vertrauenswürdig"



Nimm wie k-mere, bei denen alle Positionen vertrauenswürdig sind.

=> Filter B (oder hier rauskommt) ist ziemlich gut
Es ist möglich das in Filter B k-mere sind, die nicht in A sind.

Auch wenn ich zähle, kann es einmal sein. Wenn ich sample, dann brauche ich das.

Schritt 1 ist I/O-dominiert. Rechenaufwand im vgl. zum I/O sehr wenig. Darüber Parallelisieren wird es nicht (wirklich) schneller.

Schritt 2 hier lohnt es sich zu parallelisieren
Problem hier: schreiben des Bloom-filters B. Hier muss synchronisiert werden.

Auf jeden Fall kann man Positionsklassifizierung parallelisieren.

Es wird kein normales Bloomfilter, sondern Pattern-blocked Bloom filter genommen (Supplementary Note 1)
- Bitmuster setzen

Projekt-Idee (?): vllt. andere Bloom-Filter ausprobieren und vergleichen
→ analysieren, Eigenschaften nachweisen, zeigen das es in Praxis geht

H Hausaufgabe: Tomescu-Folien oder -Paper angucken
→ Omnitigs → auf jeden Fall
Review-Paper optional
↳ eher nicht formal (eher für Biologen) (Text)
=> kann man schnell lesen

=> Log in aus Sven schicken

CO- VL Ü

11.05.17

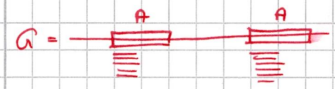
Genomassemblierung

(Vom biologischen zum formalen Problem)

1. Gegeben (Eingabe): Strings über $\{A,C,G,T\} =: \Sigma$
   - $\{s_1,..,s_n\}$
   - IUPAC-Alphabet (15 Zeichen) $:= \Gamma$
   - $\Sigma \times \{0,..,40\}$, $\Gamma \times \{0,..,40\}$ } Strings mit Qualitätswerten

   Man könnte noch was über Anzahl und Länge sagen

2. a Gesucht (Ausgabe): String G über $\{A,C,G,T\}$, so dass gilt: s_i (oder rev. comp(s_i)) Teilstring von G $\forall i = 1,..,n$ und G ist minimal lang mit dieser Eigenschaft

   "shortest common superstring"

   im Wesentlichen
   → i.W. Genome ohne lange Wdhl
   → Genom enthält auch fehlerhafte Reads

   So war das lange formuliert!

▷ NEU:

1. Gegeben (Eingabe): a) Strings über $\{A,C,G,T\} =: \Sigma$,
   Parameter $k \in \mathbb{N}$
   a) enthält noch etwas mehr Kontext
   b) Menge von k-meren über $\{A,C,G,T\}$
   ["solide k-mere"]

2a Gesucht (Ausgabe): De Brüijn Graph (V,E), V: k-mere aus $\{s_i\}$, E: $(k+1)$-mere aus $\{s_i\}$
   ? Pfad "G", der jede Kante ≥ 1 mal enthält
   ? Pfadmenge Π, so dass jede Kante in ≥ 1 Pfad enthalten
   / minimaler Kardinalität

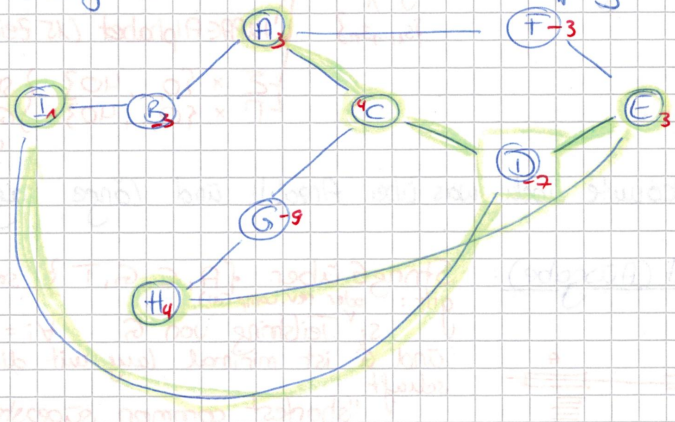
Es gibt mehrere (mind. 20) sinnvolle Formulierungen für Genomassemblierung
→ lernen sowas formulieren! (Idealerweise hat Formulierung einen) polynomielle Laufzeitalgo, bzw. Approximation

Protein-Interaktionsnetzwerk

Knoten: Proteine

Kanten: mögliche physikalische Interaktion

Knotengewichte $\in \mathbb{Z}$ * >0: auffällig, <0: unauffällig $\gamma = (\gamma_v)$



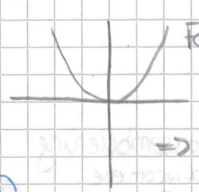
* Wo dieses Protein in einer Klasse von kranken Patienten auffällig im Vgl. zu einer Klasse von gesunden Patienten

Gesucht:

Relevante Netzwerkmodule

- Steinerbaum zwischen allen auffälligen Knoten
  ↳ wenigsten Kanten — NP-schwer
- maximum scoring connected subgraph (+ alle pos. Knoten enthalten) — NP-schwer
  ↳ relativ populär geworden
- Thresholdgraph (>0) connected components $\geq m$
- $F(x) = \sum_{v \in V} (x_v - \gamma_v)^2 + \lambda \sum_{e=\{v,w\} \in E} |x_v - x_w|$ (benachbarte Knoten)

  → lokal konstante Lösung x

  => x_v und γ_v versucht nah bei einander zu sein (Value Null)

  konvexes Optimierungsproblem in x (ähnlich gut verstanden wie lineares Problem nur schwieriger)

  => immer polynomiell lösbar

CO-V4

18.05.2017

Metagenomik

Analyse / Assemblierung mehrerer Genome in einer Probe

- z.B. Haut-Bakterien
- Stuhlproben-Bakterien (Darmflora)
- Umweltproben - z.B. Gewässer-Organismen

alle Referenz-genome bekannt

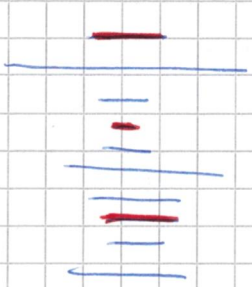
Referenz-Genome existieren nicht

Für jeden Read:
- finde Spezies
- finde Genfamilie

Gleichzeitige Assemblierung aller Genome
Zuordnung Read ↦ Spezies nicht bekannt

1. Assembliere Contigs (z.B. unitigs) wie bei der einfachen Genom-Assemblierung
(Contigs ≙ 1000 - 10.000 bp)

Metagenomikspez. Problem → 2. Gruppierung Contigs → Genome



die roten gehören zu einer Spezies!
das ist aber nicht bekannt

→ reicht nicht aus!

Contigs → Genome

▷ Eigenschaften der Contigs:

a) Tetranukleotid-Häufigkeit

Gleiche Organismen → gleiches Tetranukleotid-vektor

Anzahl 4-mere: $4^4 = 256$
$4^2 = 16$ selbstkomp.

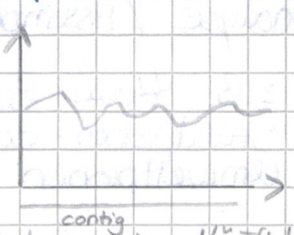
$$\frac{256-16}{2} + 16 = 136$$

Contigs → TR^{136}

$C \longmapsto x = (x_1, \ldots, x_{136})$ mit $x_i \geq 0 \quad \forall i$

$\sum x_i = 1$

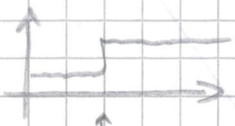
b) Durchschnittliche Abdeckung alle k-mere, aus denen das Contig assembliert würde



Abdeckung der Häufigkeit der k-mere in einem Contig

Idee:

Contigs aus dem gleichen Genom haben approximativ
a) gleiche Tetranukleotid-Häufigkeit
b) gleiche Abdeckung ("coverage")



würde man hier trennen, da man vermutet, dass es 2 versch. Organismen sind

Contig $\hat{=}$ Punkt in $\mathbb{R}^{136+1}$ → haben eine andere Dimension als → deswegen steht da auch nicht 137!

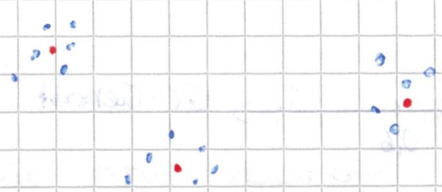
▷ Clustering-Verfahren:

- Lloyd's Algorithmus ("k-means")
- DBscan (Dichte-Basiert) (density based)

... EM-basierte Verfahren (probabilistisches Clustering, expectation maximization)

- k-means

Punkte in $\mathbb{R}^d$, $x_1, \ldots, x_n$



① Finde k Clusterzentren $c_1, \ldots, c_k$

② Zuordnung $x_i \to c_{j(i)}$, so dass Summe der Distanzen $d(x_i, c_{j(i)})$ minimal ist

2 Probleme: ① Zentren
② Zuordnung

np-schweres Problem (beide Probleme gleichzeitig) → Einzelproblem gegeben das anderen ist einfach / trivial
Zentren gegeben → nächster Nachbar
Zuordnung gegeben → in der Mitte

Lloyd's Algorithmus ist „schlau" und löst die Probleme immer abwechselnd. Der garantiert aber nicht das globale Optimum zu finden. Es kann sein, dass er in seinem lokalen Optimum hängen bleibt.
Man muss initiale Werte vorgeben.
Wie macht man das? Man macht das ganze 100x mit versch. initialen Werten und nimmt das beste Ergebnis.
Lloyd's Algorithmus ist für die Praxis wirklich gut und löst das Problem in polynomieller Zeit, er löst aber nicht das k-means-Problem da er nicht garantieren kann das globale Optimum zu finden.

- **Allgemeiner Ansatz für probabilistisches EM-Clustering**
  - Struktur wie bei k-means:
    Initiale Zentren
    → probabilistische Zuordnung der Punkte
    → neue Zentren

spezifisch für unser Modell

$$P(x_i \mid c_j) = P_{4mer}(x_i \mid c_j) \cdot P_{cov}(x_i \mid c_j)$$

$P_I(\dots)$ $P_{II}(\dots)$

mit d-dimensionalen Normalvert. $= c \cdot \exp\left(-\frac{\|x_i^I - c_j^I\|_2^2}{2\sigma_*^2}\right) \cdot \text{Poisson}_{c_j}^{II}(x_i^{II})$

Konstante, da wir grad nicht weiß, was der richtige Normalisierungsfaktor für d-dimensionales Clustering

mit $x_i = (\underbrace{x_{i1}, \dots, x_{i136}}_{x_i^I} \mid \underbrace{x_{i137}}_{x_i^{II}})$

→ Poissonvert. für Coverage

Berechne $\log P(x_i \mid c_j)$ für alle c_j für alle x_i
Berechne Zugehörigkeiten:

$$w_{ij} = \frac{P(x_i \mid c_j)}{\sum_j P(x_i \mid c_j)} \rightarrow \sum_j w_{ij} = 1 \text{ für alle } i.$$

Es gibt mir für alle i die (relative) Verteilung der Zentren zu

Man kann den Alg. auch begreifen als (offensichtliche) Verallgemeinerung des Lloyd's Algos.

* man könnte auch σ_j^2 nehmen, aber das macht es auch schwerer.

⇒ Wenn man das alles so macht, dann funktioniert Metagenomik relativ gut.
v.A. auf simulierten Daten (da kann man es überprüfen)

Metagenomik

alle Referenzgenome bekannt

xenografts

Spezialfall

Mensch. Tumor in Maus

Referenz-Genome unbekannt

→ contigs
→ Eigenschaften
→ Clustering

Sequenzierung: Mischung von Reads aus Mensch + Maus

Einteilung:
1) Mensch
2) Maus
3) beides gleich gut
4) eins von beiden (wenige)
5) widersprüchlich (teilw. 1, teilw. 2)

} schnell: Hashing, z.B. Cuckoo hashing

z.B. Mensch hat ca. 1,4 Mrd 25-mere in nicht repetitiven Bereichen.

Wie viele Bits sind minimal nötig, um Mengen der Größe 1,4 Mrd von 25-meren darzustellen?

Naiv: 25-mere $\hat{=}$ 50 Bits
→ 1,4 Mrd · 50 Bits

Es gibt $\binom{4^{25}}{1{,}4 \cdot 10^9}$ verschiedene mögliche Mengen.

Braucht $\geq \log_2 \binom{4^{25}}{1{,}4 \cdot 10^9}$ Bits. $\log_2 \binom{n}{k} = \log \frac{n!}{k!(n-k)!}$

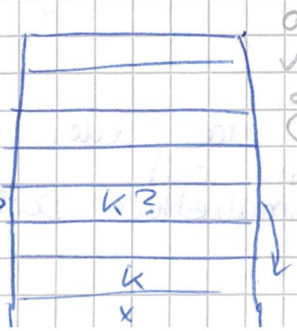
$= \log n! - \log k! - \log[(n-k)!]$

$\log 1 + \log 2 + \ldots + \log n \leq n \log(n)$

Hash:

Schlüssel k 50 Bits (25-mer)

$h(k)$



alle Schlüssel müssen vorkommen und es gibt leere Felder

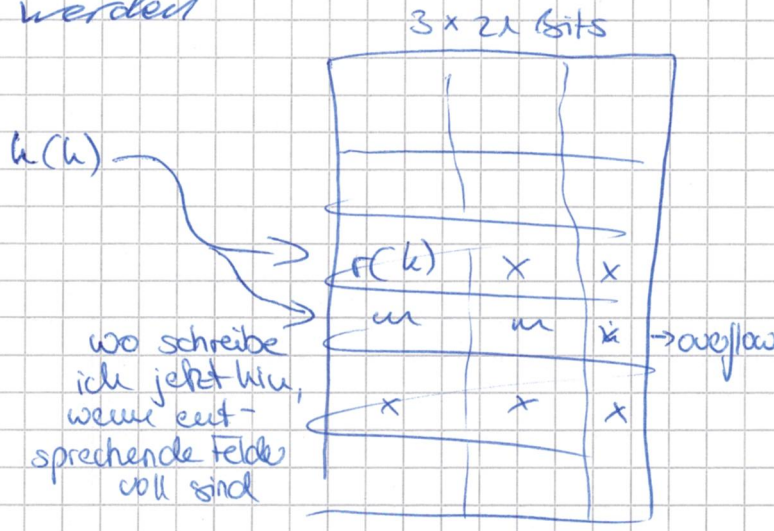
$k \mapsto (h(k), r(k))$

Adresse/ Seite i.d. Hashtabelle — "Rest" fingerprint

so dass $\exists f$ gibt mit $f(h(k), r(k)) = k$

beispielsweise Zerlegung: $h(k)$: erste 25 Bits
$r(k)$: restl. 27 Bits

→ gemeinsame Anfänge müssen nur einmal gespeichert werden



Das was überläuft sammel ich (wenn gut, dann 5% der alten Menge)

⇒ 2. Tabelle, ggf. 3. Ebene

⇒ breitere Tabelle:
wenige Überlauf, mehr verschwendeter Platz
⇒ schmalere Tabelle:
genau umgekehrt
→ passende Einstellung finden

Variantendetektion → in 2 Wochen

Hauptproblem:
Techn. Probleme von Varianten unterscheiden
⇒ statistisches Thema

danach: RNA-Sequenzierung (quantitativ) ⇒ statistisches Thema

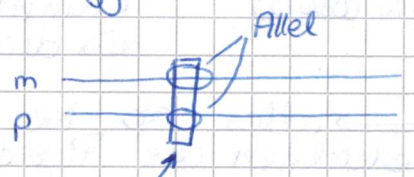
01.06.'17

CO-VL von Johannes:

DETECTING GENOMIC VARIANTS

(Folien von 2013)

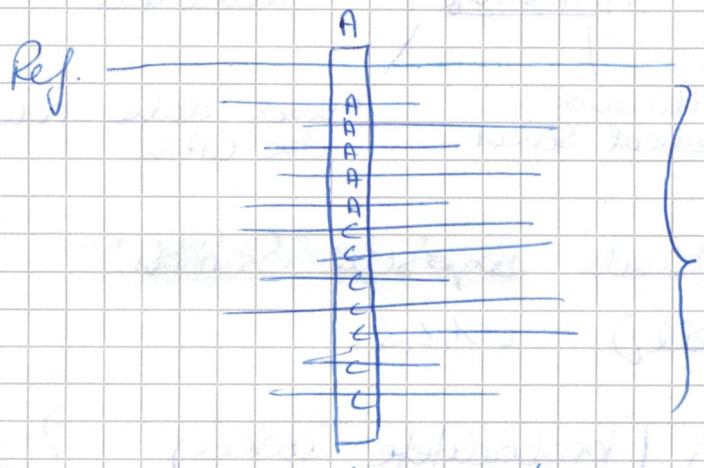
- DNA Referenzgenom: Durchschnitt aus zufällig ausgewählten Personen
- SNV ≙ single nucleotide variant
- Indel ≙ insertion or deletion
- SNP ≙ single nucleotide polymorphism (in a population)
  ⇒ muss beobachtbar sein, also nicht im Einzelfall
- Variante: homozygot: gleiche Allele (≠ Ref.)
  heterozygot: verschiedene Allele (= Ref, ≠ Ref.)



m ≙ maternal
p ≙ paternal

homozygot (aber keine Variante):	beide Allele = Ref	0
heterozygote Variante	1 Allel = Ref, 1 Allel ≠ Ref	1
homozygote Variante	beide Allele ≠ Ref (aber gleich)	2
		↑ Kodierung

Genotyp Bsp.: 0 0 0 1 1 1 2 2 0 0 ...



↑ zu erwarten: max. 2 verschiedene Basen

Probleme: - korrekt mappen
- PCR-Dublikate:

} nicht unabhängig! eigtl. 1 Read, 4 Kopien

Wann PCR-Dublikat, wann zufällig gleich lang

- Fehlerrate heute eher <1%, dennoch vorhanden

A und O: tief genug sequenzieren!
→ teuer (Geld nahezu proportional zur Sequenziertiefe)
→ zum sparen nicht alles sequenzieren, sondern Exom (bzw. etwas drum herum)
(s. Bild #7) Coverage Histogramm

#9 GATK = genome analysis tool kit
* aus typischer Exom-Analyse ~50.000 Varianten

Qualitätswerte in Reads:

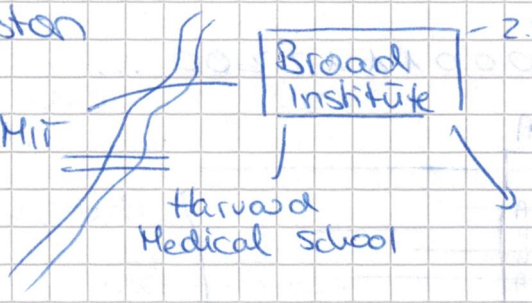
Q	10 .. 20 ..	30	(PHRED-Skala)
Fehler-wkt	10^{-1} .. 10^{-2} ..	10^{-3}	$10^{-Q/10}$

98% alle bp

Qualitätswerte kommen aus dem Sequenziergerät

→ alles ab hier ist zuverlässig (erleichtert die Prozedur, da man nicht mehr die Qualitätswerte beachten muss)

▷ GATK (ab # 18?)
kommt aus Boston



- z.B. Genomanalysen mit (informatik-technischen)
- und daher kommt das GATK

Was die machen? Im Grunde Bayes'sche Statistik!
leicht: $P(\text{Daten} \mid \text{Modell}_i)$ $i=1,2,3$
(für jedes Modell)

Wir wollen aber: $P(\text{Modell}_i \mid \text{beobachtete Daten})$

$$= \frac{P(\text{Daten} \mid \text{Modell}_i) \cdot P(\text{Modell}_i)}{P(\text{Daten})}$$

} Satz von Bayes

$P(\text{Daten})$ Konstante: durch 1 ersetzen

dann statt = $\propto$ (proportional)

=> wir wollen ja nur das Modell bestimmen

#19

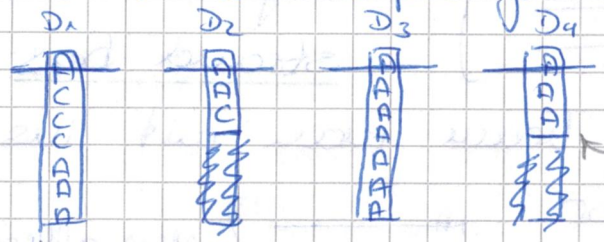
- mehrere Patienten i
- alliierte Liste von Basen D_i z.B. ACCTA...
- W'lkt in Liste i die j. Base X ist
- Tabelle ist empirisch ermittelt
  „Wenn ein C steht und ich einen Fehler mache, wie Wahrscheinlich ist dann ein A, T oder G?"
  ⇒ Sven findet das „nicht so wahnsinnig wichtig"

#20

- Formel: ½ W'lkt Mutter-Allel + ½ W'lkt Vater-Allel

#21

- GATK macht das über alle Samples gleichzeitig



(für sich alleine gesehen, kann man nichts aussagen, da es heterozygot sein kann oder ein Fehler
Wenn man aber mehrere Samples hat, dann kann man vergleichen, ob der Fall auch an anderer Stelle vorkommt (hier z.B. Sample D_1). Dann kann man das Gebiet mit größer W'lkt entscheiden

mehrere Samples gleichzeitig

#22

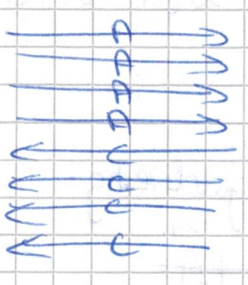
- alle Samples übereinander legen und sagen, ob es überhaupt eine Variante gibt
- für Gesamtwkt muss man über alle gehen
- sehr viele Möglichkeiten, aber man kann es einschränken
- für jedes n eine vernünftige apriori-W'lkt, die sich aus evolutionstechnischen Modellen zusammensetzt

#23

- n Samples
- ~~der Realität entsprechende~~
- $P(\text{Modell}_i)$ aus evolutionstechnischen Modellen berechnen
- Wieviele Chromosomen haben die Alternative?

#27

- Wie stellt man fest, ob man Unsinn macht? Es gibt nicht nur Basenqualität, sondern auch andere (s. Folien)
-

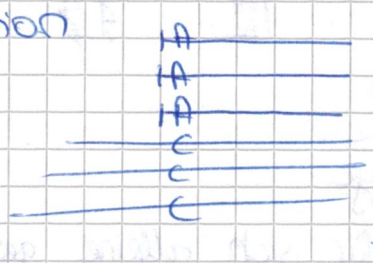


Variante hängt von der Richtung des Reads ab → dass nicht sein!

strand bias

→ sowas kann man mit ins Modell einbringen

- Read-Position



ein alignment-Fehler

Nicht zu viele Gedanken zu den Formeln. Man muss Denkweise verstehen!!!

i.d.R. Vorwärtsdenkender Prozess (leichte Teil)

andere ist Expertenwissen, ...

Test auf Krankheit X (lebensbedrohlich, selten) 10^{-6}: Inzidenz | CO-Ü

wenn Patient krank, Test positiv mit 99,8%
-"- -"- gesund, -"- -"- -"- 2%

Daten: Test +
Test −

Modell: krank
gesund

gegeben

Gesucht: $P(\text{krank}|+) = \frac{P(+|\text{krank}) \cdot P(\text{krank})}{P(\text{pos})}$

Normalisierungs-Konstante ↑ nicht gegeben

$$P(\text{krank}|+) \propto P(+|\text{krank}) \cdot P(\text{krank})$$
$$P(\text{gesund}|+) \propto P(+|\text{gesund}) \cdot P(\text{gesund})$$

$$P(\text{krank}|+) = \frac{P(+|\text{krank}) \cdot P(\text{krank})}{\sum_{m=\{\text{gesund},\text{krank}\}} P(+|m) \cdot P(m)} \quad (= P(+))$$

$$P(\text{krank}|+) \propto 0{,}999 \cdot 10^{-6} \approx 10^{-6}$$
$$P(\text{gesund}|+) \propto 0{,}02 \cdot (1 - 10^{-6}) \approx 0{,}02$$

$$\frac{10^{-6}}{10^{-6}+0{,}02} \approx 0 \qquad \frac{0{,}02}{10^{-6}+0{,}02} \approx 1$$

dieses Verhältnis bestimmt Ergebnis

also $P(\text{krank}|+) \approx 0$
$P(\text{gesund}|+) \approx 1$

Selbst bei positivem Test, bin ich ziemlich sicher gesund!

1 Mrd.	krank	gesund
+	999	20 Mio
−	1	980 Mio
	1000	1 Mrd

absolute Zahlen am Bsp. von 1 Mrd. Menschen

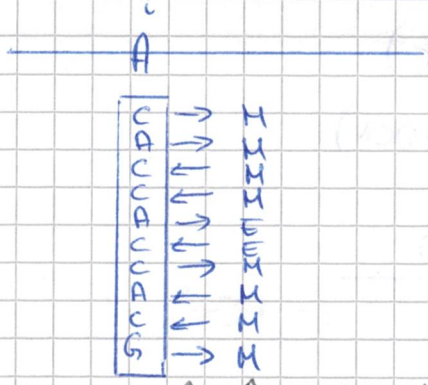
Bayes schaut in diese → Richtung

=> es lohnt sich dem Schema (Bayes) zu folgen, da sehr kontraintuitiv!

- Zufälliger Mensch:
  Was ist wahrscheinlicher? Informatiker oder Informatiker, der Linux benutzt?

$$P(A \cap B) \leq P(A)$$

Buchempfehlung: Thinking Fast and Slow



Beispiel für Daten

Genotypen:

AA		(beide Allele stimmen mit der Referenz überein)
AC, AG, AT	}	heterozygote Varianten
CC, GG, TT	}	homozygote Varianten
CG, CT, GT	}	implausibel

da man nicht sagen kann welches Allel welches ist

Genotyp (x,y) $\binom{4}{2} = 10$ Genotypen

Daten: Liste von Tripeln (c,d,e)

c : character (eigtl. Nukleotid)
d : direction
e : end

ε: Sequenzierfehler W'lkt ($1/1000$)

$$P((C,D,E) = (c,d,e) \mid (X,Y) = (x,y))$$

$=$

	AA	AC	AG	AT	CC	...
A	$1-\varepsilon$	$\frac{1-\varepsilon}{2} + \frac{1}{2}\cdot\frac{\varepsilon}{3}$	...	...	$\varepsilon/3$	
C	$\varepsilon/3$					
G	$\varepsilon/3$					
T	$\varepsilon/3$					
	$\Sigma = 1$					

eigentlich $\varepsilon/3 \cdot \frac{1}{2} \cdot 2$ (weder gut noch T) ↑ 2 Allele

$\frac{1-\varepsilon}{2}$: W'lkt, dass ich vom A-Allel ein A lese

$\frac{1}{2}\cdot\frac{\varepsilon}{3}$: W'lkt, dass ich vom C-Allel ein (falsches) A lese

Ich kann jetzt Modelle bauen wie z.B. wenn da ein C steht, geht der Read mit hoher W'lt nach rechts. Eigentlich müsste das unabhängig sein. Wenn aber so ein Modell gewinnt, dann weiß ich, dass da etwas nicht stimmt!

$P_1\,(C=c \mid (X,Y)=(x,y))$ ⇝ unsere Tabelle

$P_2\,(D=d \mid C=c, Z=z)$

erweitern: $P((C,D,E)=(c,d,e) \mid (X,Y,Z)=(x,y,z))$

Z ist Bayes??? Richtung-Bayes?

Z: -1: eher ← bei kleinem c
0: 0,5
1: eher → bei kleinem c

: usw.

Ich habe viele verschiedene Modelle und für alle kann ich in der Vorwärtsrichtung meine Daten auswerten. Und dann kommt Bayes und bewertet meine Modelle. Und da ist dann das Z? Wenn Richtung-Bayes mehr bringt als ohne, dann stimmt da etwas nicht!

⇝ Diese Modelle kann man beliebig kompliziert machen!
⇒ großer Rechenaufwand
Bei der Parallelisierung ist es sehr gut. (Trivial Parallelisierbar) Man kann z.B. pro Kern eine Stelle gleichzeitig betrachten.

HA Folien im Git
Modelle noch mal angucken!

08.06.17

vom letzten Mal zusammenfassung

Varianten-Detektion

Modell: P(Beobachtung | Genotyp)

a-Priori P(Genotyp) (Ref. am wahrscheinlichsten)

Inferenz mit Bayes: P(Genotyp | Beobachtung)

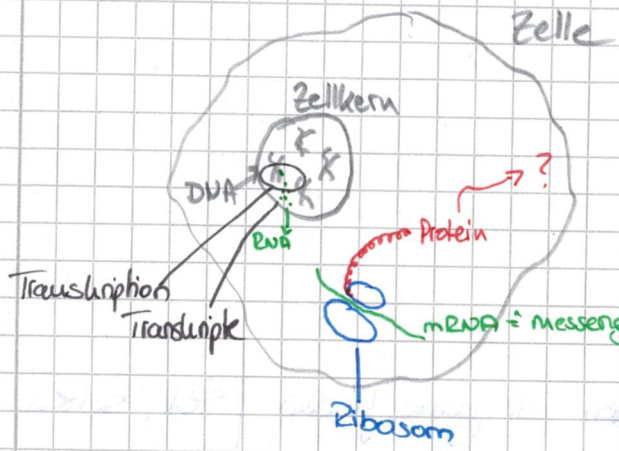
Probleme: sehr komplexe Komponenten
sehr große Zustandsräume

→ keine exakten Rechnungen → Approximationen
Markov Chain Monte Carlo (MCMC)

⇒ dient i.d.R. hier etwas auszurechnen indem die W'ten in einer Simulation berechnet werden, da man sie nicht mehr exakt ausrechnen kann

---

▷ Transkript-Expression



RNA kann Zellkern verlassen

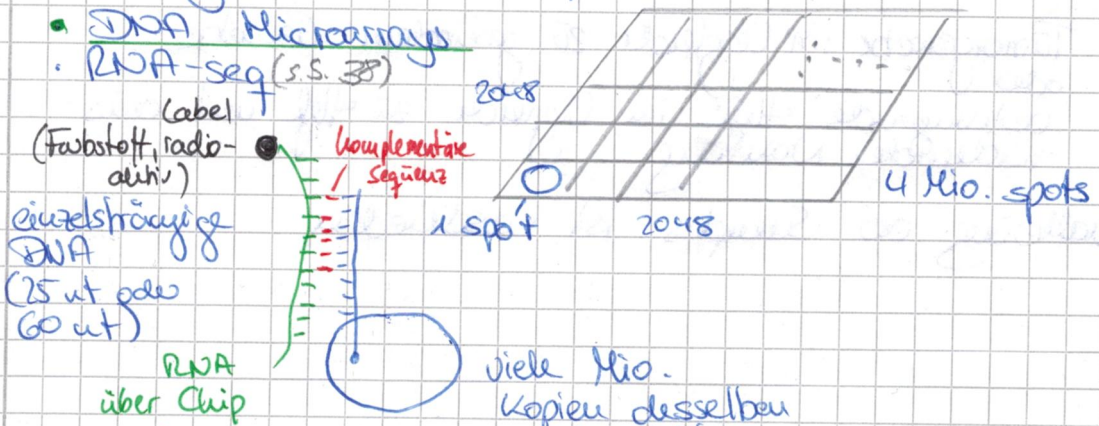
~ 20.000 – 25.000 Protein-codierende Gene

~65000 Gene

Man weiss wie häufig jedes Gen abgelesen wird (transkribiert wird) und bestimmt so dann in welchem Zustand die Zelle ist.

Technologien: (Transkripte zählen, Gene zählen)

- DNA Microarrays
- RNA-seq (s.S. 38)



Lukrativ des Verfahrens entspricht der Menge der Transkripte, also starkes Leuchten: es gibt viel was Hoben bleiben kann, etc.

Firma: Affymetrix 25-mere
Agilent 60-mere
(50-70-mere) } spezifischer, aber man braucht mehr DNA, es kann sich mehr verhaken

~60 €

⇒ sie können nicht die volle Bandbreite abfangen (ggf. Spot übersättigt) bzw. sind selbst übersättigt

⇒ vor 15 J. interessantes Bioinfo-Problem: Designen des Arrays (eindeutige DNA-Sequenzen, die gleiche thermodyn. Eigenschaften haben (Gc-Gehalt).

- RNA-seq

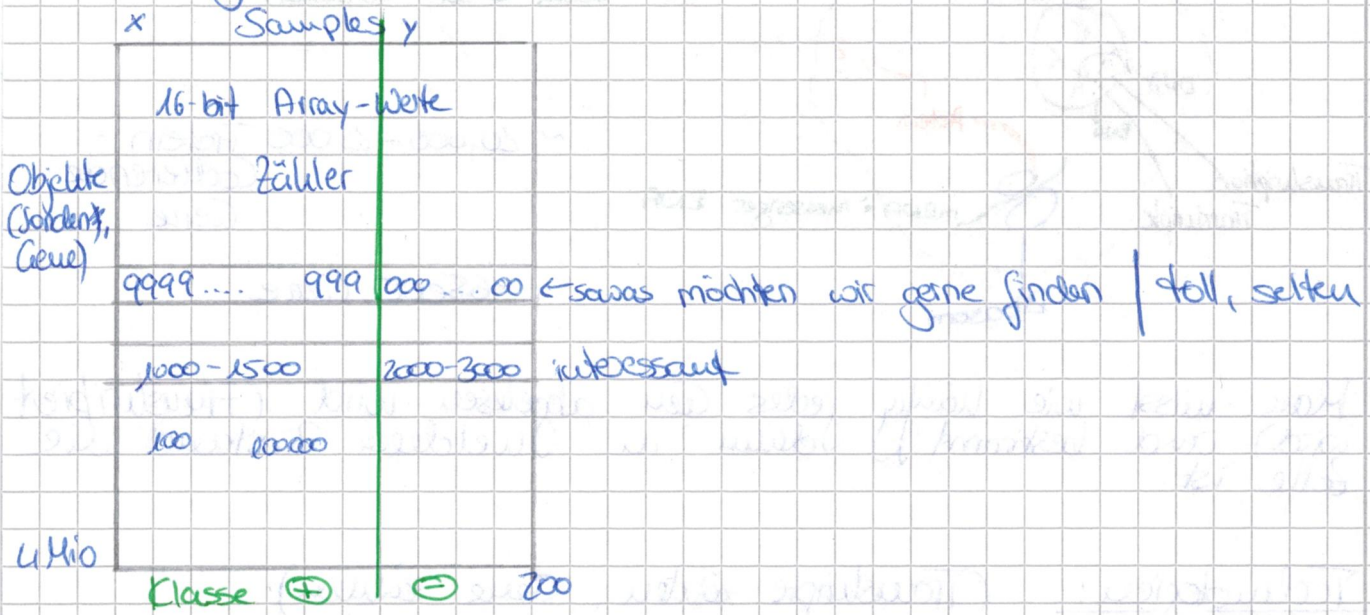
sequenziere alle RNA-Moleküle (Umschreiben in cDNA)

Gen zuordnen

Gene zählen

⇒ Beides hat ähnliche Probleme

Ausgabe: Matrix



↳ z.B. Tumorgewebe im Vergleich zu gesundem Gewebe
oder
verkümmerte Hefe im Vergleich zu Hefe mit ausreichender Nahrung

Problem: Skalierung der Samples ist verschieden

Normalisierung:

- "house keeping genes", relativ unrobust (Gene mit stets konstanter Expression)
- Division durch Summe / Mittelwert (je Sample Spalte)
- Division durch Median
  ↳ ignoriert Ausreißer
- ... " — gekürzter Mittelwert: *

a	1,5 a
b	1,5 b
c	⋮
d	⋮
⋮	
x	
y	
z	10k·z
Σ 1	2

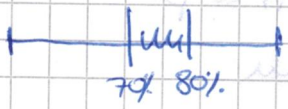
Ges. X,
sortiere X, $x_{sorted} = (x_{(1)}, x_{(2)}, \dots x_{(n)})$

$$\mu_{(k)} = \frac{1}{n-2k} \sum_{i=k+1}^{n-k} x_{(i)}$$

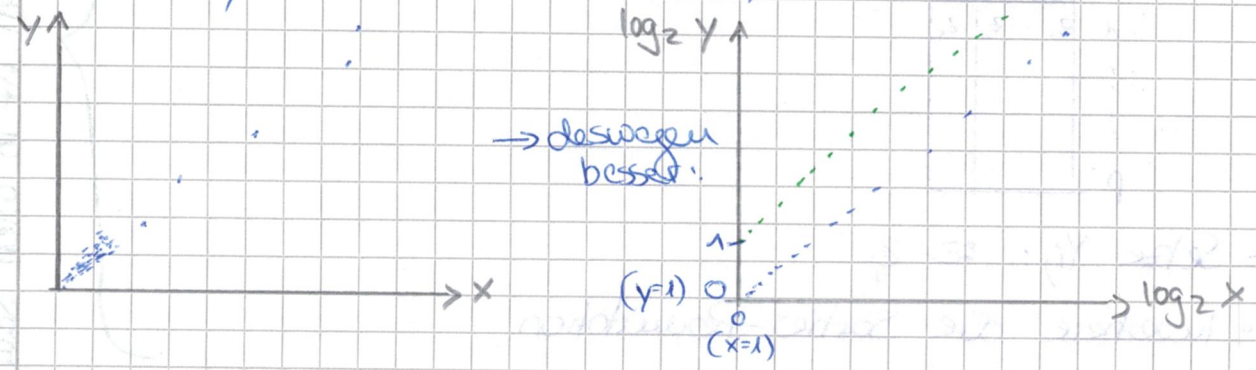
Median für $k \approx n/2$

Mittelwert für $k = 0$

* muss nicht symmetrisch sein, sondern man kann von 70-80% gehen, da sehr viele Gene gar nicht exprimiert werden und damit bei 0 liegen. (ca. 50%). Damit hat man die Gefahr, dass 0 als Median 0 herauskommt.



x und y s. S. 38 in der Matrix



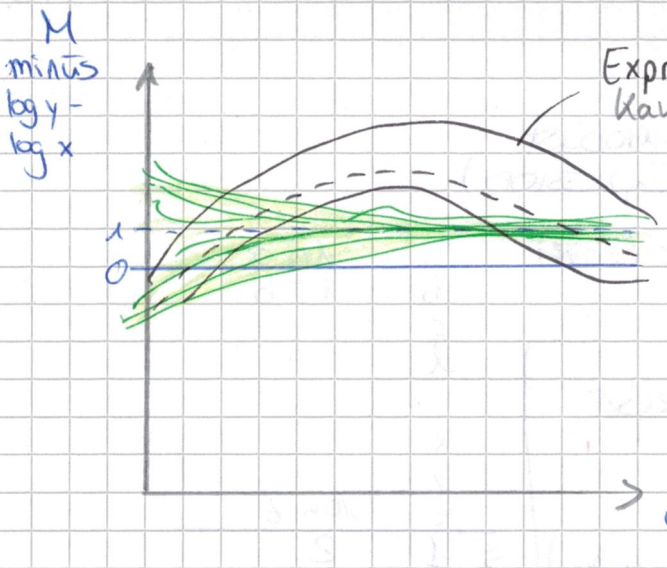
in der Praxis natürlich nicht so ideal

$y = 2x$

$\log y = \log_2 2 + \log x$
$= 1 + \log x$

grüne Kurve ⇒ macht einen Offset von 1

⇒ jeder konstante Faktor macht einen Offset



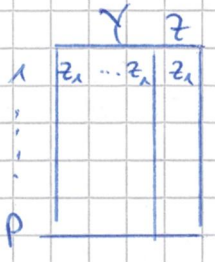
MA-Plots sind diagnostisches Tool, um 2 Spalten zu vergleichen.
Damit kann man sehen, ob die Daten in Ordnung sind.
=> Je gerader, desto besser zu vergleichen.

harte Normalisierungsmethode:

Geg.: Matrix X p Objekte × n Samples $n \ll p$

- Sortiere jede Spalte für sich
  → Y
  Jede Zeile in Y ist ein Quantil der Datensätze in X.
- Berechne durchschnittliche Quantile

$$z_i = \frac{1}{n} \sum_{j=1}^{n} Y_{ij}, \quad i = 1 \dots p$$



- Setze $Y_{ij} := z_i$
- Invertiere die Sortier-Permutation

Quantilnormalisierung
→ das i-te Quantil ist in allen Samples gleich $\forall k$

=> härteste, sinnvolle Normalisierungsmethode (die man noch vertreten kann)

-> $\frac{7}{p}$-Quantil = der 7-kleinste Wert jeder Spalte ist gleich
(Median, Mittelwert, etc. ist auch überall gleich) (?)

=> ist sehr konservativ. Nach dem Normalisieren sehe ich vermutlich wenig Unterschiede, aber die die ich sehe sind relativ sicher

(0-ü) Mit Notebook (Python)

conda create --name week8 python=3 numpy scipy (...) pandas?
nachinstallieren: source activate week8
conda install jupiter

=> s. Elias

▷ Mitschrift vom Programm hat Elias. Hier wir Notizen:

- df ≙ data frame
- 11_CT2 etc. sind sample Namen
- describe liefert zu jeder Spalte Statistiken
  - → mean ≙ Mittelwert
  - → std ≙ Standardabweichung
  - → 50% ≙ Median
  - → x% ≙ Quantile
- „Magic-Comands" %matplotlib inline
  ↳ Notebookspezifisch (direkt im Dokument)

⇒ Programmcode in Git

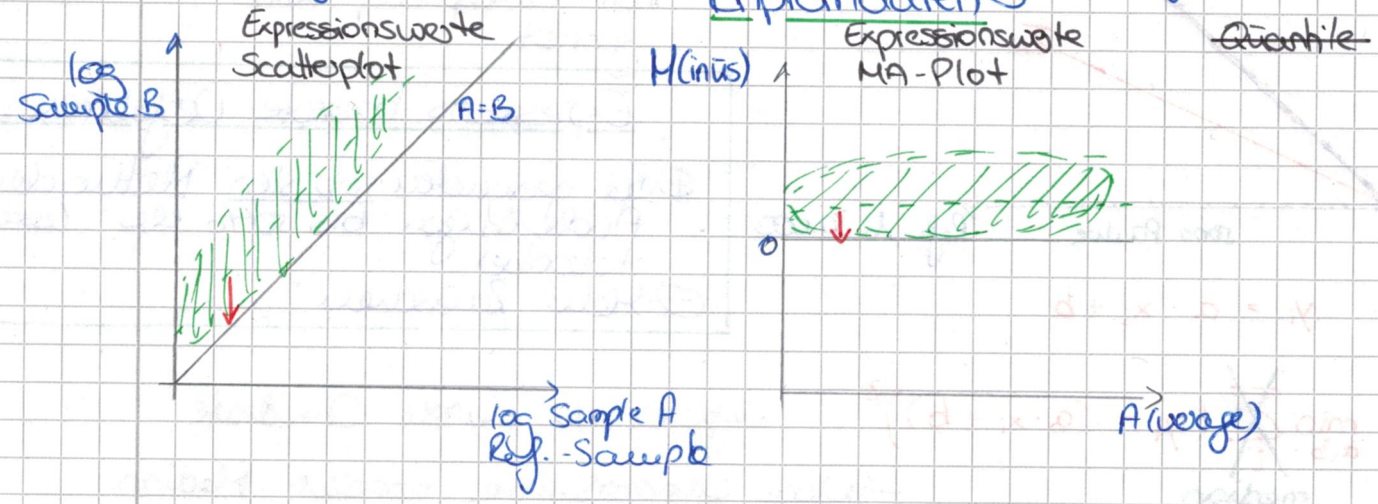
- Q-Q-Plot: Quantil gegen Quantil-Plot
  d.h. mit sortierten Werten
- pandas, um nicht auf rohen Arrays zu arbeiten

Normalisierung über alle Spalten ist sinnvoll

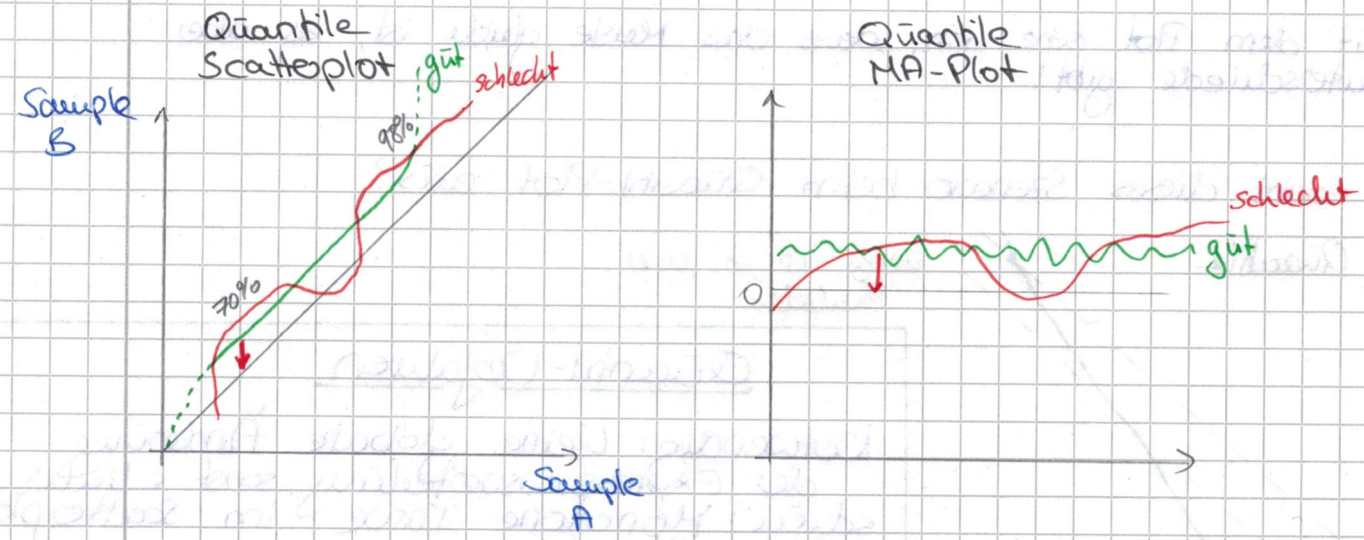
22.06.'17

Wiederholung:

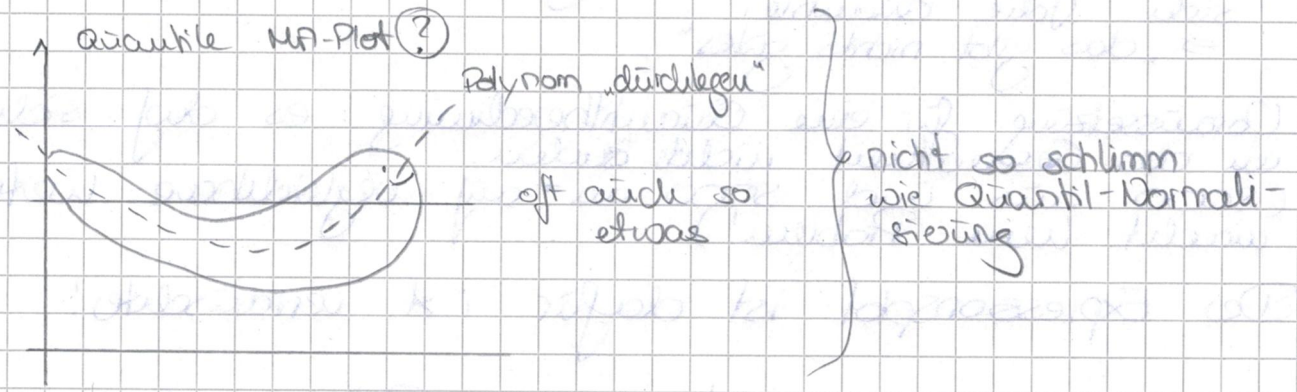
Diagnostische Plots / Normalisierungsplots für Transkriptomdaten



~~Quantile~~

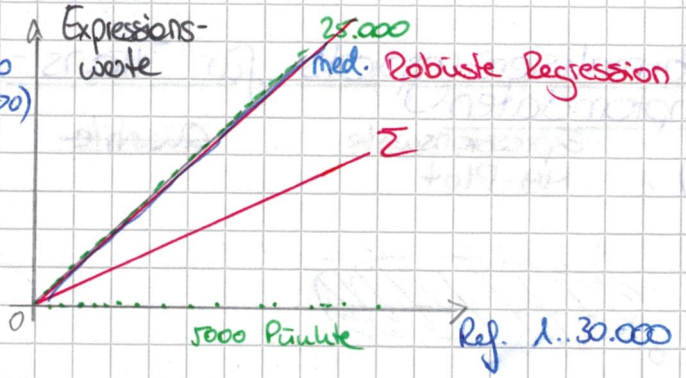


- Zweckmäßig: sich Referenzsample zu beschaffen
  - z.B. beste Qualität, meiste Sequenzierungen (= Beste)
  - Durchschnitt



- 70-98% der Quantile sollten schön gerade sein

- Ich schalte 5.000 Gene ab (sample x). Wie sieht Scatterplot aus?



→ Minderheit, die der Mehrheit widerspricht, wird ignoriert

$y_i = a \cdot x_i + b$

$\rightsquigarrow \min_{a,b} \underset{i}{\text{median}} \left(y_i - (a \cdot x_i + b)\right)^2$ Verfahren der kleinsten Quadrate

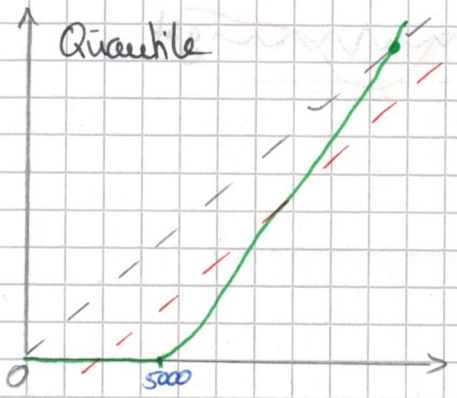
→ Nicht Quadratsumme, sondern Median
⇒ viel rechenaufwändiger

Expressions-basierte Verfahren:

⊕ Bei geeigneten robusten Methoden Ausreißer bis 50% der Gene tolerabel
⊖ Hoher Rauschen

⇒ mit dem Plot sehe ich, dass das Meiste gleich ist, es aber Unterschiede gibt!

- Wie sieht dieses Szenario beim Quantil-Plot aus?



Quantil-Verfahren

Notwendig: Keine globale Änderung der Expressionsverteilung, sonst s. links schief: Monotone Kurve im Scatterplot, kein Rauschen

⇒ d.h. wenn ein Großteil der Gene stark ändert (z.B. 0→hoch, oder auf 0), der Rest aber gleich bleibt, verschieben sich alle Quantile
⇒ „das gibt nichts gutes"

Voraussetzung für eine Quantilberechnung: es darf sich an der Gesamtheit nichts ändern.
(5000 aus und 5000 an + auf vergleichbare Werte macht kein Problem!)

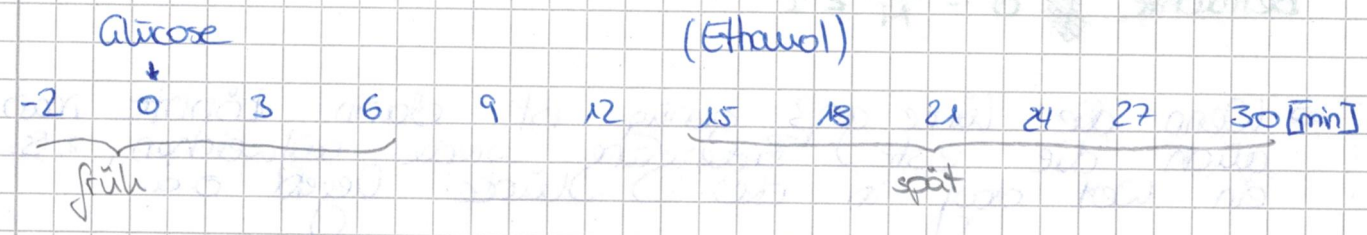
Der Expressionsplot ist dafür i.d. verrauschter!

➡ Beide Verfahren mitlaufen und Plots angucken

(Bevor man mit einer automatischen Pipeline drauf geht, sollte man sich das angucken. Deswegen macht man ja auch Plots.)

Jetzt hat man Normalisiert. Was macht man jetzt?

Beispielsatz: Hefe im Fermenter. Zugabe von sehr viel Glucose, alle 3 Minuten messen



Damit ich keine Unterschiede durch Sequenzierhefe oder Messungen bekomme, habe ich normalisiert.

Expressionsmatrix

	Samples (1-6)	
	3,4,7,9	3,4,7,9,10,9
Gene 6.500	niedrig	hoch
	$F_{früh}$	$S_{spät}$

„Prototyp" eines Gens, das deutlich unterschiedlich ist

Früh gegen spät vergleichen

Test auf Gleichheit der Mittelwerte

t-Test:

Annahme, dass die Beobachtungen normalverteilt um den Mittelwert streuen

① Berechne Teststatistik t (t-Wert) (Maß für Abweichung)

② Berechne $P_0(T \geq t) =: p$ (p-Wert)

T: Zufallsvariable zu t unter der Nullhypothese, dass keine Unterschied vorliegt.

Ist p sehr klein, lässt sich t nicht durch "Zufall" erklären. ≤ 0.01

* ≤ 0.05, ** ≤ 0.01 *** ≤ 0.001 könnte auch anders sein!

=> Fragwürdig, da es ja nicht normalverteilt sein muss! Trotzdem wird es oft gemacht.

Expressionswerte

$x_1 \leq x_2 \leq \dots \leq x_{10}$

früh früh früh früh // spät spät ...

Ist Δ groß genug?

Teststatistik z.B.
wie viele F-Werte habe ich gesehen bevor der erst S-Wert kommt?

alle Fs nach links, alle S nach rechts:

$$P = \frac{4! \cdot 6!}{10!}$$ (bei zufälliger Permutation!)

=> ist relativ klein, aber gar nicht mal so gut.

$$= \frac{1}{210} = 0.005$$

↳ da Permutation → wir haben 6.500 Gene

=> Jetzt kann ich mir noch die Lücke angucken, um Aussagen darüber zu treffen!

M z.B. verdoppeln. Aber Problem der kleinen Zahlen, deswegen addieren wir was dazu.

Sei $M := \max(F) + 10$

Betrachte $\delta := \frac{\Delta}{M} \geq 1$

⇒ Wenn die Lücke groß genug ist, dann könnte man auch die erste Bedingung wieder aufweichen, z.B. ein Wert darf in der Lücke liegen o.ä.

⇒ 1. Teil: Normalisierung → wichtig! Wenn man die verbotene ist der Rest sinnlos

2. Teil: Testen auf Unterschiede → statistische Mittel

z.B. t-Test (nicht so gut)
Range (wird erst signifikant, wenn man genug samples hat)

bei 4-6 Samples kann man nicht über 1/210 kommen. Mehr Samples werden aussagekräftiger ←

Anm.: wenn man zu strikt ist, findet man häufig 0 Gene, da es oft streut!

CO-VL

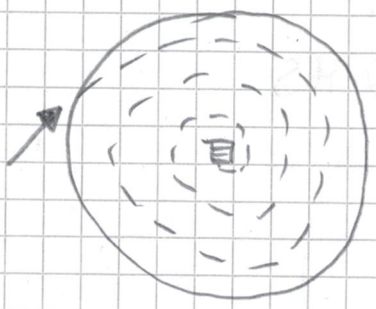
29.6.17

PROTEOMIK / MASSENSPEKTROMETRIE

ca. 100.000 €

TOF Spektrometer
(time-of-flight)
Zeit → Masse

Problem: ich brauche ein Vakuum

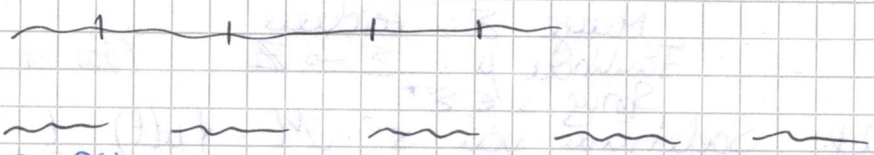


ca. 500.000 €

Proteine sind relativ große, schwere Objekte. So ein ganzes Protein bekomme ich vermutlich gar nicht überschleunigt.
Ich kann Proteine zerbrechen.

Idee

Protein
↓
lineare Kette von Aminosäuren:
↓ enzymatisch verdaut
viele Peptide (5-20 AS)



→ MS → Massen $\{m_1, m_2, m_3, \ldots\}$

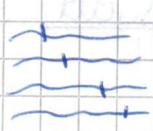
„Ist so 90er Jahre"

Fingerprint

- Man kann keine Reihenfolgen unterscheiden, d.h. 2 Peptide mit exakt den gleichen AS in unterschiedlicher Reihenfolge haben die gleiche Masse
- die Massen sind sich auch sonst oft sehr ähnlich

Was hat man sich überlegt?

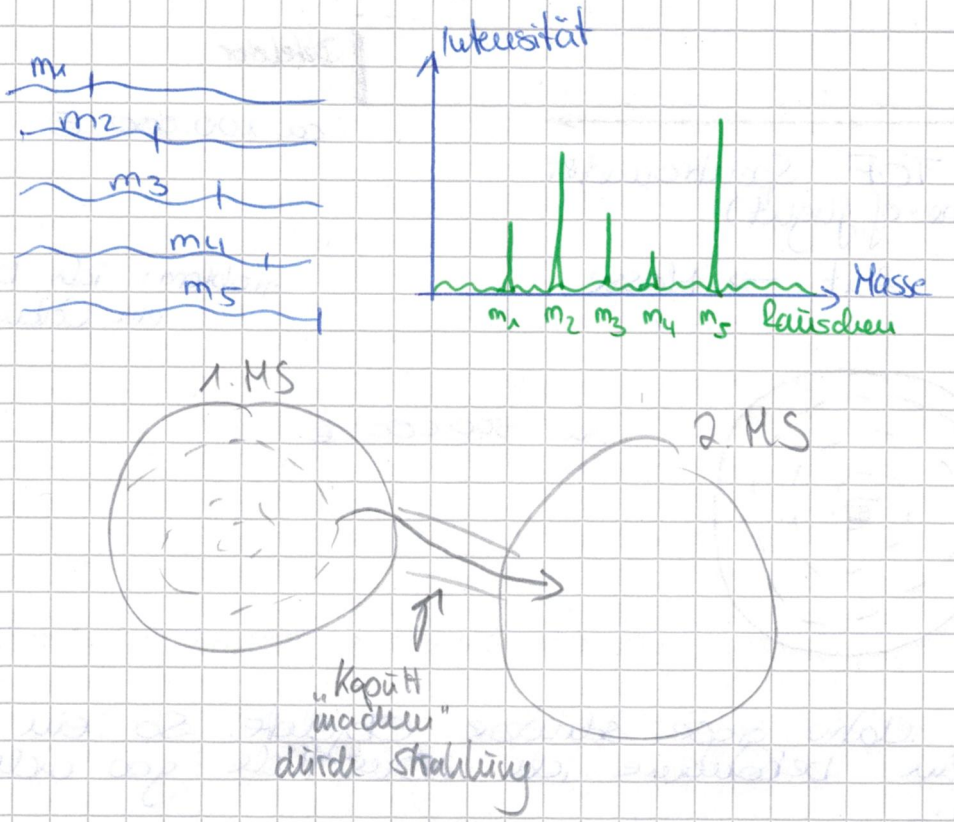
Für einzelne Fragmente 2. Spektrometrie



alle Präfixmassen

⇒ Damit lässt sich dann meist eindeutig das Protein bestimmen.

Nennt sich MS/MS



- Wie kann man das weit abstrahieren, damit man „allgemeine Dinge für kann"?

Geg.: Gewichtetes Alphabet (Σ, μ)

Menge Σ: Zeichen
Funktion $\mu: \Sigma \to \mathbb{Z}$ (das ist keine große Einschränkung)
String $s \in \Sigma^*$

Def.: Spektrum von s: $M = \{\mu(t) : t$ ist echtes Präfix oder Suffix von $s\}$
Mittelmasse (parent mass): $\mu(s)$

Bsp: $\Sigma = \{a, b, c, d\}$ μ: a: 2, b: 3, c: 7, d: 10

$s = a\ c\ a\ b$ (14)

$M = \{0, 2, 3, 5, 9, 11, 12, 14\}$

Annahme: Alle Präfix- und Suffixmassen sind verschieden. (oft falsch!)

Problem: Geg.: $M, \mu(s)$ Gesucht: s

In Praxis:

- Rauschen
- Massen fehlen
- Massen sind falsch
- Massen kommen mehrfach vor
- ...

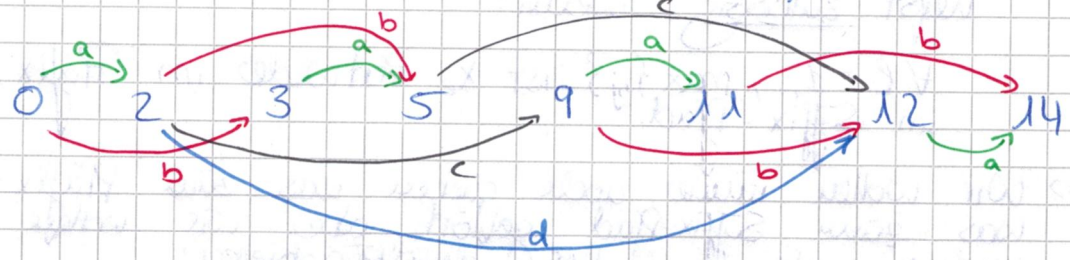
} hier: sehr starke Vereinfachung

Def: Spektrum-Graph

Knoten $V := M \cup \{0, M\}$

Kanten: $\{\bar{u} \to v : \exists \sigma \in \Sigma \text{ mit } \bar{u} + \mu(\sigma) = v\}$

Bsp:



Idee: Pfad $0 \rightsquigarrow M$ entspricht String

Beob.: $x \in M \Rightarrow M - x \in M$

Benenne Massen in $M \cup \{0, M\}$ als

$0 = x_0 < x_1 < \dots < x_n \leq y_n < \dots < y_1 < y_0 = M$

$2n$ verschiedene Massen

$x_i + y_i = M$

Einfaches Algorithmus:

2^n mögl. Zuordnungen $x_1, \dots, x_n$

→ Präfix oder Suffix

Probiere alle durch,
Teste ob String zulässig ist

Bsp: $M = \{0, 2, 3, 5, \dots\}$

p ✓ => Masse = 1, geht nicht ↯ a

p ✓ a b

s ✓ a ab

s ✓ Ende, wir brauchen $u+1$, also $= 7$

d.h. es fehlt $7 = c$

→ $a < ab$

Laufzeit: $O(2^n \cdot n)$

→ dann überprüfen!

Idee: Konstruiere 2 Pfade gleichzeitig
- Präfix - Pfad
- Suffix - Pfad

! disjunkt (wg. Annahme)

! Jeder Knoten muss besucht werden

Def.: Paar von Pfaden

$x_0 \rightsquigarrow x_i$ Präfix-Pfad
$x_0 \rightsquigarrow x_j$ Suffix-Pfad

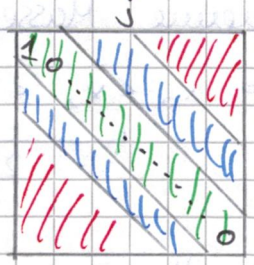
heißt zulässig, wenn:

$\forall \ell = 1, \dots, \max\{i,j\}$ ist x_ℓ entweder im Präfix-Pfad oder im Suffix-Pfad.

⇒ Wir wollen immer noch zählen was zum Präfix- und was zum Suffix-Pfad gehört, aber wir wollen es besser machen als (alles) durchzuprobieren!

Definiere Zulässigkeitsmatrix $D[0,\dots,n, 0,\dots,n]$

$$D[i,j] := \begin{cases} 1 & \text{wenn es ein zulässiges Pfadpaar } x_0 \rightsquigarrow x_i \text{ Präfixpfad}, \ x_0 \rightsquigarrow x_j \text{ Suffixpfad} \\ 0 & \text{sonst} \end{cases}$$



$D[0,0] = 1$

$D[j,j] = 0 \quad \forall j = 1, \dots, n$

$i \geq j+2$: $D[i,j] = \begin{cases} D[i-1,j] & \text{falls } x_{i-1} \to x_i \text{ Kante ist} \\ 0 & \text{sonst} \end{cases}$

$j \geq i+2$: analog

$i = j+1$ (Nebendiagonale)

$$D[j+1,j] = \begin{cases} \max\{D[\ell,j] : (x_\ell, x_{j+1}) \text{ ist Kante}, \ell = 0,\dots,j\} \\ 0 \text{ falls Menge leer} \end{cases}$$

$j = i+1$ symmetrisch

da die Diagonale nur Nen hat bis dahin alle 1

Laufzeit: $O(n^2)$ ($O(n)$-Zellen mit Laufzeit $O(n)$)
Konstant
Konstant

(SO) ⇒kann man verallgemeinern (ohne (falsche) Annahme)

Wiederholung

~~29.06.17~~
06.07.17

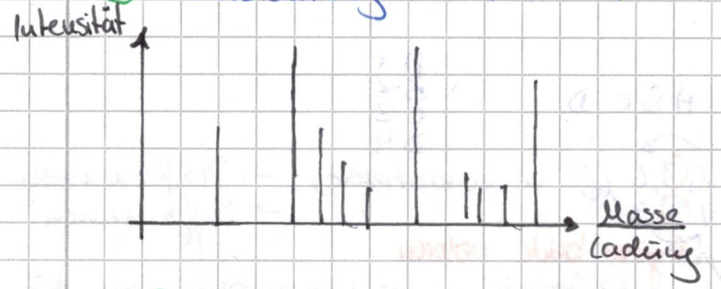
Def.: $D[i,j] = 1 \quad :\Leftrightarrow$ $\exists$ zulässiges Pfadpaar $(P\text{-}, S\text{-})$, so dass
Präfixpfad bei Masse x_i endet,
Suffixpfad bei Masse x_j endet
alle Massen bis $x_{\max\{i,j\}}$ werden genau 1x benutzt.

$$D[i,j] = \begin{cases} 1 & \text{falls } i=0,\ j=0 \\ D[i-1,j] & \text{falls } i \geq j+2,\ x_{i-1} \to x_i \\ D[i,j-1] & \text{falls } j \geq i+2,\ x_{j-1} \to x_j \\ \max\{D[\ell,j] : \ell=0,\dots,i-1 : x_\ell \to x_i\} & \text{falls } i=j+1 \\ \max\{D[i,\ell] : \ell=0,\dots,j-1 : x_\ell \to x_j\} & \text{falls } j=i+1 \\ 0 & \text{sonst} \end{cases}$$

$\max\{\} := 0$

Reale Bedingungen:

1. zusätzliche Peaks?
2. fehlende Peaks?
3. Präfixe / Suffixe mit gleicher Masse?
4. Gewichtung der Peaks?



mit meinem Peptid

▶ Zu (1): Ich will möglichst viele übereinstimmende Peaks haben, bzw. möglichst wenige zusätzliche.
Dabei kann man entweder nur zählen oder mit Gewichtung von Peaks arbeiten.

$Q[i,j]$:= maximale Anzahl (allg. max. Gewicht) von Peaks, die sich durch ~~zusätzli~~ zulässige Pfadpaare (Präfix $x_0 \to x_i$, Suffix $x_0 \to x_j$) erklären ~~kann~~ lassen.
$[x_0 = 0]$ $[\max\{\} = -\infty]$

$Q[0,0] = 0$

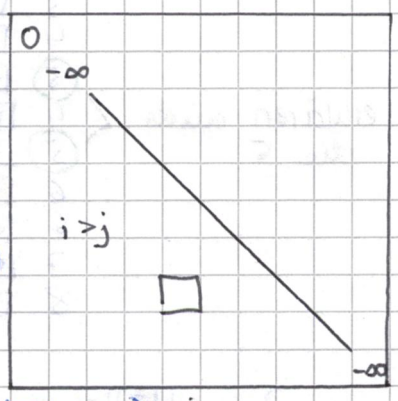
$Q[j,j] = -\infty$

$$Q[i,j] = \begin{cases} \max Q[\ell,j] + w(x_\ell, x_i) : 0 \leq \ell < i \text{ mit } x_\ell \to x_i \} & (i>j) \\ \text{oder:} \\ \max\{Q[\ell,j]+1 : 0 \leq \ell < i \text{ mit } x_\ell \to x_i\} & \\ \text{alternativ} \\ := \begin{cases} f(\text{Gewicht/Intensität der Masse } x_i, x_\ell) \\ -\infty \text{ sonst} \end{cases} & \end{cases}$$

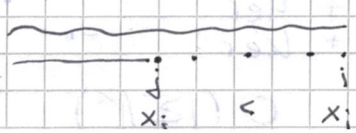
Kante von x_ℓ nach x_i hat Gewicht

$w(x_\ell, x_i) := \begin{cases} 1 & \text{wenn } x_\ell \to x_i \\ -\infty & \text{sonst} \end{cases}$

1 zusätzliches Peak erklärt



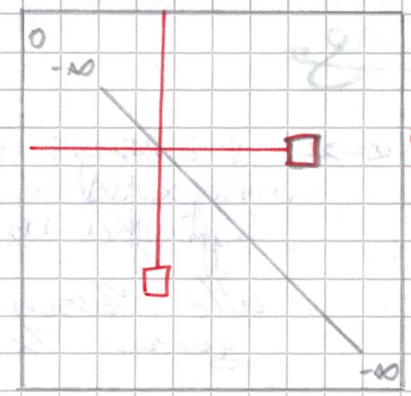
Suffixpfad endet bei i



Zeit: $O(n^3)$

Speicher: $O(n^2)$

Im Allg. kann man hier wenig tun um Speicher oder Laufzeit zu sparen.
Aber Peptide sind relativ klein.
n ca. 30, deswegen nicht ganz so schlimm.
Es ist superschnell!



um das zu berechnen muss ich die ganze Zeile nehmen

um das zu berechnen muss ich die ganze Spalte nehmen

„Eigentlich ist n eine Konstante in dem Algorithmus. Wenn man in der Prüfung sagt $O(1)$. Warum? n ist hier die Konstante (=30)"

▶ Zu ②: fehlende Peaks

a) von einem Paar fehlt nur P oder S (nicht beide)

$$X \longmapsto X_{sym} = \{x, M-x : x \in X\}$$

symmetrisch

b) beide Massen fehlen

Wie kann das passieren? A B C D

A 1
B 2
C 3
D 4

o 1, 3, 6, 10 (10 ist Gesamtmasse) → Präfixmassen
4, 7, 9 → Suffixmassen

gehen beide verloren
=> Spektrum ist symmetrisch zu 10

Gap von 5
Es gibt keine Masse von 5. Kann ich 5 trotzdem erklären?

=> Ich möchte mir nicht einen neuen Algorithmus einführen, deswegen kommt jetzt ein Kunstgriff: Ich vergrößere das Alphabet:

2	AA		
3	AB	=	BA
4	AC	=	CA
(5)	AD	=	DA
4	BB		
(5)	BC	=	CB
6	BD	=	DB
6	CC		
7	CD	=	DC
8	DD		

erklären auch die 5

alle Multimengen mit 2 Buchstaben

=> möglichst immer die einfachste Lösung nehmen! Bei Masse 3 => C

+ 3er Multimengen
+ 4er —"—
+ ker: —"—

} erst wenn ich nicht weiterkomme. Explodiert kombinatorisch

(52) $O(|\Sigma|^k)$

Alphabet auf der Multimengen erweitern
⋮
→ zusätzliche Massenerklärungen.

⇒ Algo aus ④ kann verwendet werden. Es muss da nichts mehr zusätzliches gemacht werden.

Es gibt ein kleines Problem mit der Laufzeit ist aber Interpretationssache.
Bisher Alphabet klein → konstante Zeit
Jetzt bläht ich das Alphabet auf. Eigentlich immer noch konstant, dann bleiben die Laufzeiten.
In der Praxis ist es sinnvoll ein Array [oder Hashmap (O(1) erwartet)] nach Größe sortiert und dann binäre Suche.

Zeit: $O(n^3) \cdot \log(|\Sigma|^k)$ (binäre Suche, theoretisch auch perfektes Hashing $O(1)$)
Speicher: $O(n^2) + O(|\Sigma|^k)$

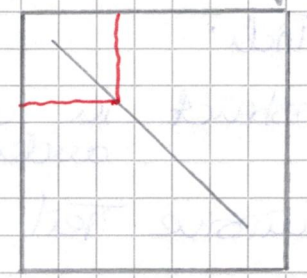
kann irgendwann genauso groß sein wie $O(n^2)$

▷ Zu ③: Präfixe / Suffixe mit gleicher Masse

Fall $Q[j,j]$ neu überlegen

Peak darf nicht doppelt gezählt werden. Ich will das nicht auch noch belohnen (ich will es aber auch nicht bestrafen)

Gleiche Formel wie vorher, nur nicht mit +1



Zulässig ist jetzt: Präfixmasse, Suffixmasse, beides, zusätzliche Masse

$$Q[j,j] \overset{\max}{=} \begin{cases} \max\{Q[\ell_j] : 0 \le \ell < i \text{ mit } x_\ell \to x_i\} \\ \max\{Q[\ell_i] : 0 \le \ell < j \text{ mit } x_\ell \to x_j\} \end{cases}$$

„Ist elegant und symmetrisch."

Wir haben jetzt so getan, als könnte man die Strings zwischen den Buchstaben durchschneiden. Peptide sind Molekülketten, die man an verschiedenen Stellen trennen kann.

A · {· {· B · · · C · · · D

a-x Paare
b-y -"-
c-z -"-

⇒ Es gibt drei verschiedene Möglichkeiten dazwischen durch zu schneiden.

Einfach Alphabet vergrößern? In Praxis nicht so einfach möglich.

„Prüfsummen werden sehr unübersichtlich". Besser zusätzliche Typen, als Alphabet vergrößern.

Bisher nur kombinatorisch.

(Informatiker lieben kombinatorisch, „zählen Liebe". Statistiker machen das alles mit Wahrscheinlichkeiten.
Statt Maximum summen über Wkten.
=> Umformulieren in probabilistische Modelle)
→ Man könnte das Ganze mit HMMs (hidden Markov Modelle) machen.

=> Alle Probleme gelöst!

06.07.17 [CO-Ü]

Oben steht das „wahre String", dann die „wahren Massen", hier als String. Dann die Massen ohne ihre Vielfachheiten und ganz unten das Alphabet.

Traceback ist implementiert.
Das mit dem * ist das, den ich skriptiert habe und das sollte mir einmal rauskommen.

! secret.py ist auf dem Laptop, aber nicht im Git. ☺
invalphabet = invertierte Alphabet; Hashtabelle: Masse zu Buchstaben.
Secret in 61 Zeilen möglich (inkl. Kommentare)
Kleines Trick:
- reconstruct in innere Fkt zeigen, die rekursiv ist
andere Teil der dann Rest macht

Rekursive Teil → factore von Python (überhaupt möglich in C++?
falls jemand weiß: 2 Longisland-Iceteas im Atlantico von Sven spendiert. Damach erklären ☺)

→ Hepatozyten schreiben ohne Gedanken zum Methoden zu machen.

→ Es kann mehrere Vorgänger geben

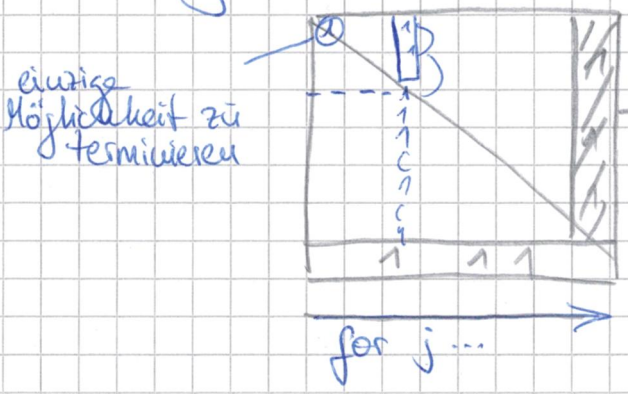
→ (...)
"yield from", "yield" benutzen

(ich bin zu langsam und lerne nicht alles auswendig)

Traceback: wenn oben links, dann ist Präfix und Suffix aufgebaut und dann weiß ich damit muss ich gucken, wie ich es aufbaue?

2 Möglichkeiten rekursiv aufbauen:
- globale Liste aufbauen leer, bei möglicher Lösung erweitern kann groß werden, da es viele Lsg. gibt

- elegante: Kopfhöhe so zu gestalten, dass Ausgabe verschränkt ist mit Generierung



Die aktuelle Funktion wird mitten im Ablauf jeweils gebrochen, damit die andere Funktion sie direkt weiterverarbeitet.

for s in reconstruct (...)

↓

dann "yield s"

(...)

↓

wieder ein "yield r"

↓

(...)

Jens hat es verstanden!

⇒ Ich definiere keine normale Fkt, sondern einen Generator.

Vgl. "[x**2 for x in count()]" vs. "g=(x**2 for ...)"

(bzw. generator-function)

➡ in Python Generator-Functions mit yield-Statements

⇒ Secret nach ins Repo

▷ Noch erweitern: • Argumente für loss, additional
• dann für Q-Matrix
• Alphabetvergrößerung

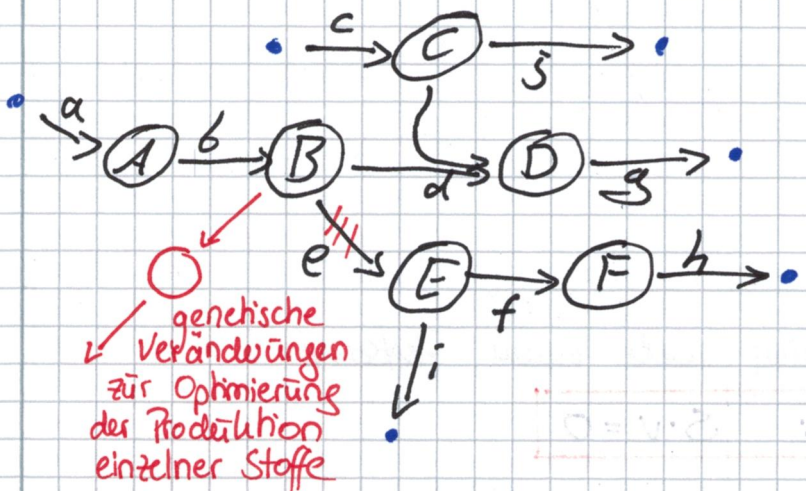
Man berechnet in der Klasse i.d.R. die TOP50 oder 20 Formeln (bei Profis vllt. Top 10).
Im Prinzip auch suboptimale (sg.) berechnen.

co-nl

13. Juli 2017

Metabolomics

Analyse betabolischer Netzwerke



- externe Metabolite (nicht betrachtet)

A-F: Stoffe, compounds, Metabolite

a-5: Reaktionen

- häufig tatsächlich grafisch dargestellt
  -> schlecht.

➡ stöchiometrische Matrix

▶ Allgemein: chemisches Reaktionsnetzwerk "ist" eine stöchiometrische Matrix

$$S \in \mathbb{Z}^{\#\text{Stoffe} \times \#\text{Reaktionen}}$$

S_{ij}: Anzahl Moleküle von Stoff i, die in Reaktion j produziert ($S_{ij} > 0$) bzw. verbraucht ($S_{ij} < 0$) werden.

$S =$

	a	b	c	d	e	f	g	h	i	j
A	1	-1								
B		1		-1	-1					
C			1	-1						-1
D				1			-1			
E					1	-1			-1	
F						1		-1		

sehr dünn besetzt

- Nullen sind nicht eingetragen!

Hier der Einfachheit wurden -1en genommen, aber es könnten auch andere Werte gewählt werden
=> im Gegensatz zum Netzwerkbild oben kann man Verhältnisse darstellen

▶ Flux-Vektor:

$$v \in \mathbb{R}^{\#\text{Reaktionen}}$$

beschreibt Aktivität jeder Reaktion (Rate, mit der Reaktion Stoffe umsetzt)

Bsp.:

$$v = \begin{matrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \\ i \\ j \\ \vdots \end{matrix} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ \vdots \end{pmatrix}$$

Jede Reaktion ist gleich aktiv!
(⇒ einfaches Beispiel)

$$\Delta \text{Stoffe} = S \cdot v$$

gibt die Stoffänderung nach einem Zeitschritt

stationärer Zustand: $S \cdot v = 0$

Reaktionen:

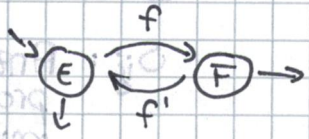
- reversibel ($\text{Flux} \in \mathbb{R}$)
- nicht reversibel ($\text{Flux} \geq 0$)

reversible Reaktionen können wir immer in nicht reversible Reaktionen zerlegen (Hin- und Rückrichtung jeweils separate Reaktion)

Konvention:

Alle Reaktionen nicht reversibel!
(Wenn doch: zerlegen in nicht reversibel
- vorwärts
- rückwärts)

$$v \geq 0$$



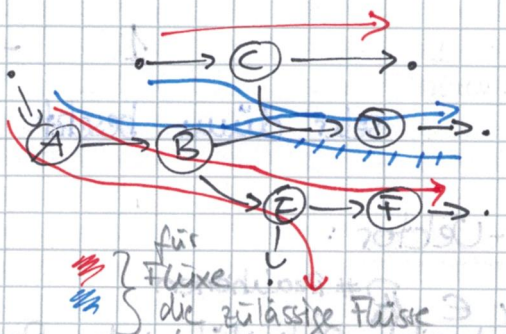
Aufgabe:

Charakterisiere $Z = \{v : S \cdot v = 0, v \geq 0\}$

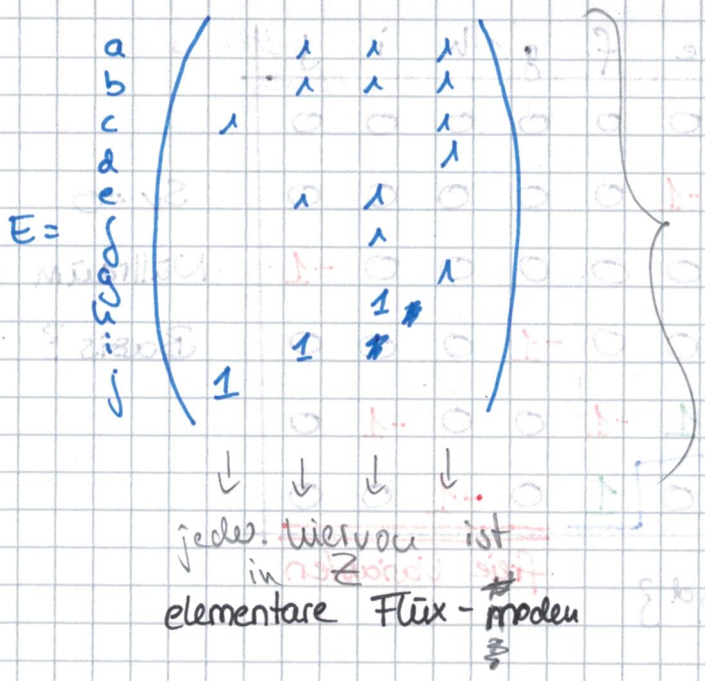
Menge der zulässigen Flüsse

$0 \in Z$ ✓

$$v = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$ ✓



für Flüsse die zulässige Flüsse haben ⇒ 4 verschiedene



Idee: Beschreibe Σ als $\{v : v = E \cdot \lambda,\ \lambda \geq 0\}$

nicht negative Linearkombinationen von Spalten (Flüxe aus Σ) von E, E minimal.

Spalten von E: "Elementarmoden"
EFMs
↑ elementare Flux-Moden

Frage: $S \rightarrow E$?

(in den Spalten)

Matrix der Basisvektoren des Nullraums (Kerns) von S

Bekannt: $\{v : Sv = 0\} = \{E \cdot \lambda,\ \lambda \in \mathbb{R}^d\}$

Lösung: Vektorräume

Geg.: S
Ges.: E

homogenes lineares Gleichungssystem

Hier: $\{v : Sv = 0,\ v \geq 0\} = \{E \cdot \lambda,\ \lambda \geq 0\}$ erhöht enorm die Komplexität

Geg.: S
Ges.: E

In der Praxis kann man Probleme ohne E lösen.

Typisches Problem (ohne E):

$$\max.\ w^T v$$

Gewichts-vektor ↗

$$\text{s.d.}\quad Sv = 0$$
$$v \geq \underline{\varepsilon}$$
$$1^T v = 1$$
~~$by \geq \underline{\varepsilon}$~~

w: Zielfunktion
$\underline{\varepsilon}$: Vektor von Mindestflüssen

$$(1 \ldots 1)\begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix} = 1$$
$$= v_1 + \ldots + v_n = 1$$

CO-Ü

$S =$

	a	b	c	d	e	f	g	h	i	j
A	1	-1	0	0	0	0	0	0	0	0
B	0	1	0	-1	-1	0	0	0	0	0
C	0	0	1	-1	0	0	0	0	0	-1
D	0	0	0	1	0	0	-1	0	0	0
E	0	0	0	0	1	-1	0	0	-1	0
F	0	0	0	0	0	1	0	-1	0	0

freie Variablen

$S_V = 0$

Nullraum von S.

Basis?

$\{v : v = B \cdot \lambda,\ \lambda \in \mathbb{R}^d\}$

Beispiel um Rechnung zu verdeutlichen:

$$\begin{matrix} \text{I} \\ \text{II} \\ \text{III} \end{matrix} \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

$$\begin{matrix} \text{II} - 4 \cdot \text{I} \\ \text{III} - 7 \cdot \text{I} \end{matrix} \begin{pmatrix} 1 & 2 & 3 \\ 0 & -3 & -6 \\ 0 & -6 & -12 \end{pmatrix}$$

$$\text{III} - 1 \cdot \text{II} \quad \begin{pmatrix} 1 & 2 & 3 \\ 0 & 1 & 2 \\ 0 & 1 & 2 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 2 & 3 \\ 0 & -1 & 2 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

x_3 beliebig : λ_1

x_3: beliebig

$x_2 = -2x_3$

$x_1 = -3x_2 - 2x_2$

$= -3x_3 + 4x_3 = x_3$

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} x_3 \\ -2x_3 \\ x_3 \end{pmatrix} = \lambda_1 \cdot \begin{pmatrix} 1 \\ -2 \\ 1 \end{pmatrix}$$

alle Vektoren dieser Form lösen dieses System

$$= \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix} \cdot [\lambda_1]$$

Basisvektor — Koeffizient

mit ein Basisvektor da Dimension 1

=> Unsere Matrix S ist schon in Stufenform.

v_g, v_h, v_i, v_j beliebig $= \lambda_1, \lambda_2, \lambda_3, \lambda_4$

$v_f = v_h$

$v_e = v_f + v_i$

$= v_h + v_i$

$v_d = v_g$

$v_c = v_d + v_j$

$= v_g + v_j$

$v_b = v_d + v_e$

$= v_g + v_h + v_i$

$v_a = v_g + v_h + v_i$

4 verschiedene beliebige Variablen, deswegen hat meine Basis die „Größe" 4.

4 Basisvektoren!

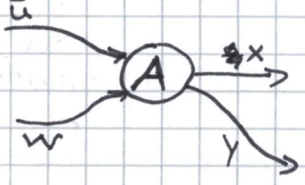
zufällig alle ≥ 0

$$v = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \lambda_4 \end{bmatrix} \begin{matrix} = v_g \\ = v_h \\ = v_i \\ = v_j \end{matrix}$$

ab hier die Identität

i. d. R. viel mehr Elementarmoden als Basisvektoren!

ANDERES BEISPIEL



4 Elementarmoden: ū x, ū y, w x, w y

Weg: $\bar{u} + w + 2y$ ist keine Elementarmode, da es eine Kombination von $\bar{u}y$ und wy ist.
⇒ Elementarmoden dürfen keine Kombination aus anderen sein

$$S = \begin{array}{c|cccc} & \bar{u} & w & x & y \\ \hline A & 1 & 1 & -1 & -1 \end{array}$$

3 freie Variablen

$S \cdot v = 0$

$v_{\bar{u}} = (-1\ 1\ 1) \begin{pmatrix} v_w \\ v_x \\ v_y \end{pmatrix}$

$$v = \begin{matrix} \bar{u} \\ w \\ x \\ y \end{matrix} \begin{pmatrix} -1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{pmatrix}$$

Allgemeiner:

$$A = \begin{bmatrix} S \\ -S \\ I \end{bmatrix} \rightarrow \text{Identität}$$

sehr viel allg. für beliebige Matrix

$\{v : Av \geq 0\}$

$\Leftrightarrow \{P\lambda : \lambda \geq 0\}$

Hyperebenenform

Linearkombination von zwei Ebenen?

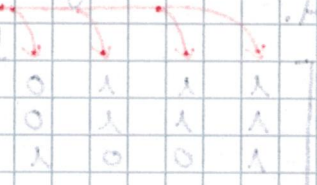
„P steht für polyedrischer Kegel oder so"

$A \rightarrow P$
$P \rightarrow A$

=> Problem aus der Geometrie

Double Description Method

Da man nicht reversible Zyklen zulässt, bekommt man auch Elementarmoden, die unnötig aufblähen, aber keinen Mehrwert haben.



20. Juli '17

Geg.: Matrix S, dann Flüxkegel

$$F := \{v : Sv = 0,\ v \geq 0\}$$
$$= \{R \cdot \lambda,\ \lambda \geq 0\}$$

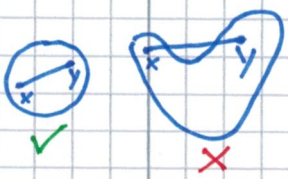
$S \mapsto R$?
R minimal

Warüm Kegel?

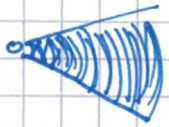
Def.: Punktmenge heißt

- Kegel, wenn $x \in C$ auch $\alpha x\ \forall \alpha \geq 0$ enthalten ist

- konvex, wenn mit $x, y \in K$ auch $\lambda x + (1-\lambda) \cdot y \quad \forall \lambda \in [0,1]$ enthalten ist (Verbindungsstrecke)



✓ ✗



konvexer Kegel

- Konvexer Kegel heißt polyedrisch, wenn $P = \{x : Ax \geq 0\}$ für eine Matrix A.

$\Rightarrow F$ ist polyedrischer konvexer Kegel $\left(A = \begin{bmatrix} S \\ -S \\ I \end{bmatrix}\right)$ $\quad \left.\begin{matrix} Sx \geq 0 \\ -Sx \geq 0 \\ x \geq 0 \end{matrix}\right\} Sx = 0$

↑ relativ spezielle Form, die wir auch ausnützen, aber man kann auch alles was wir sagen allgemein fassen

- Für einen konvexen polyedrischen Kegel P heißt (A, R) Doppelbeschreibungspaar, wenn

$$P = \{x : Ax \geq 0\} = \{Rx : \lambda \geq 0\}$$

↑ Schnitt von Halbräumen (Zeile ≙ Hyperebene, Normalvektor)

↑ positive Kombination von („Extrem)strahlen" des Kegels (Spalte ≙ Strahl in P)

Algorithmüs: Doppelbeschreibungsmethode (engl. double description method)

$$A \mapsto R \mapsto A$$

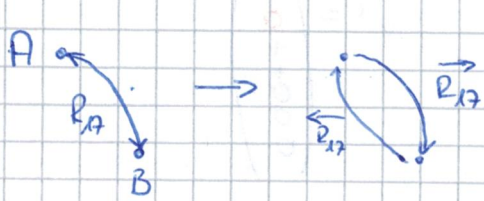
ist die allgemeinst mögliche Lösung für das Problem. Sollte man mal gehört haben, aber da unser Problem spezieller ist, kann man das ausnützen.

Nullraum-Methode (Wegue, ca. 2005)

1. Bestimme Basis von $\{v : Sv = 0\}$ (Nullraum von S)

Basis B hat die Form $B = \underbrace{\left[\frac{I}{K}\right]}_{\text{Dim. Nullraum}}$ } # Reaktionen für ein K.

(Falls es reversible Reaktionen gab und wir daraus zwei irreversible gemacht haben:
Betrachte diese separat; danach nicht mehr.
$B = \left[\frac{I}{K}\right]$



Es gibt Basisvektoren (Fahle cycles)

$\vec{R}_{17}$ $\begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$ $\overleftarrow{R}_{17}$

2. Iteration: Schritt j: Annahme: Zeilen bis $j-1$ enthalten nur Einträge ≥ 0

Zeilen ab j evtl. auch neg. Einträge

Zu Beginn: Schritt $j = d+1$

Teile Spalten von B an Hand Vorzeichen in Zeile j in 3 Klassen

≥ 0		
+ | 0	+ | -	0 | -
? ? ?		

j

3 Möglichkeiten +, - oder 0

$K^+ := \{k : B_{jk} > 0\}$ übernehmen

$K^0 := \{k : B_{jk} = 0\}$ übernehmen

$K^- := \{k : B_{jk} < 0\}$ weglassen

⊕ neue zulässige Strahlen generieren, die die aus K^- enthalten

Generierung neuer Strahlen: Alle Paare aus $K^- \times K^+$, so dass in Zeile j Nullen erzeugt werden:

$r^*_{i,k+} := r_{j,k+} \cdot r_{j,k-} - r_{j,k-} \cdot r_{i,k+}$

Zahl > 0 $\begin{array}{|c|}\hline \geq 0 \\ - \\ \hline \end{array}$ $\underbrace{-\text{Zahl} < 0}_{\text{pos. Linearkombination}} \cdot \begin{array}{|c|}\hline \geq 0 \\ + \\ \hline \end{array} = \begin{array}{|c|}\hline \geq 0 \\ 0 \\ \hline \end{array}$ j

3. Abschluss: Reduziere das finale $\bar{B}$ auf Elementarmoden:

1. $v \in F$
2. Minimalitätseigenschaft:

$Z(v) := \{j : v_j = 0\}$ Nullmenge von v

Es gibt kein v' in $F (v' \neq 0)$ mit $Z(v') \supsetneq Z(v)$

$$v = \begin{pmatrix} 0 \\ + \\ + \\ 0 \\ 0 \\ 0 \end{pmatrix} \qquad v' = \begin{pmatrix} 0 \\ 0 \\ + \\ + \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

keine El. mode

CO-Ü Metabdomics-Beispieldatei im GIT yeastmodel.txt → als Excel-Datei gespeichert (nur 1 Spalte) ⇒ jetzt wieder als

1. Dateiformat

// Kommentarzeile
– Headline / Überschrift
ENZ ≙ Enzym
REV ≙ Reversibel
MET ≙ Metaboliten
INT ≙ intern
EXT ≙ extern
CAT ≙ ?

Name : Edukte = Produkte .

ecdimodel.txt 2. Dateiformat

⇒ nicht so schön
⇒ man sieht nicht was reversibel ist

CO-VL

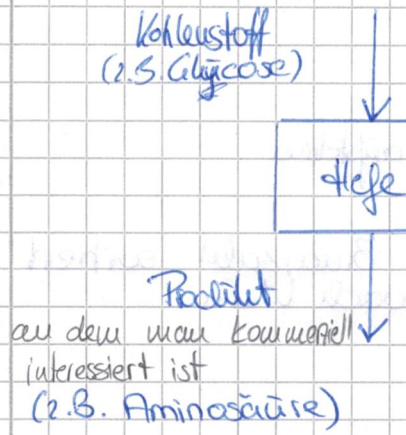
27. Juli 2017

Metabolic Engineering

z.B. Hefe, E.coli

- Löschen von Reaktionen (heute)
- Hinzufügen von Reaktionen
  (Reaktion ≙ Enzym(e) ≙ Gen(e)).

▶ Ausgangssituation: Stöchiometrische Matrix S, Elementarmoden-Matrix E

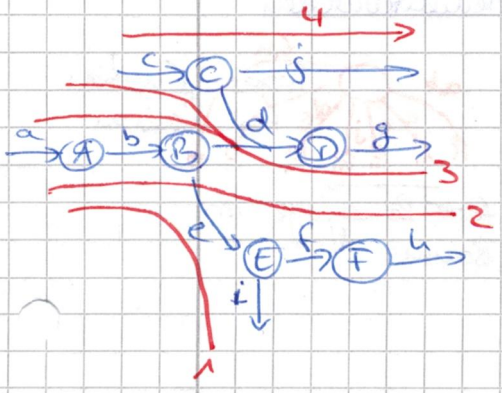


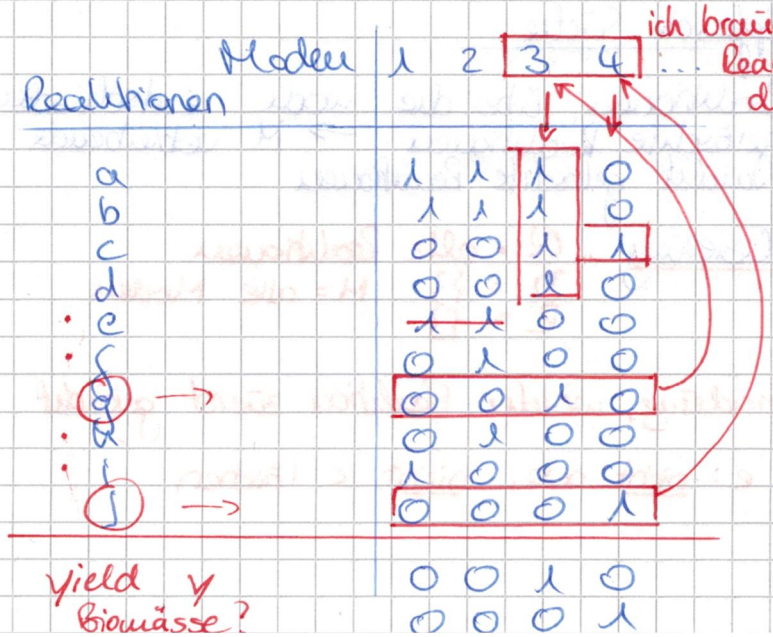
Wie optimiert man das?
Nachdem man das einmal gemacht hat, braucht man keine Elementarmoden.

Löschen: Welche Reaktionen muss ich löschen?

▶ Idee: Lösche ~~alle~~ wenige (?) Reaktionen, so dass Hefe nur Produkte (~~max~~ hoch) und Biomasse (≥T) produziert!
≥(1-ε)max

„Recht" noch kombinatorischem Aussuchproblem.



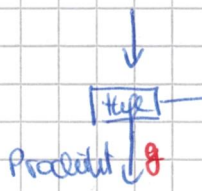


ich brauche alle Reaktionen die da drin sind

e, f, h, i sind zu viel wenn ich e lösche, dann sind direkt El.moden 1 und 2 weg

Elementarmoden 1-4

Wir haben Interesse an Produkt g und Biomasse j



} es muss nur e ratisgewogen werden (f,h und j unnötig)

⇒ s. Matrix auf der letzten Seite.

Aufstellen von ILP oder selber durchprobieren? Was geht schneller?

▶ Vorgehen:
1. Reaktionen sortieren
2. Reaktionen komprimieren
3. Inklusionsbez. zwischen Reaktionen aufstellen
4. Suchbaum

1.

a	1	1	1	0	] =
b	1	1	1	0	
e	1	1	0	0	
i	1	0	0	0	
f	0	1	0	0	] =
h	0	1	0	0	
c	0	0	1	1	
d	0	0	1	0	
g	0	0	1	0	
j	0	0	0	1	

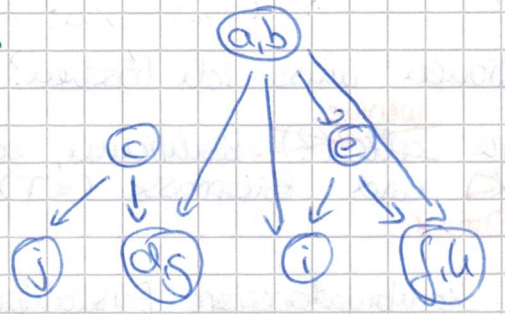
jede Binärzahl sortiert von groß nach klein

Transitive Inklusionen werden mit reingenommen (deswegen kein Hasse-Diagramm)

2.

a,b	1	1	1	0
e	1	1	0	0
i	1	0	0	0
f,h	0	1	0	0
c	0	0	1	1
d	0	0	1	0
g	0	0	1	0
j	0	0	0	1

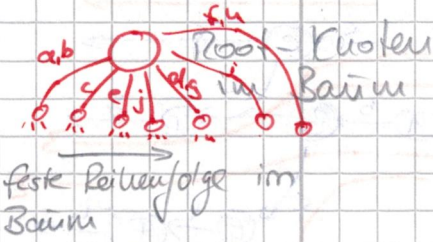
3.



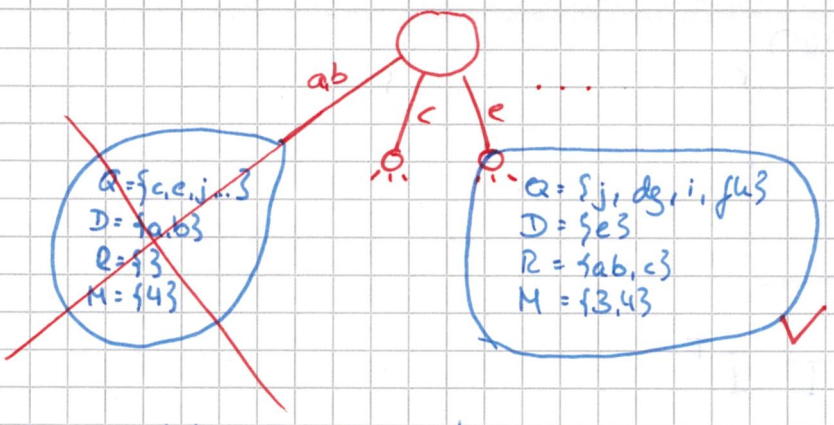
4. Erschöpfende Suche

Q: Reaktionen, über die noch entschieden werden muss
D: gelöschte Reaktionen → M verbleibende Elementarmoden
R: nicht gelöschte Reaktionen

Initialisierung Q = alle Reaktionen
D = {}
R = {}
M = alle Moden



- Entscheidung, welche Reaktion zuerst gelöscht wird
(z.B. e: nicht a,b, nicht c löschen



- Prüfe verbleibende Elementarmoden:

min product yield	~~0~~	0	• Lösung?
max product yield	~~0~~	1	• zulässig?
biomass	1	1	• weiter verzweigen?
reactions deleted	1	1	
	ab	e	

↓

eigentlich will ich es vermeiden, dass es Elementarmoden gibt, die kein Produkt machen, aber das kann man nicht immer vermeiden

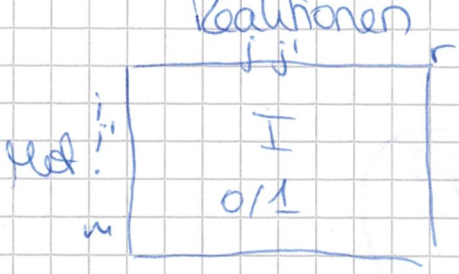
Sehr stumpfsinniges Algorithmus. Alles aufzählen. Früh abbrechen, wenn es keinen Sinn macht.
„Eigentlich macht ein ILP-Solver auch nichts anderes"

Es werden alle Lösungen angezeigt. Dann kann Biotechnologe pragmatisch auswählen, z.B. eine der Lösung ausschließen, weil man eines der Gene nicht auskneoken kann und stattdessen lieber die 5e Lösung wählen.

Berechnungen für's Löschen braucht wesentlich weniger lang als Berechnung der Elementarmoden!

Funktioniert wirklich gut!

Suchbaumalgorithmen sind wichtiges Werkzeug (auch wenn es intellektuell nicht so anspruchsvoll ist).



$I^T \cdot I$ $\quad$ $I \cdot I^T$

$(r \times r)$ $\quad$ $(m \times r)(r \times m) = (m \times m)$

$$(I I^T)_{i,i'} = \sum_j I_{ij} \cdot I_{i'j}$$

$$= \sum 1[i,i' \text{ in } R_j]$$

= # Realtionen mit i und i' generieren

Frage aus Fragekatalog (1 Seite 4. Pkt. von unten)

2. Frage: Mais = Säuge

18 besetzt den De Bruijn Graph voll, also mind. 16. Gerade ist doof, also 17. Da Abstand: 25