IEEE ICCABS

UCONN
UNIVERSITY OF CONNECTICUT

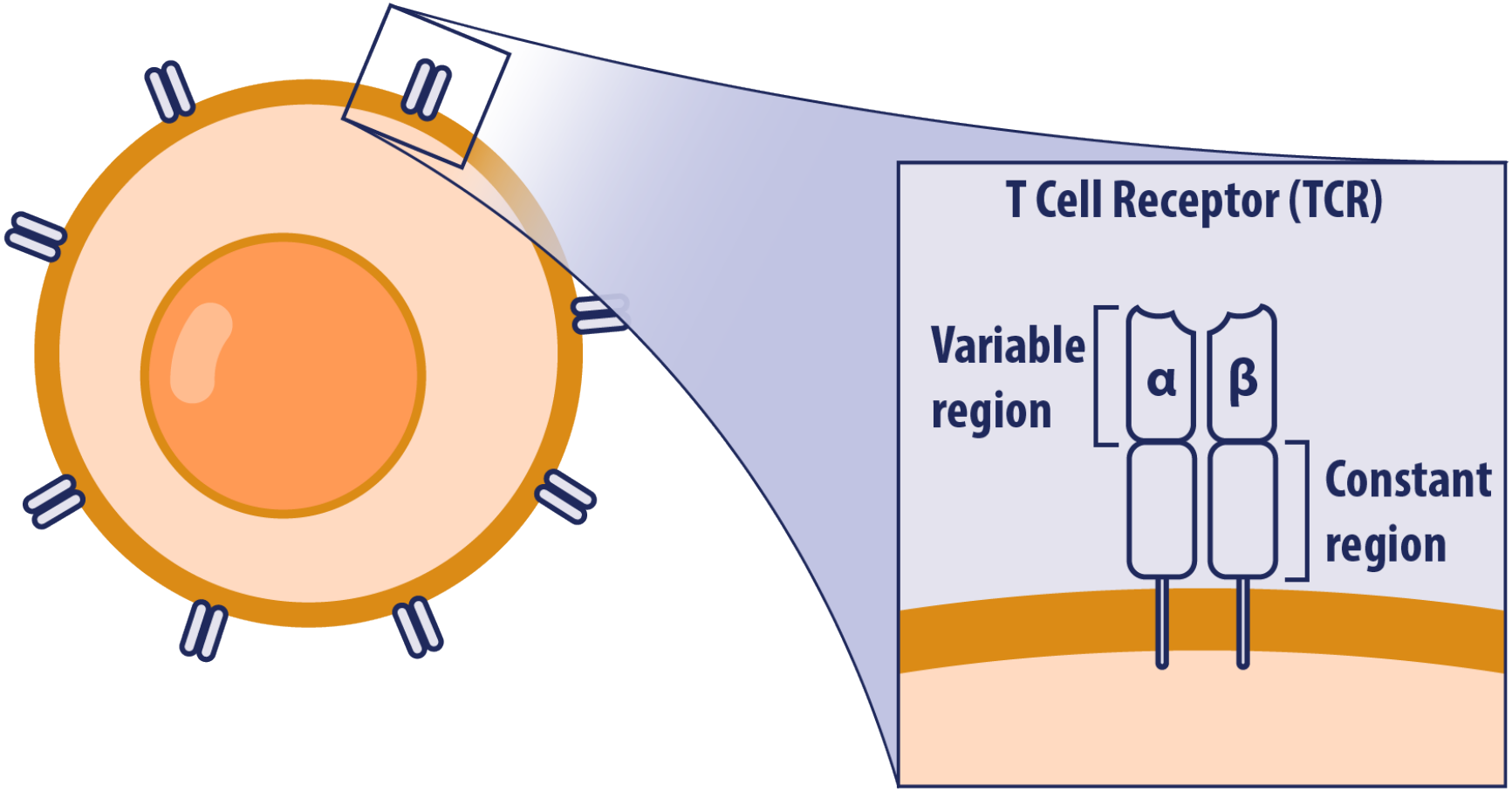
Identifying αβ T cell clones via pooling and b-matching*

Tyler Daddio, Ion Măndoiu
University of Connecticut

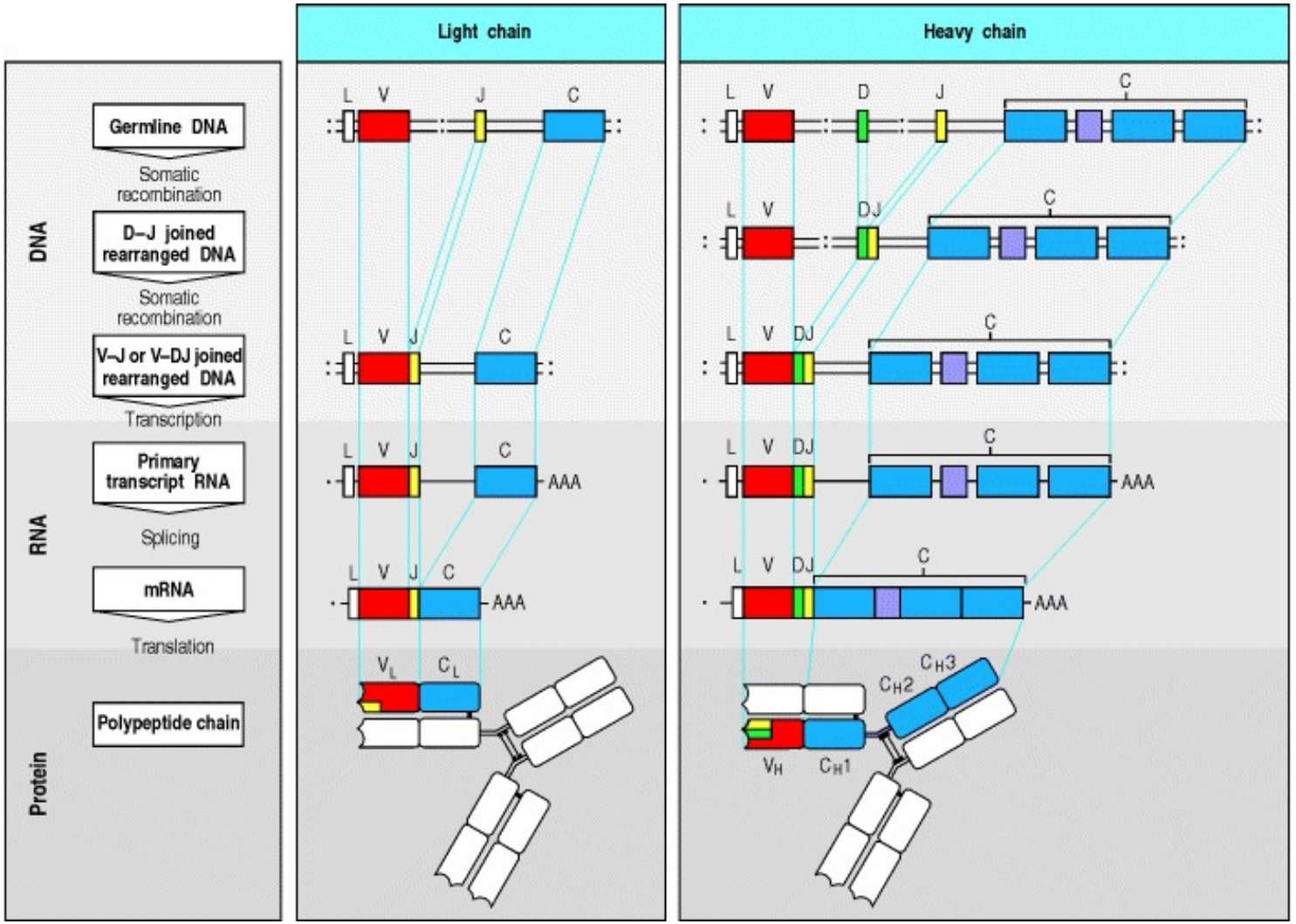
ICCABS 2017

* but mostly MinHashing

αβ T cells



Formation of T cell receptors



Janeway CA Jr, Travers P, Walport M, et al. Immunobiology: The Immune System in Health and Disease. 5th edition. New York: Garland Science; 2001.

Multiple rearrangements?

- Up to 30% of T cells contain two in-frame α-recombinants (i.e. dual alpha)
- Up to 2-10% contain two in-frame β-recombinants (i.e. dual beta)
- Non-productive rearrangements also occur
  - Challenge or opportunity?

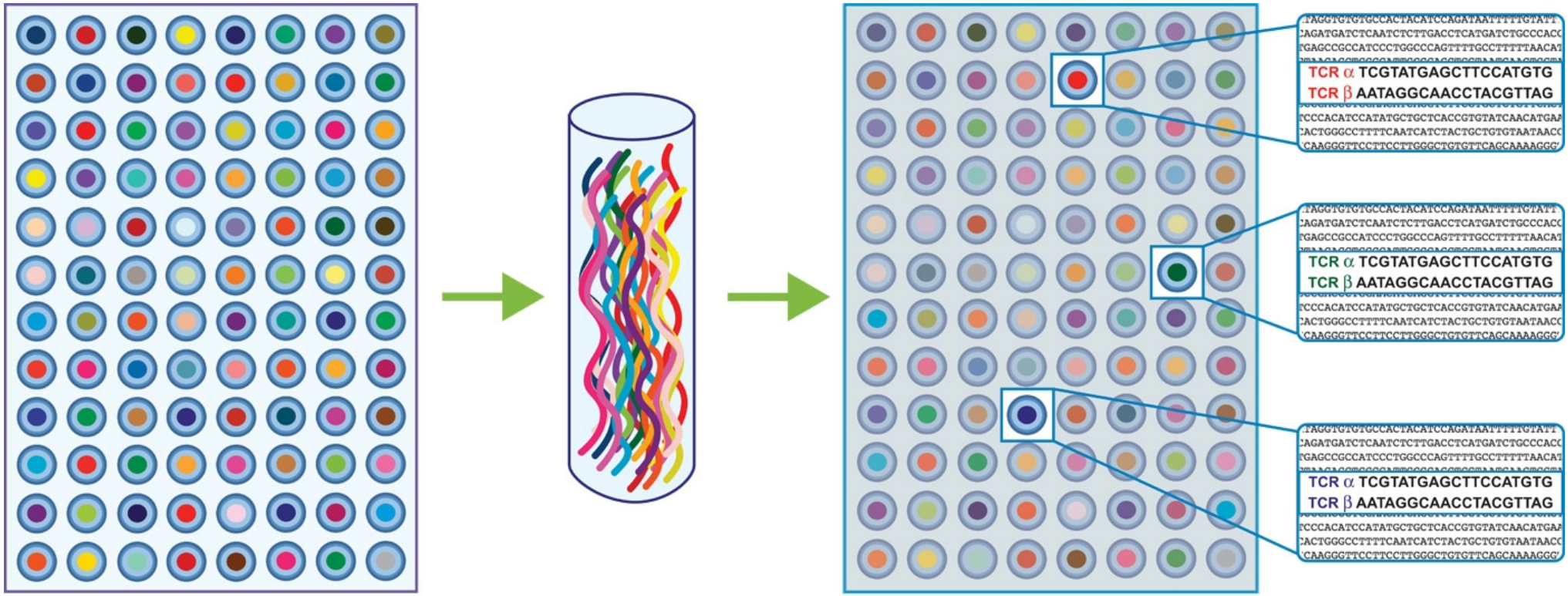
Brady BL, Steinel NC, Bassing CH. Antigen Receptor Allelic Exclusion: An Update and Reappraisal. Journal of immunology (Baltimore, Md : 1950). 2010;185(7):3801-3808. doi:10.4049/jimmunol.1001158.

Sharing recombinants?

- Clones can share a single β-recombinant due to:
  - Convergent recombination
  - Maturation process
- Clones can only share a single α-recombinant due to convergent recombination
- Amount of sharing varies significantly
  - anywhere from 0% to ~50% of α-recombinants
  - anywhere from 0% to ~40% of β-recombinants

Lee ES, Thomas PG, Mold JE, Yates AJ. Identifying T Cell Receptors from High-Throughput Sequencing: Dealing with Promiscuity in TCRα and TCRβ Pairing. PLoS Comput Biol 2017. 13(1): e1005313. https://doi.org/10.1371/journal.pcbi.1005313

PairSEQ



T cells are distributed across a 96-well plate and assigned well-specific barcodes.

Barcoded cDNAs are sequenced together.

TCR pairs are identified by finding sequences that occupy the same wells.

Howie B, Sherwood AM, Berkebile AD, Berka J, Emerson RO, Williamson DW, et al. High-throughput pairing of T cell receptor α and β sequences. Sci Transl Med. 2015;7(301):301ra131. pmid:26290413

Clone Identification Problem (CIP)

- **Input:**
  - observed alpha recombinants $\alpha_1, \alpha_2, \ldots, \alpha_n$
  - observed beta recombinants $\beta_1, \beta_2, \ldots, \beta_m$
  - recombinant well occurrences (i.e. well subsets)

$$W : \{\alpha_1, \ldots, \alpha_n\} \cup \{\beta_1, \ldots, \beta_m\} \rightarrow P(\{1, 2, \ldots, w\})$$

- **Output:**
  - set of identified T cell clones
    - singles, duals, sharing

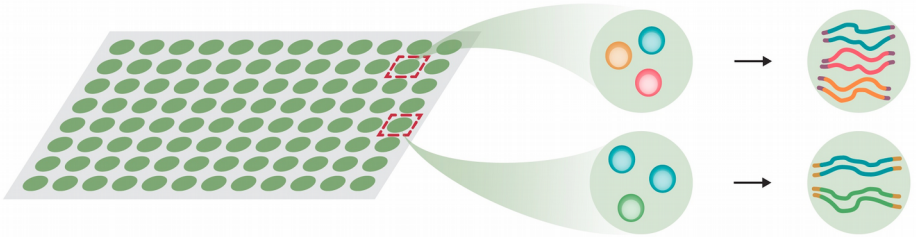
Challenges

0. Pairing “normal” recombinants? (i.e. *singles*)
1. Multiple rearrangements? (i.e. *duals*)
2. Shared recombinants? (i.e. *sharing*)
3. Computational efficiency?
4. Elegance?

ALPHABETR

A T cells are sampled onto 96-well plates at 10-300 cells/well

B cDNA libraries are created from each well with RT-PCR

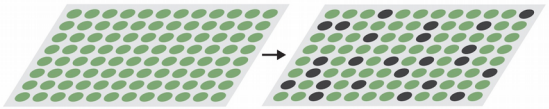


C Unpaired CDR3α and CDR3β are sequenced in each well

α_1 CAVTGGDKLIF	α_1 CAVTGGDKLIF
α_2 CALDGDKIIF	α_3 CAVTYGYLNF
β_1 CASGLARAEQYF	α_4 CALTASGLTF
β_2 CASSEGDKVIF	β_1 CASGLARAEQYF
	β_2 CASSHSRYEQYF
	β_3 CSEVHTARTQYF

D A resampling strategy is used to obtain a list of possible TCR pairs by repeatedly performing steps (i), (ii), (iii)

(i) A subset of wells is randomly selected for steps (ii) and (iii)



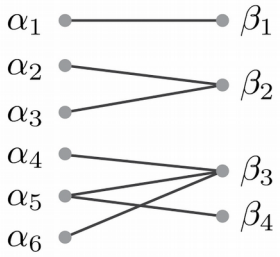
(ii) Association scores are calculated for every α and β chain across all wells in the subset

$$S_{ij} = \sum_{k=1}^{W} \left(\frac{\delta_{ij}^k}{c_\alpha^k} + \frac{\delta_{ij}^k}{c_\beta^k} \right)$$

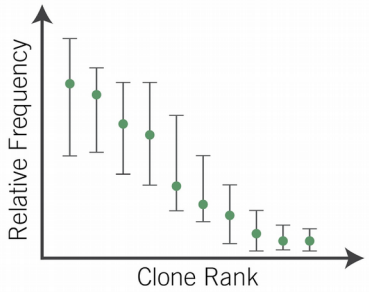
(iii) Scores are used to select likely αβ pairs within each well

	α_1	α_2	α_3	α_4	α_5
β_1	24.0	0.6	1.2	4.3	8.2
β_2	0.4	0.2	60.2	0.7	2.2
β_3	1.0	0.2	1.2	9.0	3.0
β_4	3.2	30.1	0.1	0.4	2.1

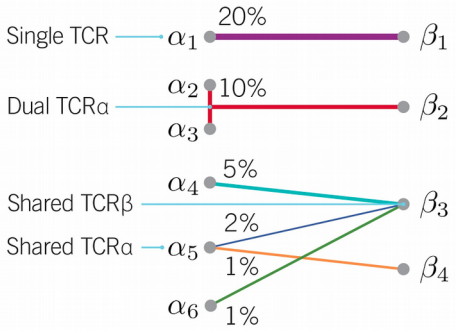
E Pairs from Step D present in more than a threshold proportion of replicates are candidate TCRs



F Clonal frequencies are estimated with maximum-likelihood and used to distinguish β-sharing and dual TCRα

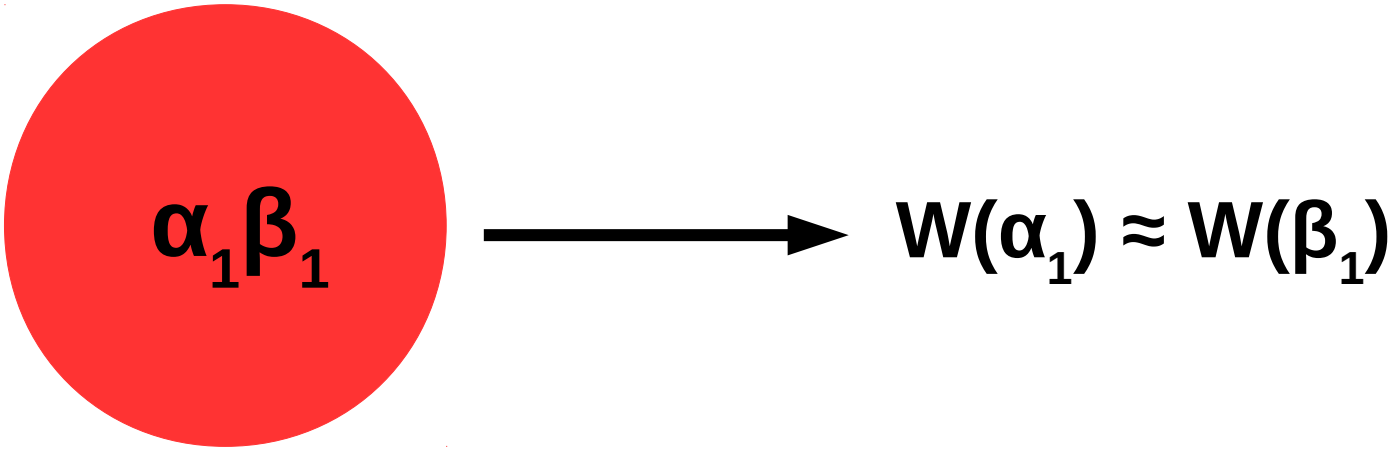


G The output is a list of single and dual TCR clones with their respective clonal frequencies



Lee ES, Thomas PG, Mold JE, Yates AJ. Identifying T Cell Receptors from High-Throughput Sequencing: Dealing with Promiscuity in TCRα and TCRβ Pairing. PLoS Comput Biol 2017. 13(1): e1005313. https://doi.org/10.1371/journal.pcbi.1005313

CHALLENGE 0: Pairing recombinants



- Expect recombinants to appear together in wells more often than they appear apart
- Pair well subsets, **not** recombinants!
  - Much fewer of the former!

“Pair well subsets, not recombinants!”

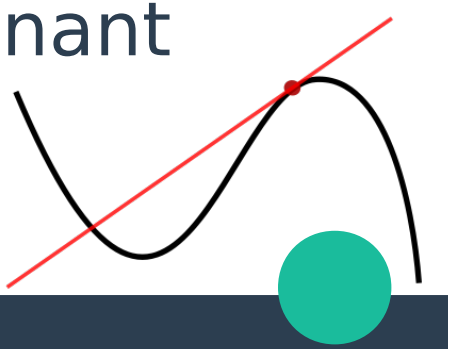
- Minor tangent:
  - map alpha well subsets to alpha recombinants

$$A : P(\{1, 2, ..., w\}) \rightarrow P(\{\alpha_1, ..., \alpha_n\})$$

  - map beta well subsets to beta recombinants

$$B : P(\{1, 2, ..., w\}) \rightarrow P(\{\beta_1, ..., \beta_m\})$$

- If `|A(s)|=1`, we say the well subset `s` is (alpha) ***unique***, and the corresponding recombinant `A(s)`is ***uniquely identifiable***



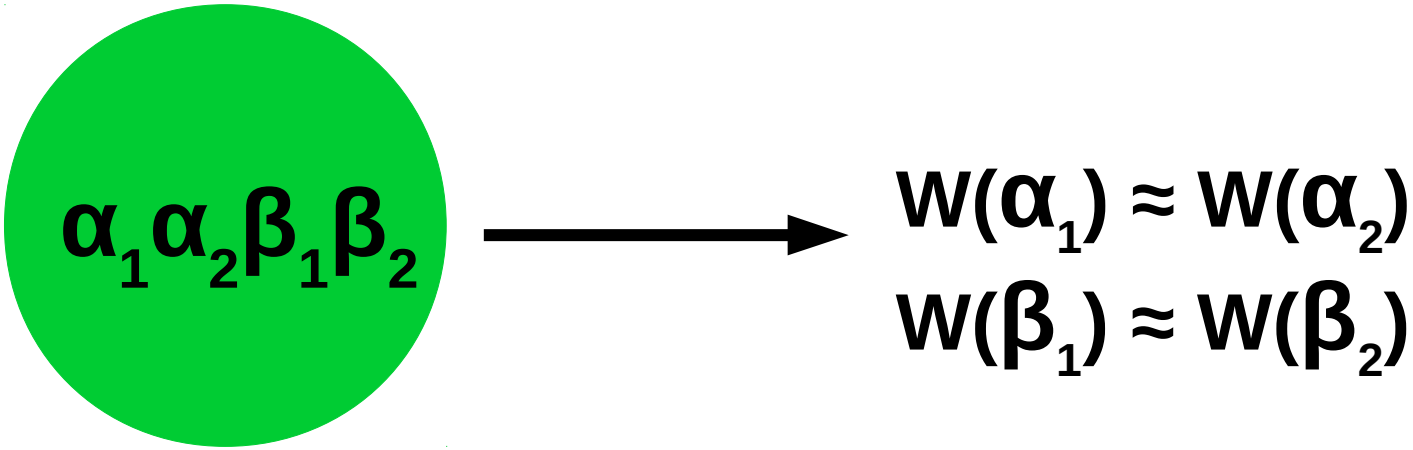
CIP Algorithm v0.0

```
Function CLONEIFY(A,B):
    C = PAIR(A,B)
    Return C

Function PAIR(A,B):
    Return { (a_i,b_j) : a_i ≈ b_j, a_i ∈ A, b_j ∈ B }
```

To define later...

CHALLENGE 1: Multiple rearrangements



- Modeling rearrangements is **easy**!
- Natural imputation of well sets

$$(a_1 \text{ U } a_2 : (a_1,a_2) \in \textbf{PAIR}(A,A))$$

$$(b_1 \text{ U } b_2 : (b_1,b_2) \in \textbf{PAIR}(B,B))$$

CIP Algorithm v1.0

Function **CLONEIFY**(A,B):

A_D = **PAIR**(A,A)

B_D = **PAIR**(B,B)

A′ = **AUGMENT**(A,A_D)

B′ = **AUGMENT**(B,B_D)

C = **PAIR**(A′,B′)

Return C

Function **AUGMENT**(W,W_D):

W′ = W

$W'((w_1,w_2)) = (W(w_1), W(w_2)), \forall (w_1,w_2) \in W_D$

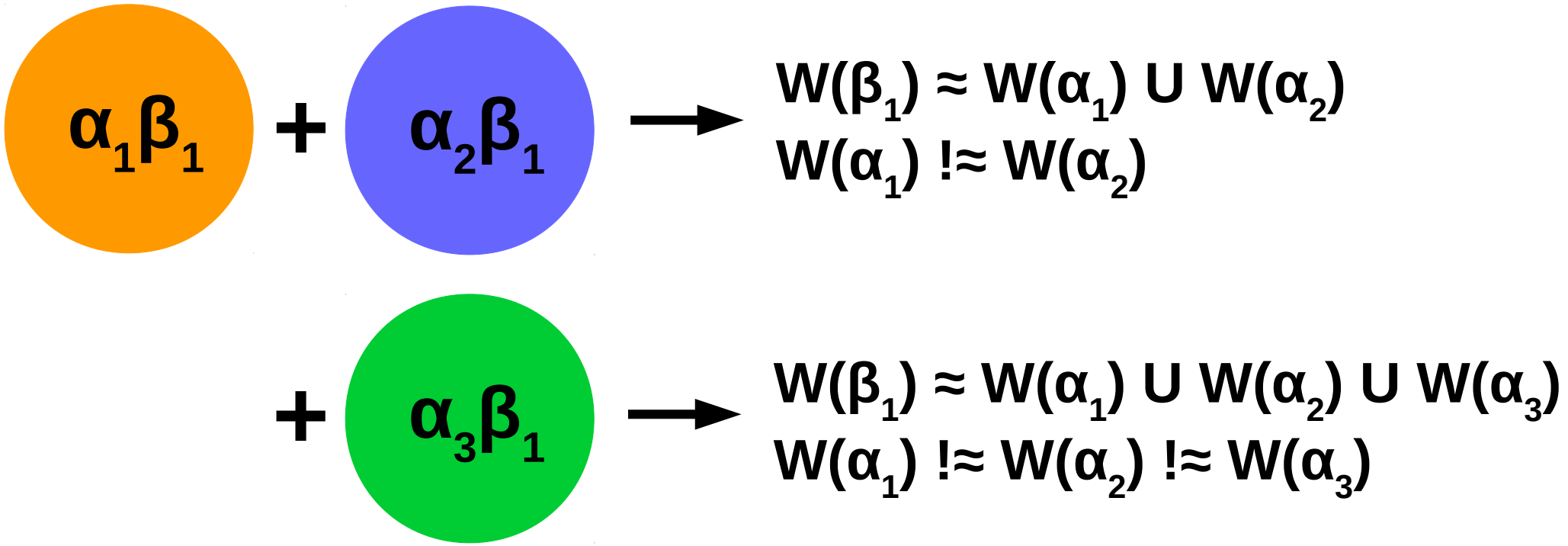
Return W′

Note:

$|W'((w_1,w_2))|=1$ if

$|W(w_1)|=1$ and $|W(w_2)|=1$

CHALLENGE 2: Shared recombinants



- Modeling sharing is *hard*!
- *Compromise*: check all pairs?
  - Cubic running time?? (not really)

CIP Algorithm v2.0

```
Function CLONEIFY(A,B):
    A_D = PAIR(A,A)
    B_D = PAIR(B,B)
    A' = AUGMENT(A,A_D)
    B' = AUGMENT(B,B_D)
    C = PAIR(A',B')
    C' = SHAREIFY(A',B',C)
    Return C'
```

```
Function SHAREIFY(A',B',C):
    C' = C
    For a in A':
      For b_1,b_2 in C with a:
        If b_1 and b_2 relatively
            random && a ≈ b_1 U b_2:
          C' = C' U {(a,b_1),(a,b_2)}
    Return C'
```

CHALLENGE 3: Computational efficiency?

- Need efficient implementation of `PAIR(A,B)`

CHALLENGE 3: Computational efficiency!

- Implement `PAIR(A,B,W)` using locality-sensitive hashing (LSH)
  - with Jaccard similarity coefficient (i.e. MinHashing)

$$SIM(a,b) = \frac{a \cap b}{a \cup b}$$

  - `a ≈ b` then means `SIM(a,b)` is greater than some defined threshold
  - time almost linear in the sizes of `A` and `B`
- Try b-matching to put global constraint on pairing?
  - `O(e·(n+e)log(U)log(n))` by reducing to a min-cost max-flow and using Capacity Scaling algorithm

'Greedy' vs *b-matching* CIP Solvers

```
Function CLONEIFY(A,B):
    A_D = PAIR(A,A)
    B_D = PAIR(B,B)
    A' = AUGMENT(A,A_D)
    B' = AUGMENT(B,B_D)
    C = PAIR(A',B')
    C' = SHAREIFY(A',B',C)
    Return C'
```

'Greedy'

```
Function CLONEIFY(A,B):
    A_D = PAIR(A,A)
    B_D = PAIR(B,B)
    A' = AUGMENT(A,A_D)
    B' = AUGMENT(B,B_D)
    E = PAIR(A',B')
    C = BMATCH(A',B',E)
    C' = SHAREIFY(A',B',C)
    Return C'
```

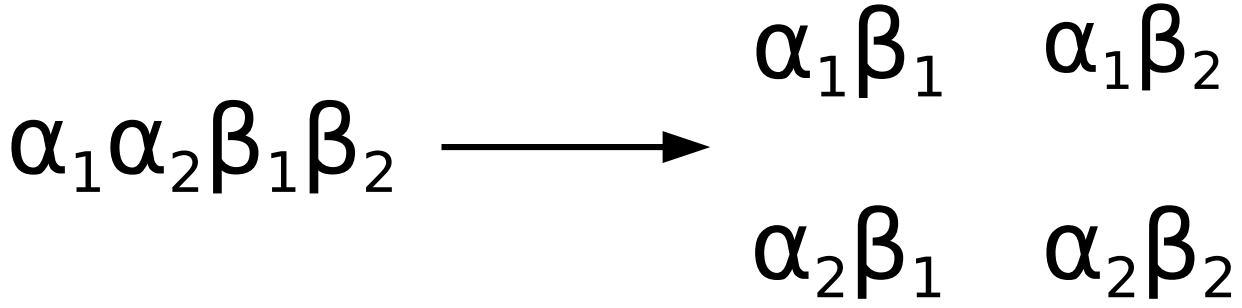
b-matching

CHALLENGE 4: Elegance?

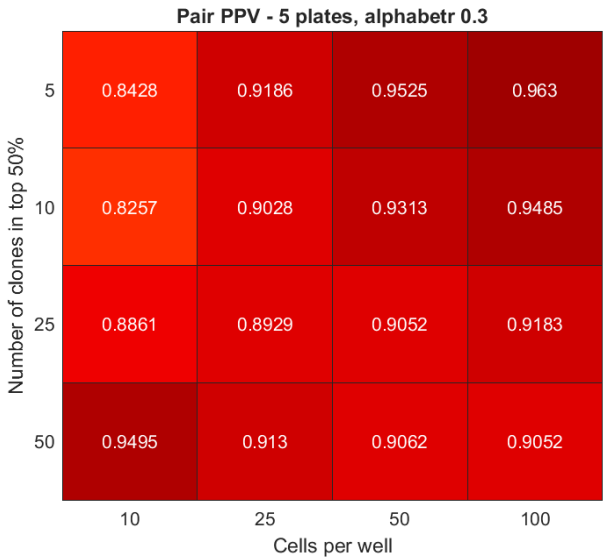
- Kind of!
  - Simple but comprehensive model for identifying singles and duals
  - Sharing identification needs improvement...
- But how well does it all work?

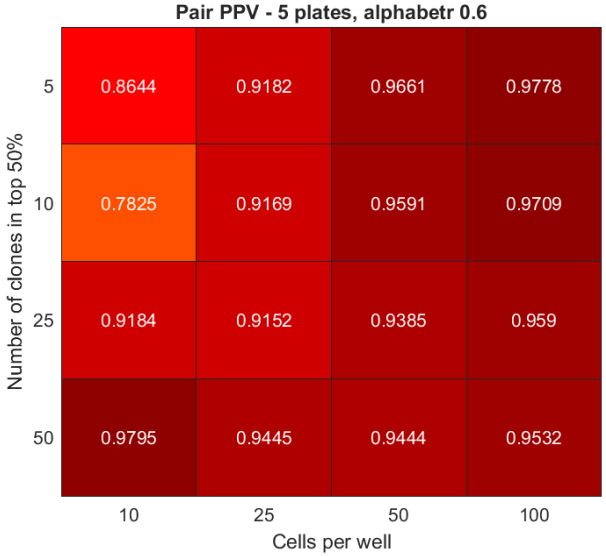
Results

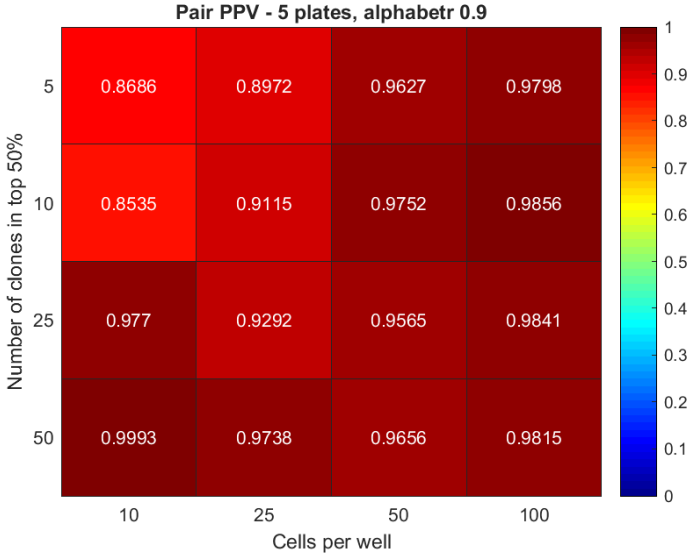
- ALPHABETR simulator
  - 2000 clones
  - 30% dual alpha, 6% dual beta
  - same sharing distribution
  - average of 50 runs
- Clone performance vs pair performance

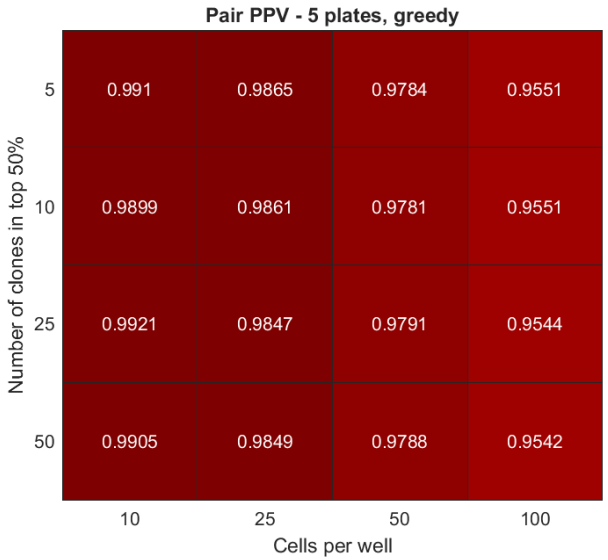


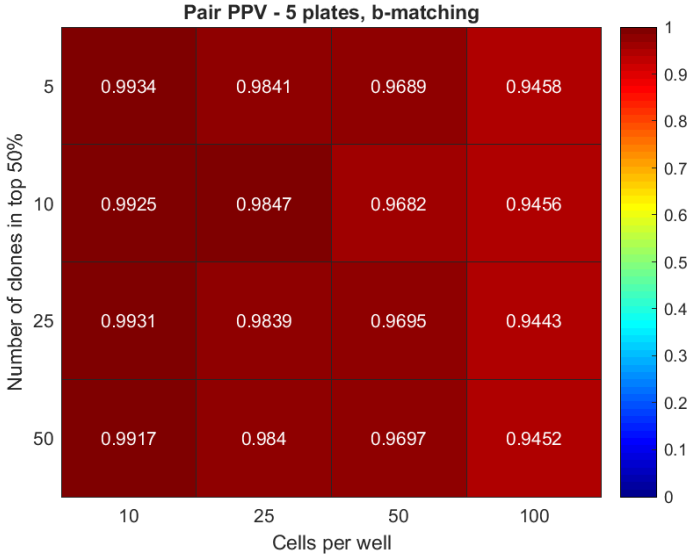
Pair positive predictive value (PPV)





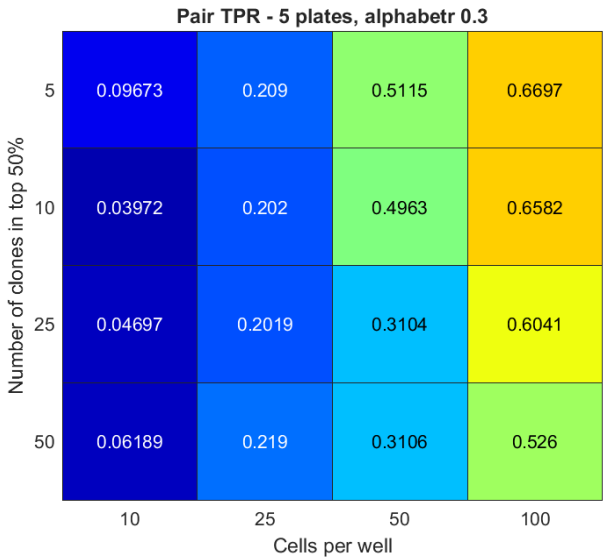


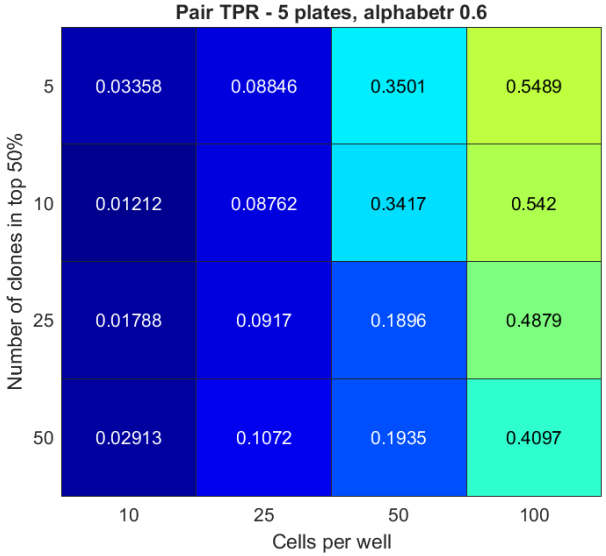


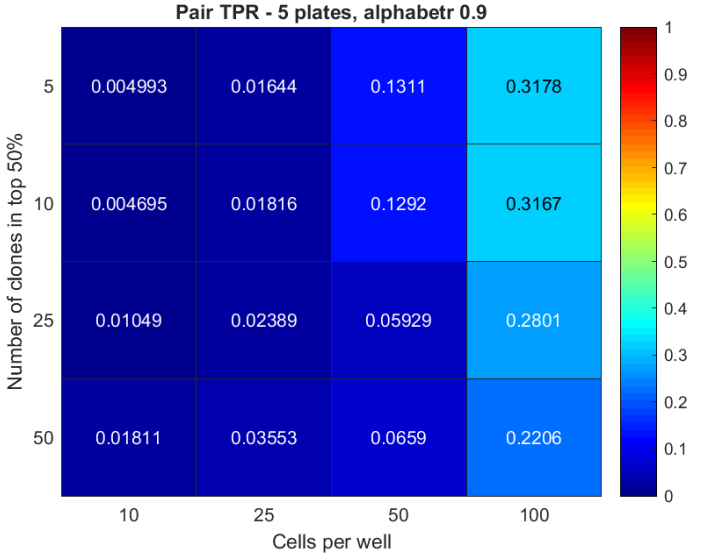


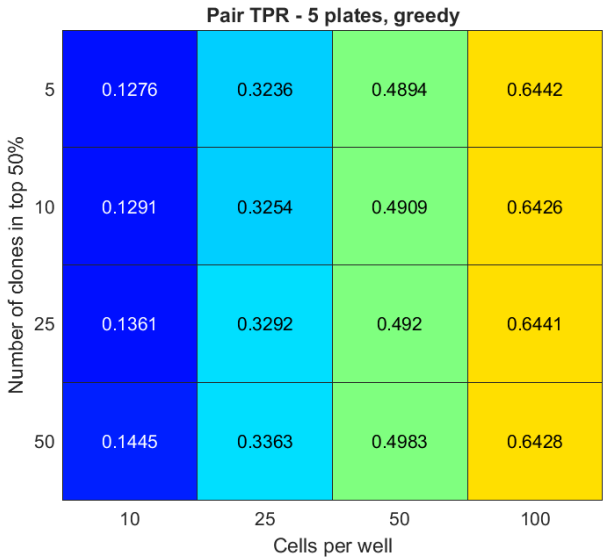
$$PPV = \frac{TP}{TP + FP}$$

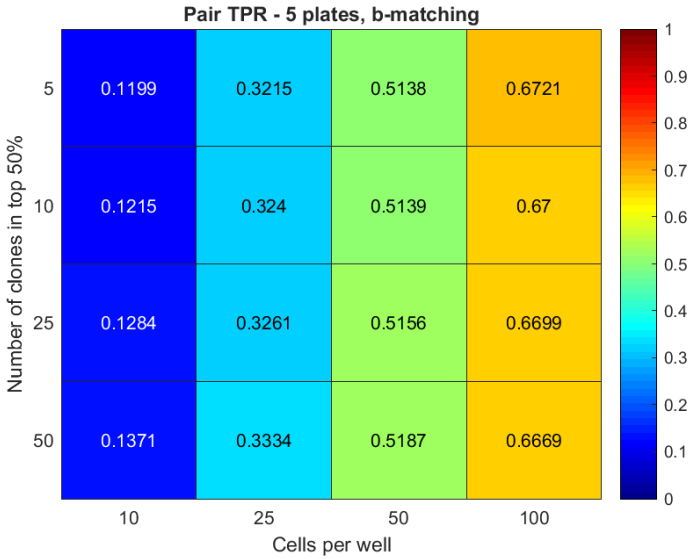
Pair true positive rate (TPR)





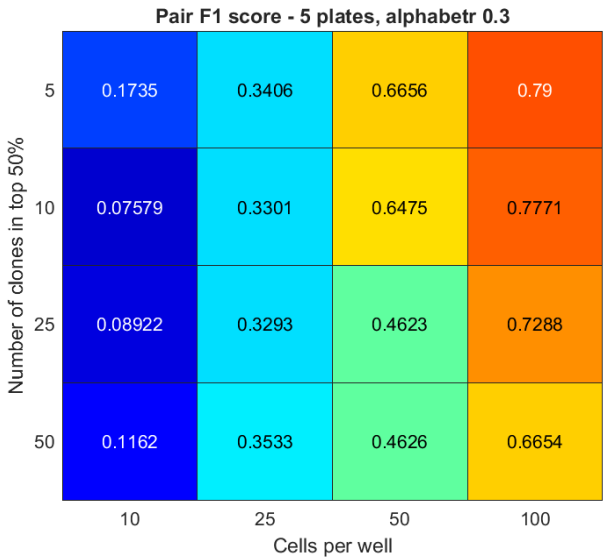


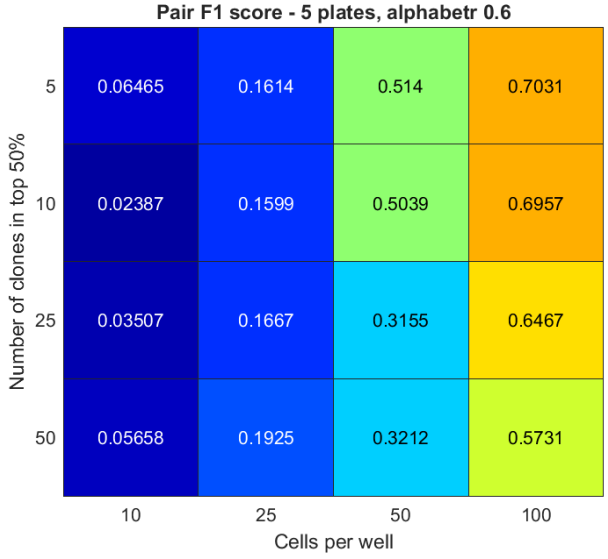


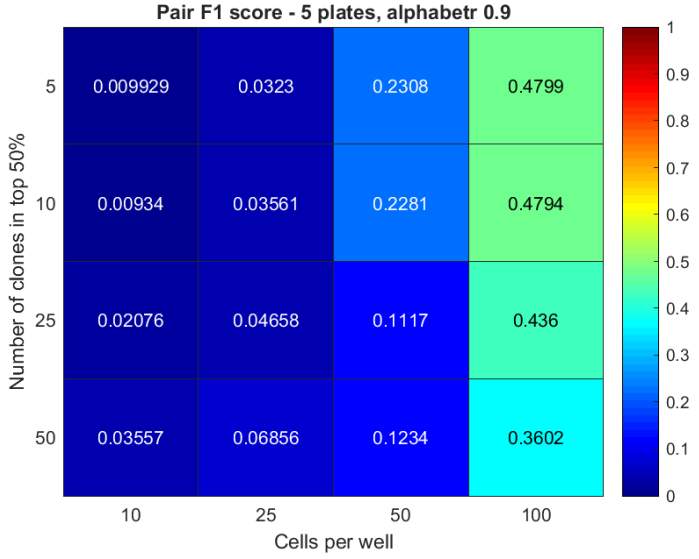


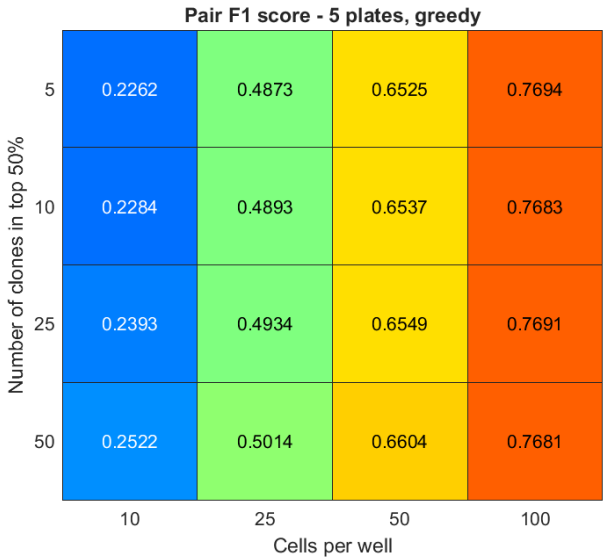
$$TPR = \frac{TP}{TP + FN}$$

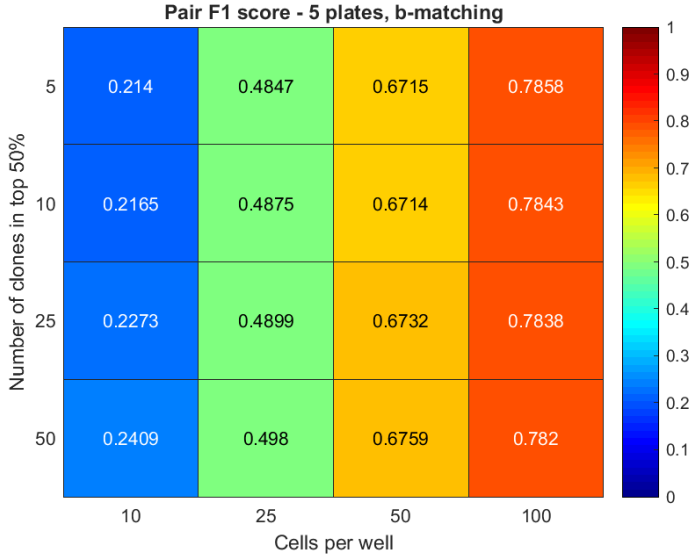
Pair F1 score





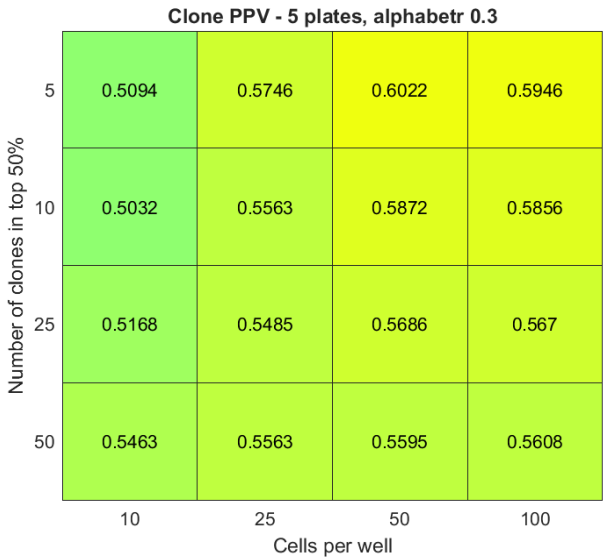


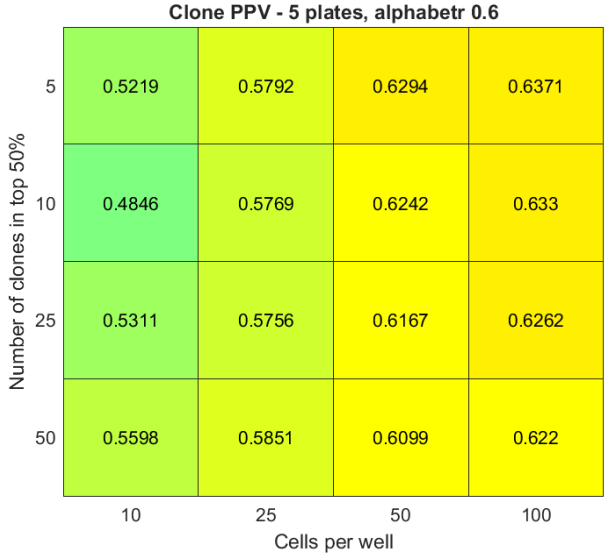


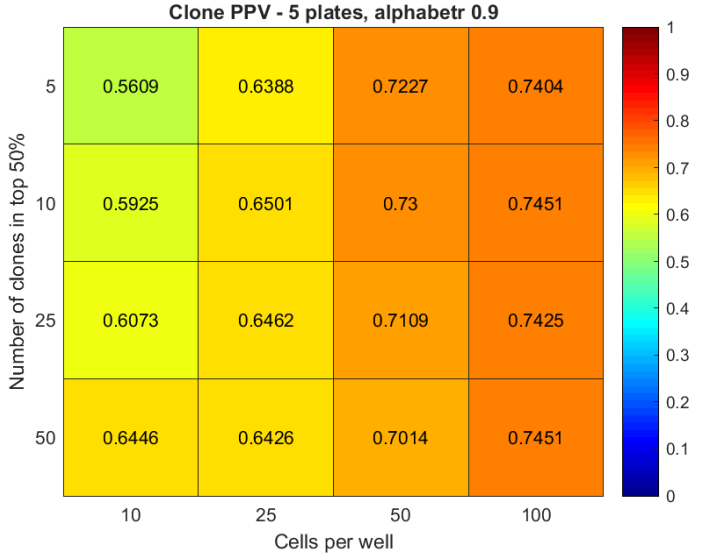


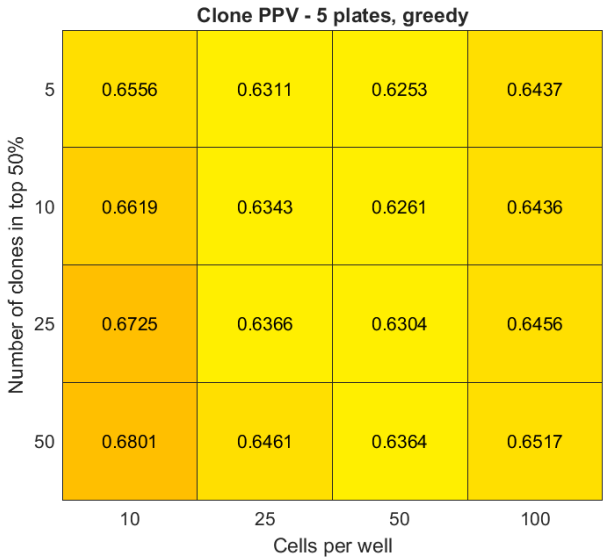
$$F1 = \frac{2 \cdot PPV \cdot TPR}{PPV + TPR}$$

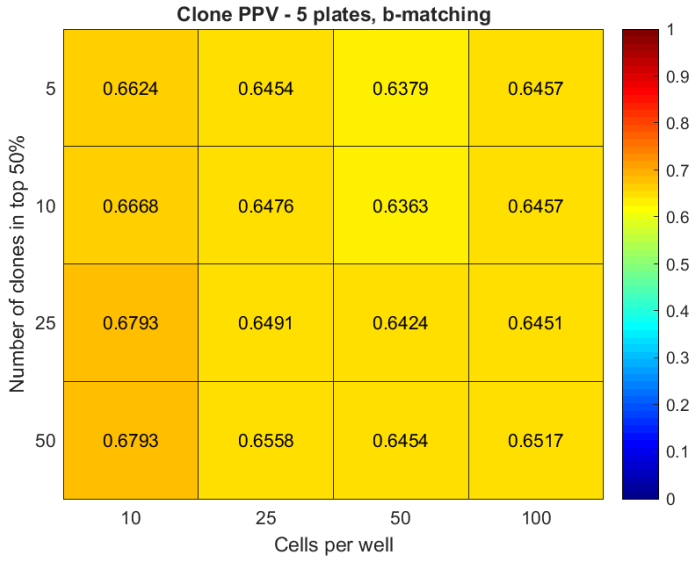
Clone positive predictive value (PPV)





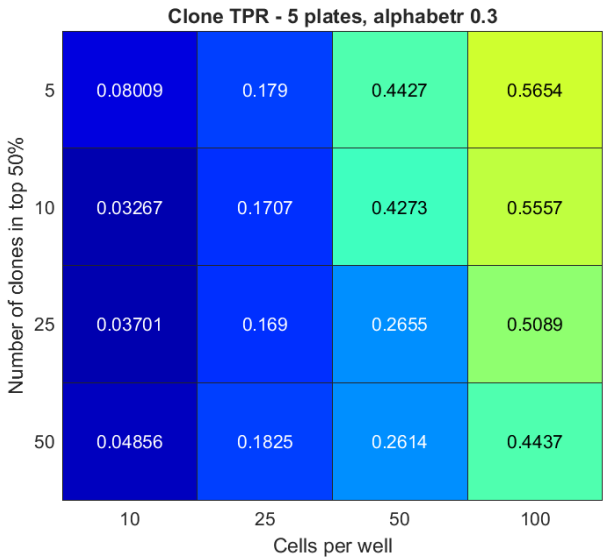


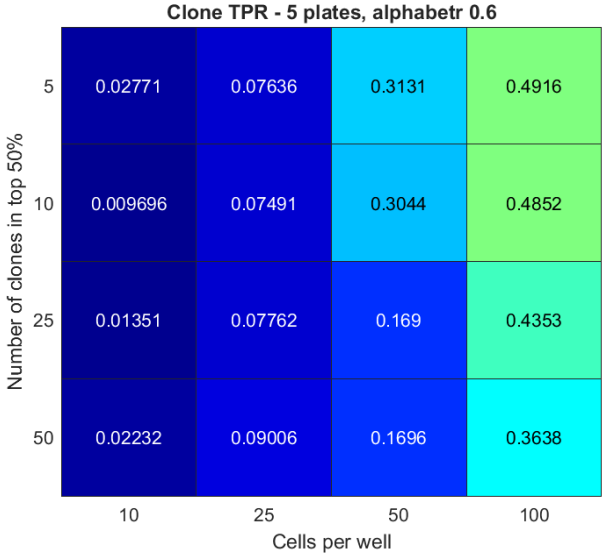


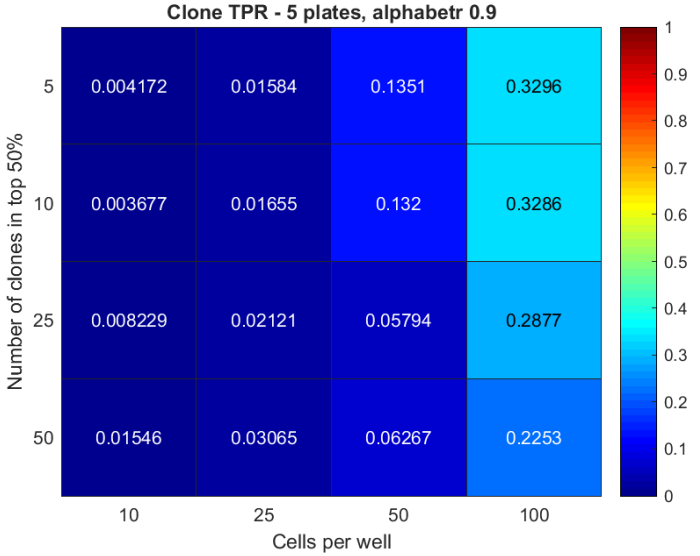


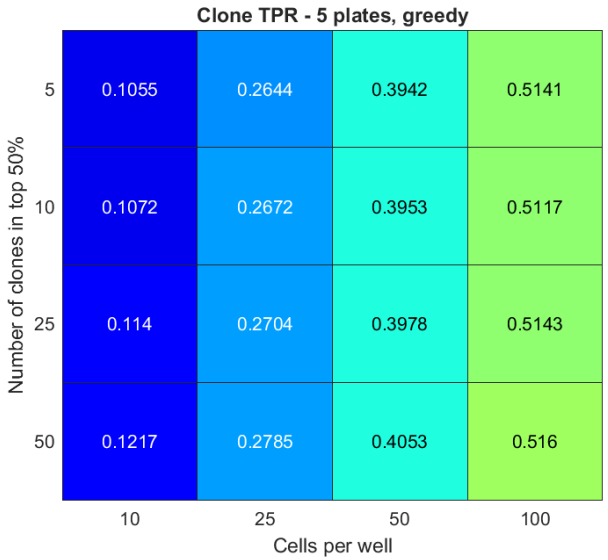
$$PPV = \frac{TP}{TP + FP}$$

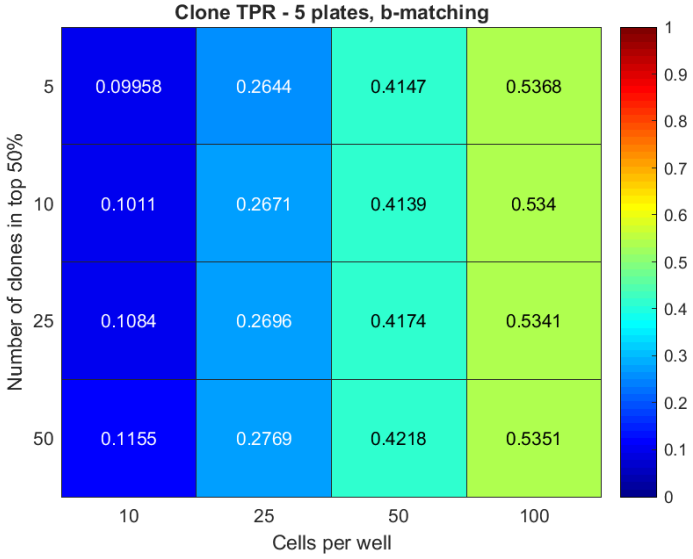
Clone true positive rate (TPR)





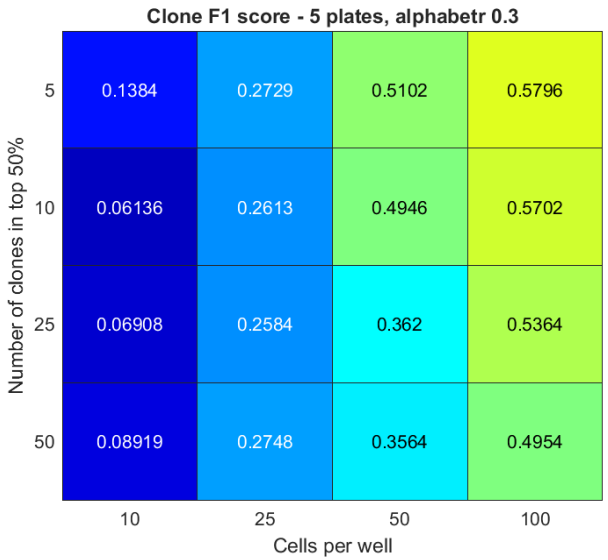


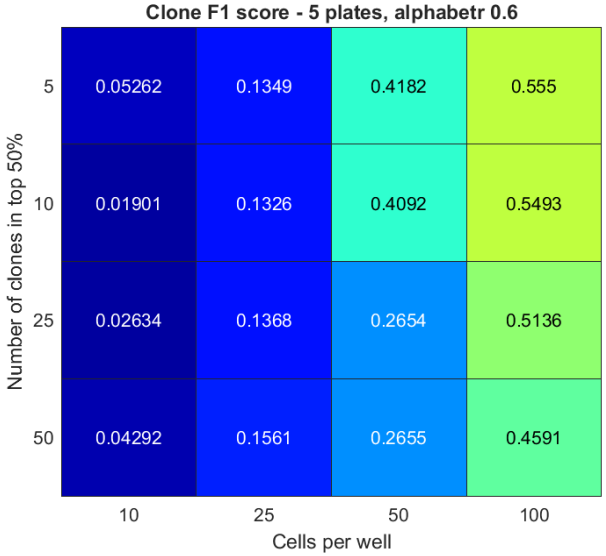


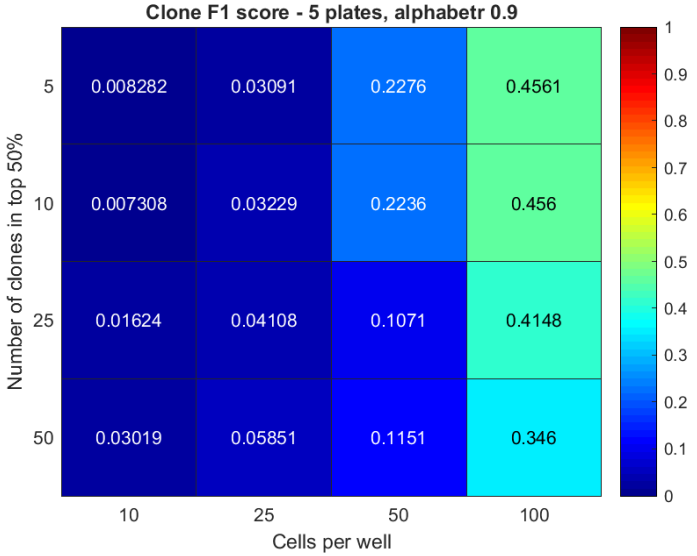


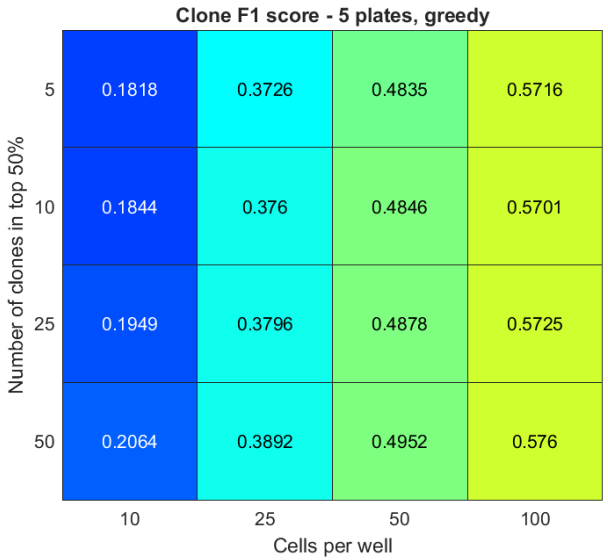
$$TPR = \frac{TP}{TP + FN}$$

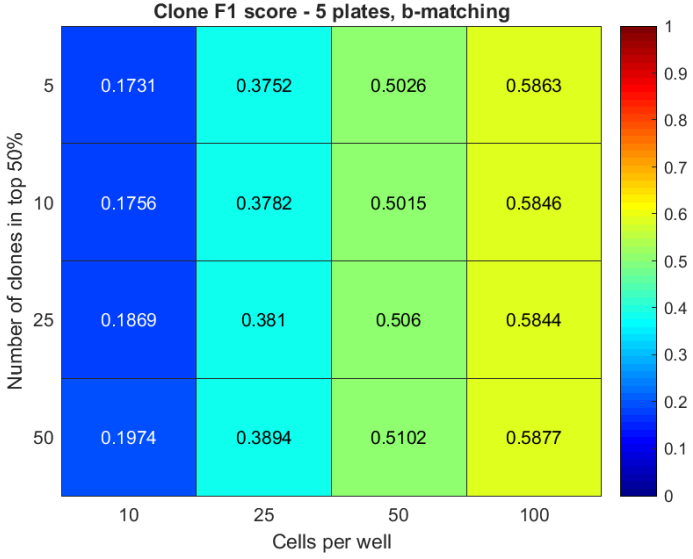
Clone F1 score











$$F1 = \frac{2 \cdot PPV \cdot TPR}{PPV + TPR}$$

Ongoing and future work

- Integrate MinHashing with solvers
- Run scaling experiments to get a concrete sense of how runtime grows
- Test solvers against real data

Acknowledgement(s)

- Thanks Ion!

Questions?