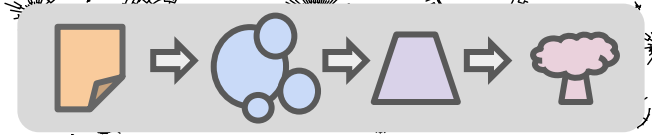
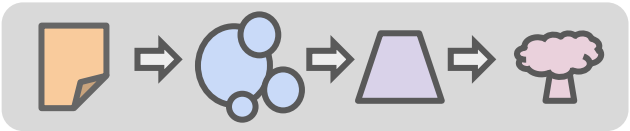
Reconstruction and analysis of B-cell lineage trees from single cell data using Immcantation

Kenneth B. Hoehn
Susanna Marquez
Noah Yann Lee
Kleinstein Lab
Department of Pathology
Yale School of Medicine



Outline of tutorial

1. Immcantation setup.
2. B cell phylogenetics background.
3. Combining gene expression and BCR sequences.
4. Identifying clonal clusters, reconstruct germlines.
5. Building and visualizing trees.
6. Tree analysis, detecting ongoing evolution.



Immcantation: A start to finish suite for BCR analysis

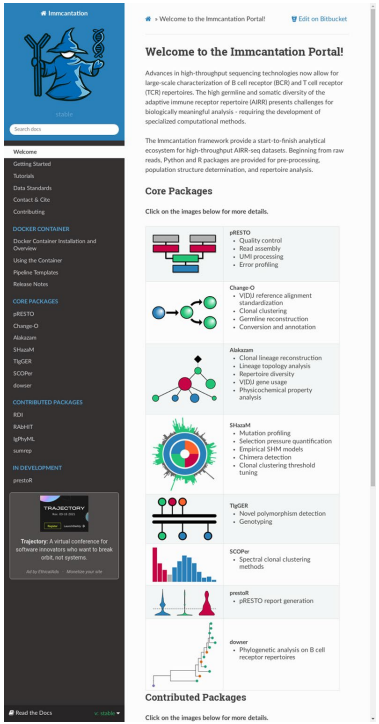
Start to finish suite of programs for adaptive immune (AIRR) repertoires
Focus on BCR, possible to work with TCR: immcantation.org
Modular, actively maintained and adapted to new data types
User help immcantation@googlegroups.com

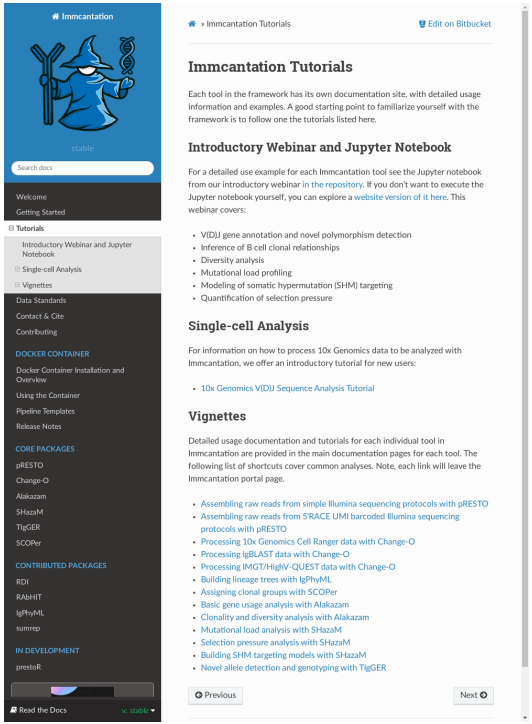
AIRR-C sw-tools v1 compliant

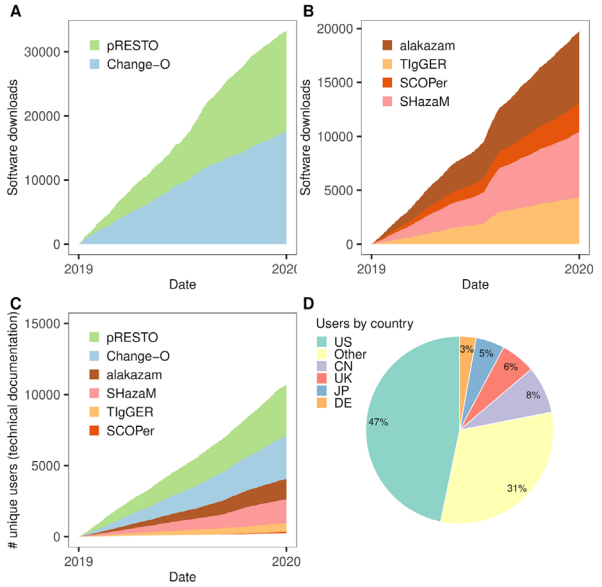
R Packages used in tutorial

alakazam (Core)	alakazam.readthedocs.io
shazam (SHM)	shazam.readthedocs.io
scoper (Clones)	scoper.readthedocs.io
dowser (Trees)	dowser.readthedocs.io

Immcantation history







immcantation.org

Two routes to Immcantation

Docker Container (first install Docker)

```
docker pull immcantation/suite:<version>

# Start interactive session,
# bind ~/my-folder/ to /data,
# use /data as working dir
docker run -it --workdir /data -v ~/my-
    folder:/data:z \
kleinstein/immcantation:<version> bash
```

Native installation

```
Install all tools and dependencies (software, databases…)

# Install Python packages from PyPI
pip3 install presto --user
pip3 install changeo --user

# Install R packages from CRAN
# Helper packages
install.packages("ggplot2")
install.packages("dplyr")
install.packages("devtools")
install.packages("Seurat")
if (!requireNamespace("BiocManager", quietly = TRUE))
      install.packages("BiocManager")
BiocManager::install("ggtree")

# Immcantation packages
install.packages("alakazam")
install.packages("shazam")
install.packages("scoper")
install.packages("dowser")

# Download muscle, usearch and phylip and
# place them in your PATH
# Example, in Fedora, edit ~/.bashrc to add
# export PATH=$PATH:/path/to/software
```

Instructions: Immcantation.org
Help: immcantation@googlegroups.com

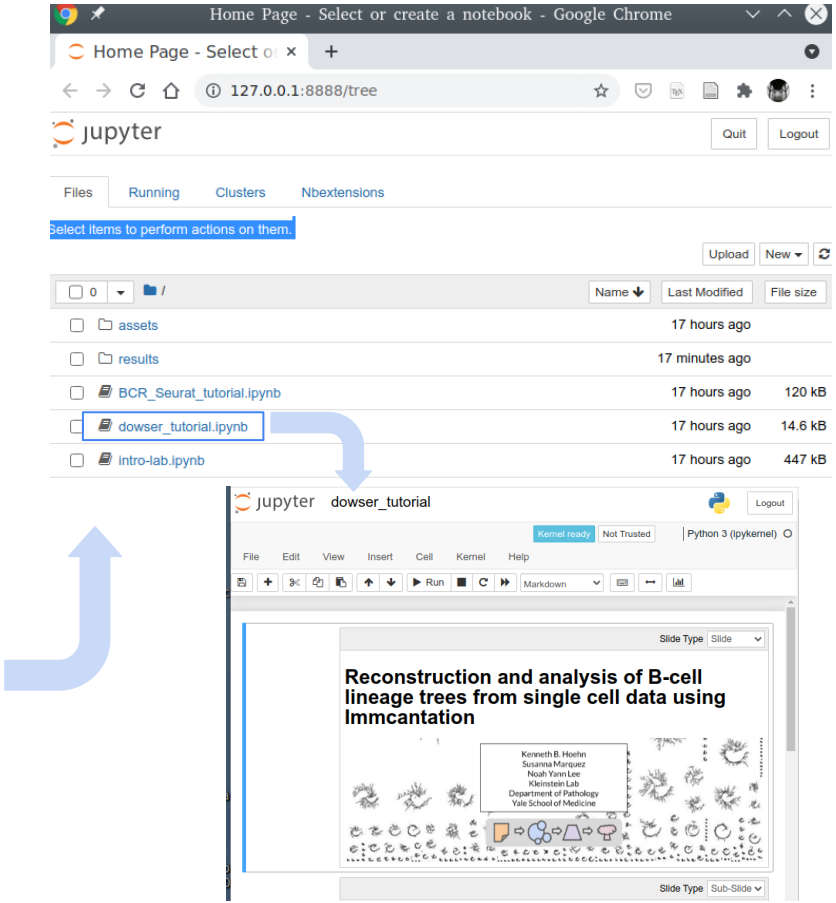
Immcantation training container and notebooks

launch binder

https://mybinder.org/v2/gh/immcantation/immcantation-lab/master

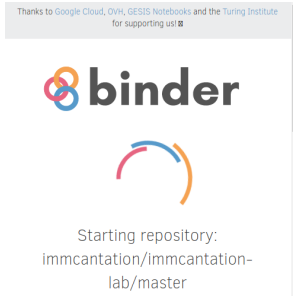
```
# Pull the training container
docker pull immcantation/lab:devel

# Run the container
# Linux/Mac OS X (may need sudo)
docker run --network=host  -it --rm -v
    $(pwd):/home/magus/notebooks/result
    s:z -p 8888:8888
    immcantation/lab:devel
```

"To access the notebook, open this file in a browser or copy and paste one of these URLs".



The notebooks are available online:
https://bitbucket.org/kleinstein/immcantation/src/master/training/

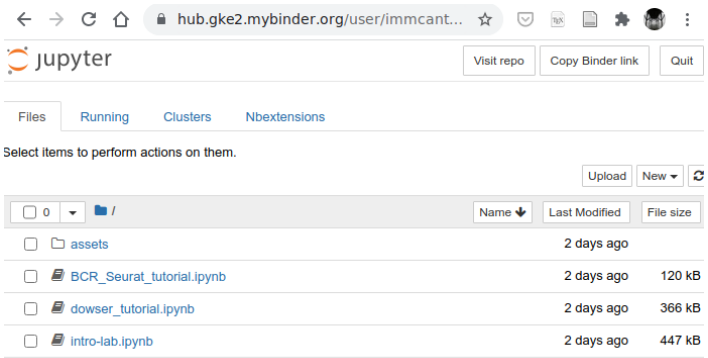
Hands on



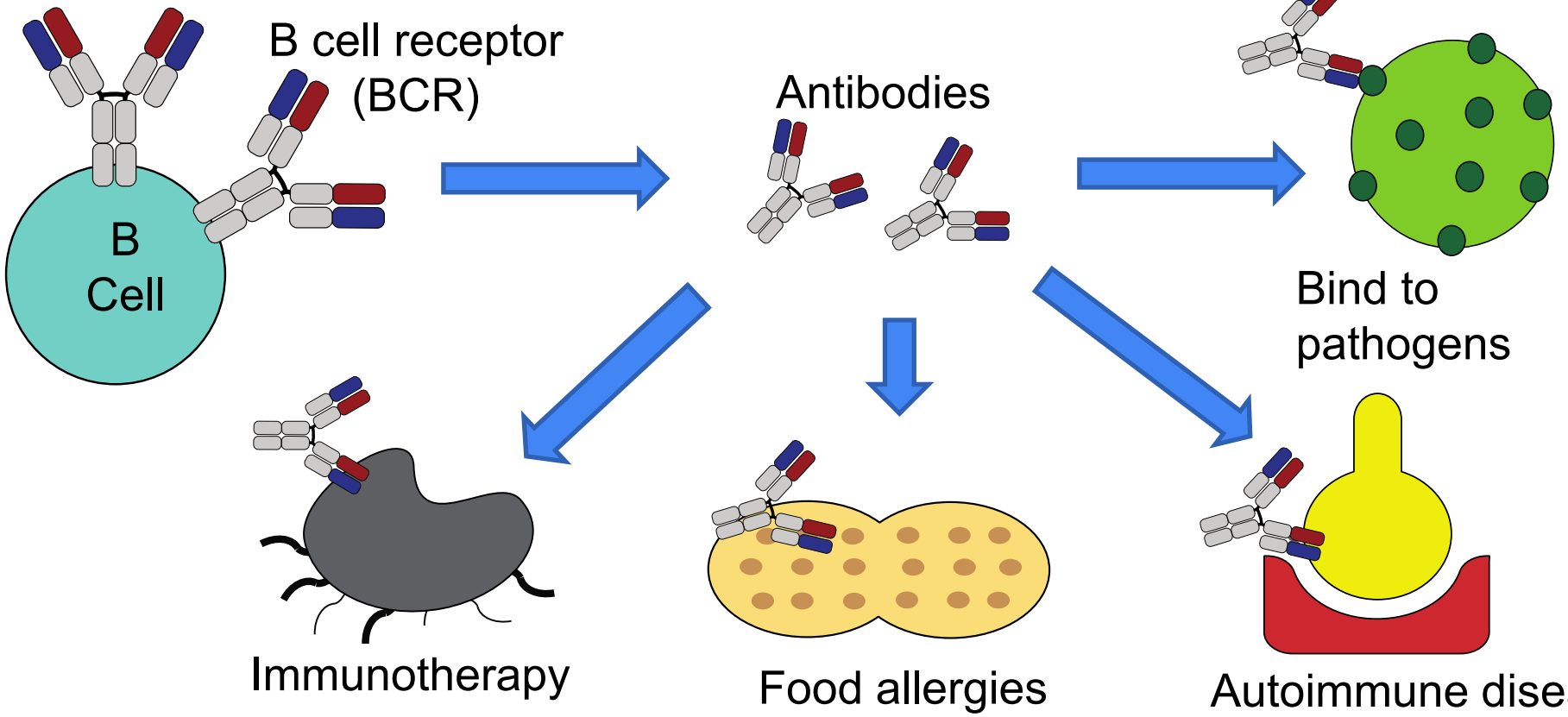
Go to https://mybinder.org/v2/gh/immcantation/immcantation-lab/master

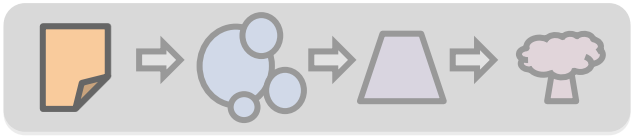
Let the site load in the background, in another browser window, while Ken continues with the presentation.

Open the file dowser_tutorial.ipynb.



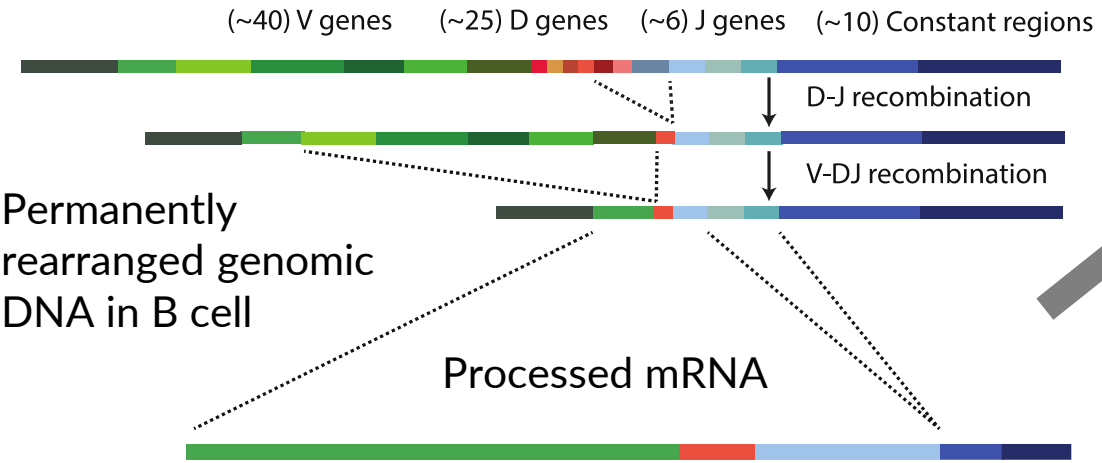
B cells underlie both immune function and pathology

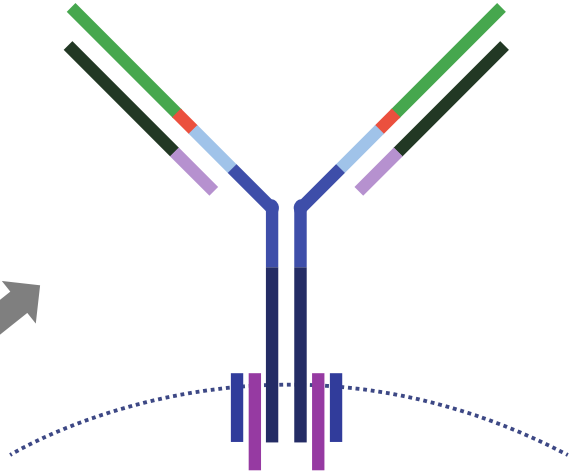




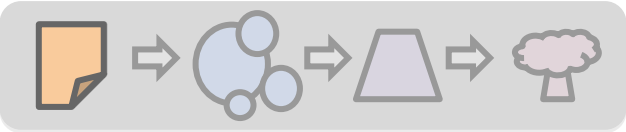
BCRs are first produced by random recombination

Immunoglobulin heavy chain locus during B cell development





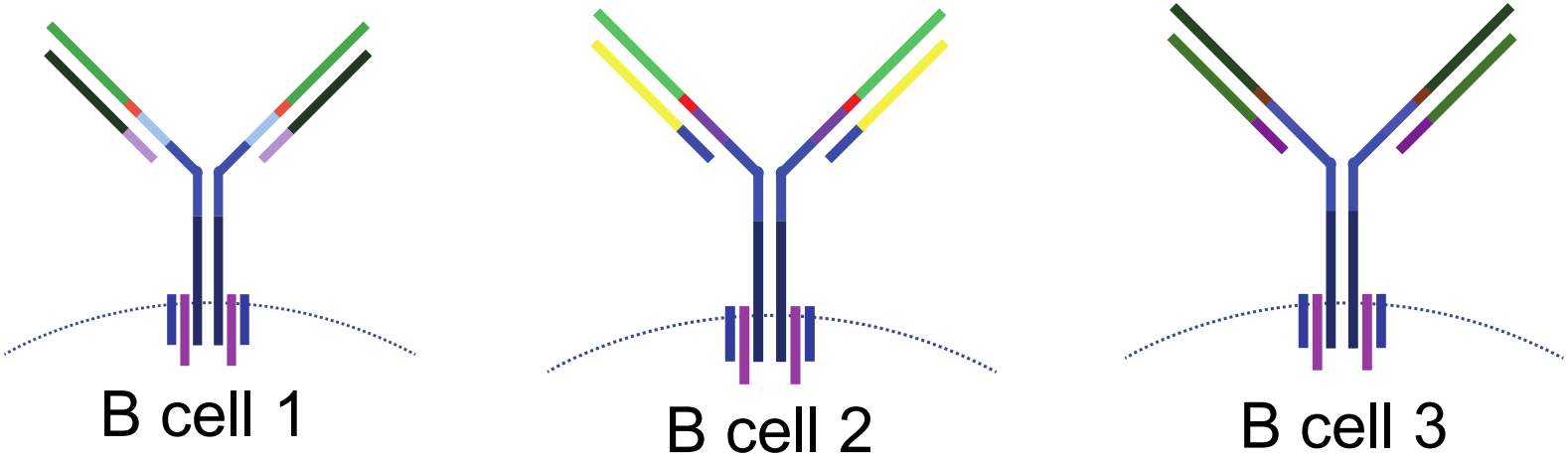
Each new B cell is a stochastic guess in antibody space



Each B cell has a single type of receptor

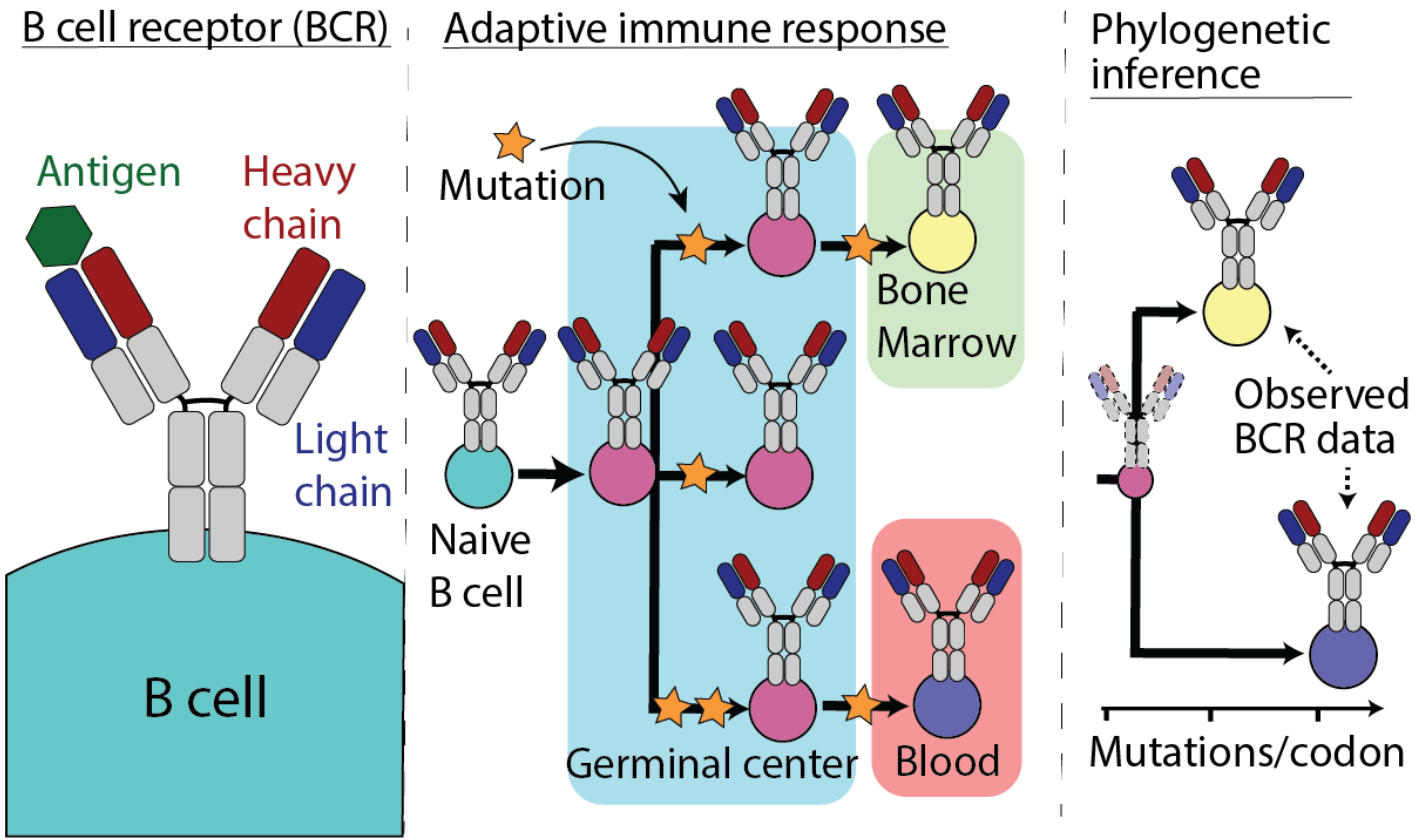
VDJ recombination repeats for each new B cell.
Produces large diversity of “naïve” B cells.
Each expresses a different receptor structure.

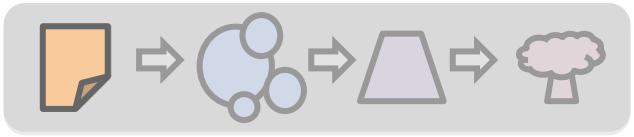
~10^{11} B cells in body
>10^{14} Possible receptors



The repertoire of B cell receptors can recognize a vast array of proteins

B cell affinity maturation



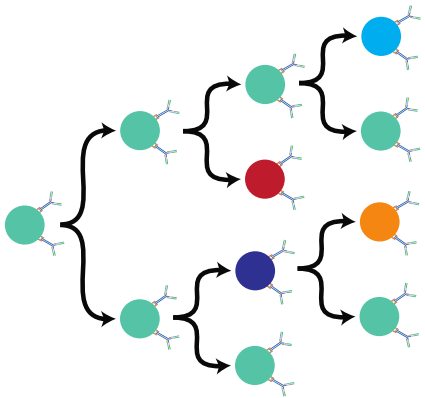


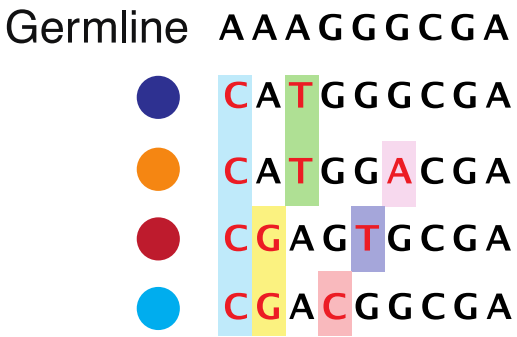
B cell phylogenetic inference

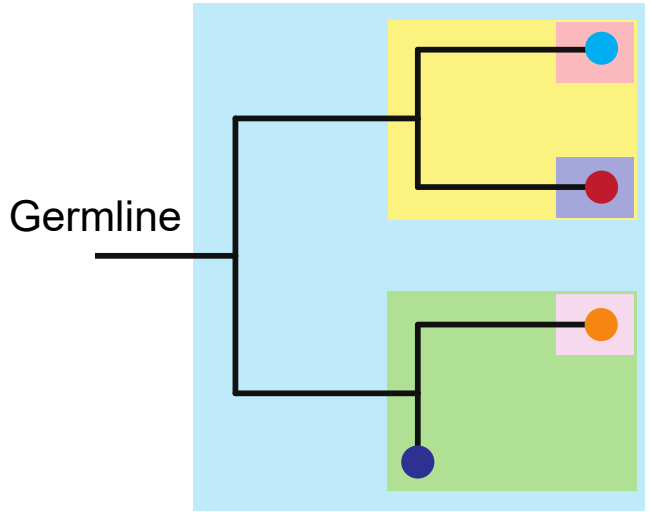
Somatic hypermutation

Sample and sequence cells

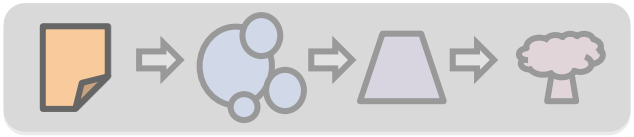
Infer phylogeny





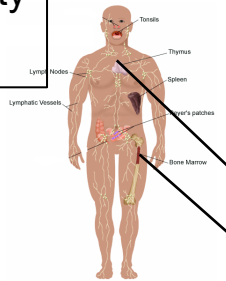


Hierarchy of mutations allows reconstruction of mutation history



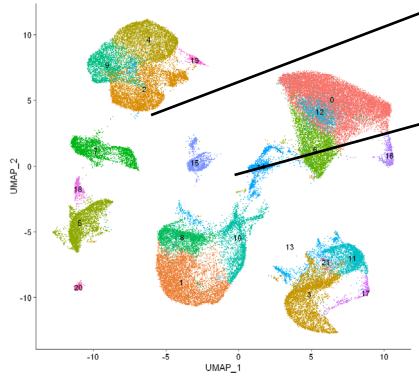
Trees link sources of B cell diversity

Immunological connectivity between tissues



Tips are observed BCRs, branch lengths represent somatic hypermutations

Cell type transitions



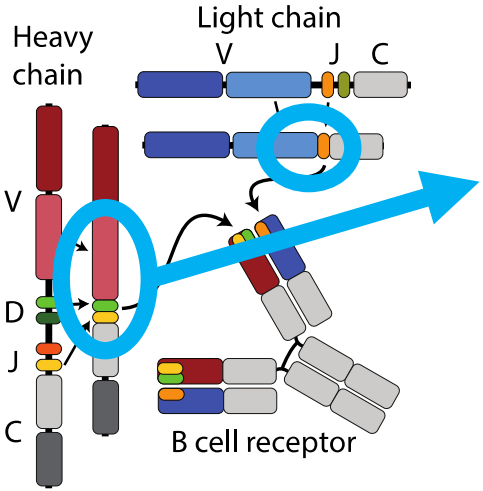
Persistence or re-activation

Time

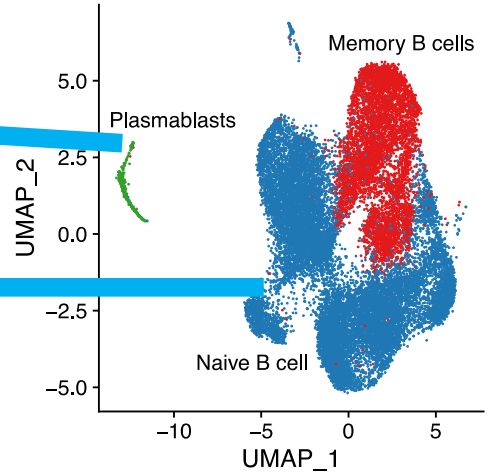
Adaptive Immune Receptor Repertoire (AIRR) sequencing

Sequence V(D)J region. Millions of reads (bulk), thousands of cells (single cell).
AIRR-seq be paired with single-cell RNAseq to include gene expression information

B cell development



>V4-59
TCGGAGACCCTGTCCCTCACCTGCACT
>442_92
TCGGGGACCCTGTCTCTCACCTGCAGT
>193_92
TCGGGGACCCTGTCCCTCACCTGCAGT



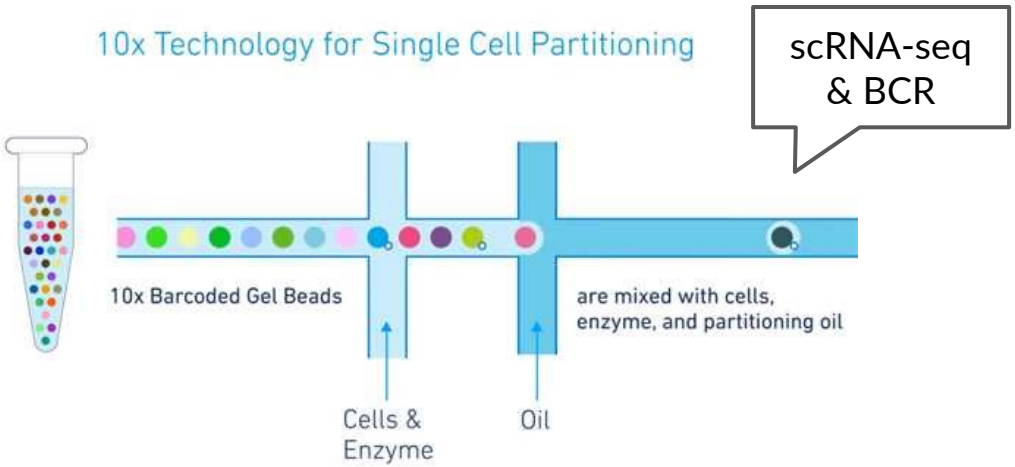
BCR sequencing provides snapshot of B cell diversity within a sample

Read in data to R session

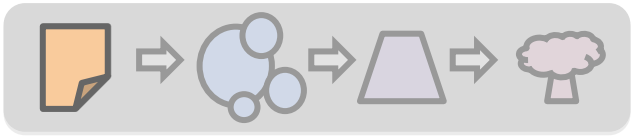
```
library(alakazam)
library(Seurat)

bcr_db <- readChangeoDb("BCR.data.tsv")
gex_db <- readRDS("GEX.data.rds")
```

B-Cell Receptor Data

Gene Expression Data



Turner, J.S., Zhou, J.Q., Han, J. et al. **Human germinal centres engage memory and naive B cells after influenza vaccination.** Nature 586, 127–132 (2020).



What's in the box?

Seurat object (gene expression data)
RNA-seq data already processed and annotated.
For examples:
https://satijalab.org/seurat/articles/pbmc3k_tutorial.html

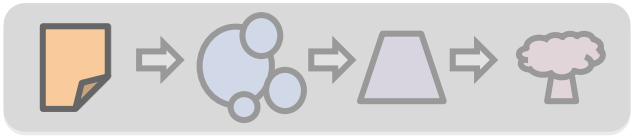
Adaptive Immune Receptor Repertoire (AIRR) tsv
BCRs already aligned to IMGT V, D, and J genes
For instructions:
https://immcantation.readthedocs.io/en/stable/tutorials/tutorials.html

```
library(Seurat)

# Object summary
print(gex_db)

An object of class Seurat
18989 features across 3865 samples within 1 assay
Active assay: RNA (18989 features, 1726 variable features)
 2 dimensional reductions calculated: pca, umap

# Cell type annotations
head(Idents(gex_db),1)

P05_FNA_12_Y1_TCACAAGTCAAACAAG
                        CD4 T
Levels: CD4 T Naive B CD8 T DC/Monocyte GC B NK RMB PB
```

Cell ID

Annotation

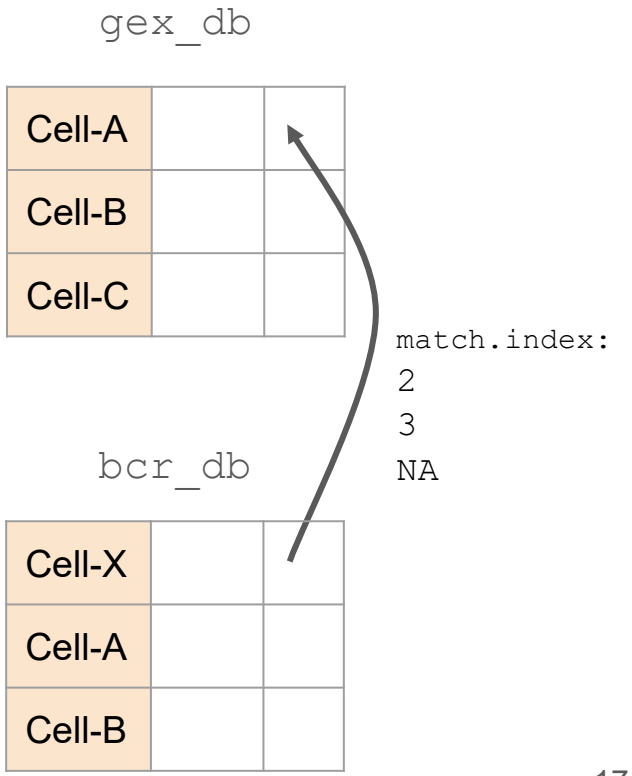
```
library(dplyr)

# Object summary
head(bcr_db,1)

# A tibble: 1 x 70
sequence_id sequence rev_comp productive v_call d_call j_call sequence_alig
  <chr>          <chr>            <lgl>            <lgl>            <chr>  <chr>
<chr>  <chr>
1 CCACTACCAG… ATACTCT… NA         TRUE             IGHV4… IGHD3… IGHJ3…
CAGGTGCAGCTGCAG…
… with 62 more variables

# check out select columns
head(select(bcr_db, cell_id, v_call, j_call, sample, day),1)
# A tibble: 1 x 5
  cell_id                       v_call           j_call   sample        day
  <chr>                         <chr>            <chr>            <chr>
              <dbl>
1 CCACTACCAGTATCTG-1 IGHV4-59*01 IGHJ3*02 P05_FNA_3_12_Y1          12
```



Add BCR data to Seurat object

```
# Make cell IDs in BCR match those in Seurat Object
bcr_db$cell_id_unique = paste0(bcr_db$sample, "_", bcr_db$cell_id)
bcr_db$cell_id_unique = gsub("-1","", bcr_db$cell_id_unique)

bcr_db$cell_id_unique[1]
[1] "P05_FNA_3_12_Y1_CCACTACCAGTATCTG"

Cells(gex_db)[1]
[1] "P05_FNA_12_Y1_TCACAAGTCAAACAAG"

# match index from BCR to GEX information
match.index = match(Cells(gex_db), bcr_db$cell_id_unique)

# What proportion of cells don't have BCRs?
mean(is.na(match.index))
[1] 0.2455369

# Just to double check..
mean(Cells(gex_db) == bcr_db$cell_id_unique[match.index],na.rm=TRUE)
[1] 1
```

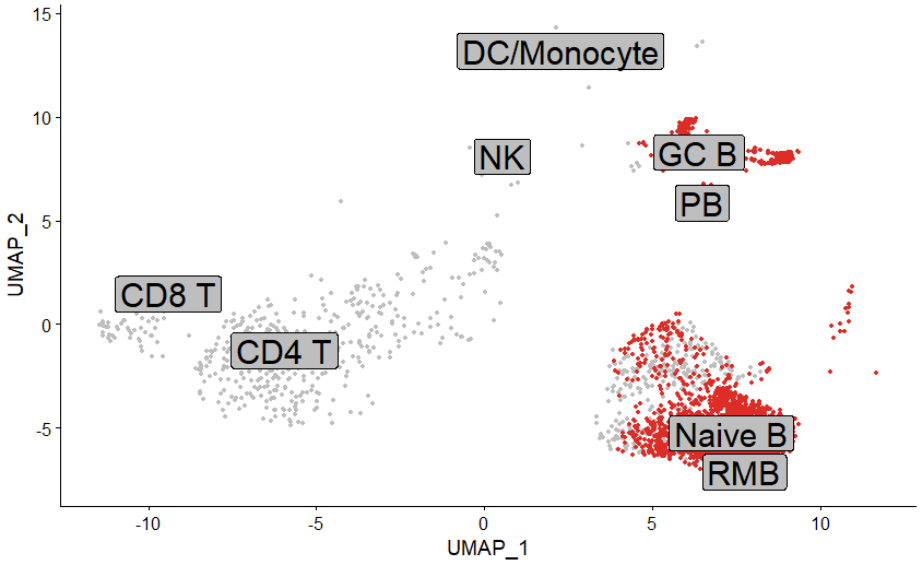
Same form, different order



Add BCR data to Seurat object

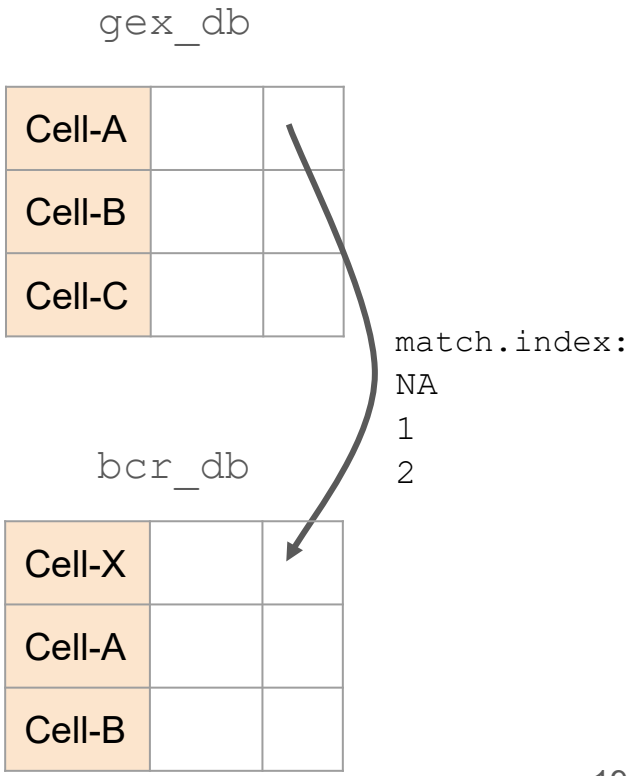
Do cells annotated as B cells actually have BCRs?
Do non-B cells express BCRs?

```
# label whether BCR found in cell
gex_db$contains_bcr = !is.na(match.index)

# List of cells with BCRs
highlighted.cells =
      Cells(gex_db)[which(gex_db$contains_bcr)]

# Plot UMAP with BCR-containing cells
DimPlot(object = gex_db, reduction = "umap",
     cells.highlight = highlighted.cells, label =
     TRUE, cols="gray", pt.size = 1.0,
     label.size=8, label.box=TRUE) + NoLegend()
```



Add GEX data to BCR object

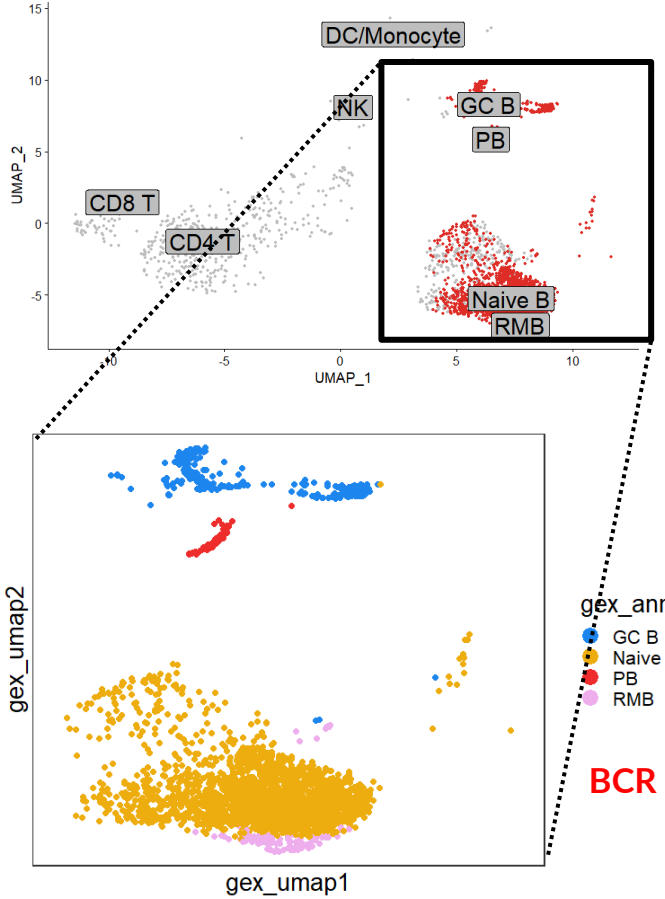
```
# Match indexes from GEX to BCR data
# Different from BCR to GEX!
match.index = match(bcr_db$cell_id_unique, Cells(gex_db))

# What proportion of BCRs don't have GEX information?
mean(is.na(match.index))
[1] 0.09243697

# Add annotations to BCR data
cell.annotation = as.character(Idents(gex_db))
bcr_db$gex_annotation= unlist(lapply(match.index,
    function(x){ifelse(is.na(x),NA, cell.annotation[x])}))

# Add UMAP coordinates to BCR data
umap1 = gex_db@reductions$umap@cell.embeddings[,1]
umap2 = gex_db@reductions$umap@cell.embeddings[,2]

bcr_db$gex_umap1= unlist(lapply(match.index,
    function(x){ifelse(is.na(x),NA, umap1[x])}))
bcr_db$gex_umap2= unlist(lapply(match.index,
    function(x){ifelse(is.na(x),NA, umap2[x])}))

# Remove cells that didn't match
bcr_db = filter(bcr_db, !is.na(gex_annotation))
```



Add GEX data to BCR object

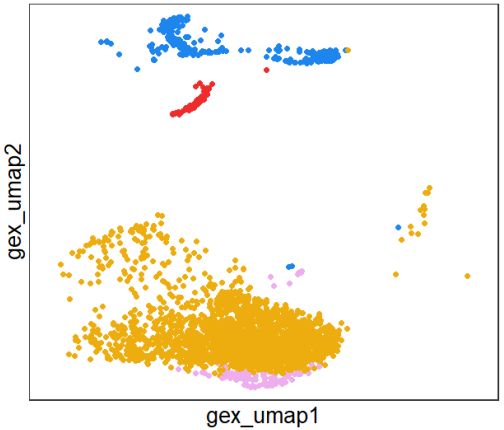
Ensure information transferred from Seurat object

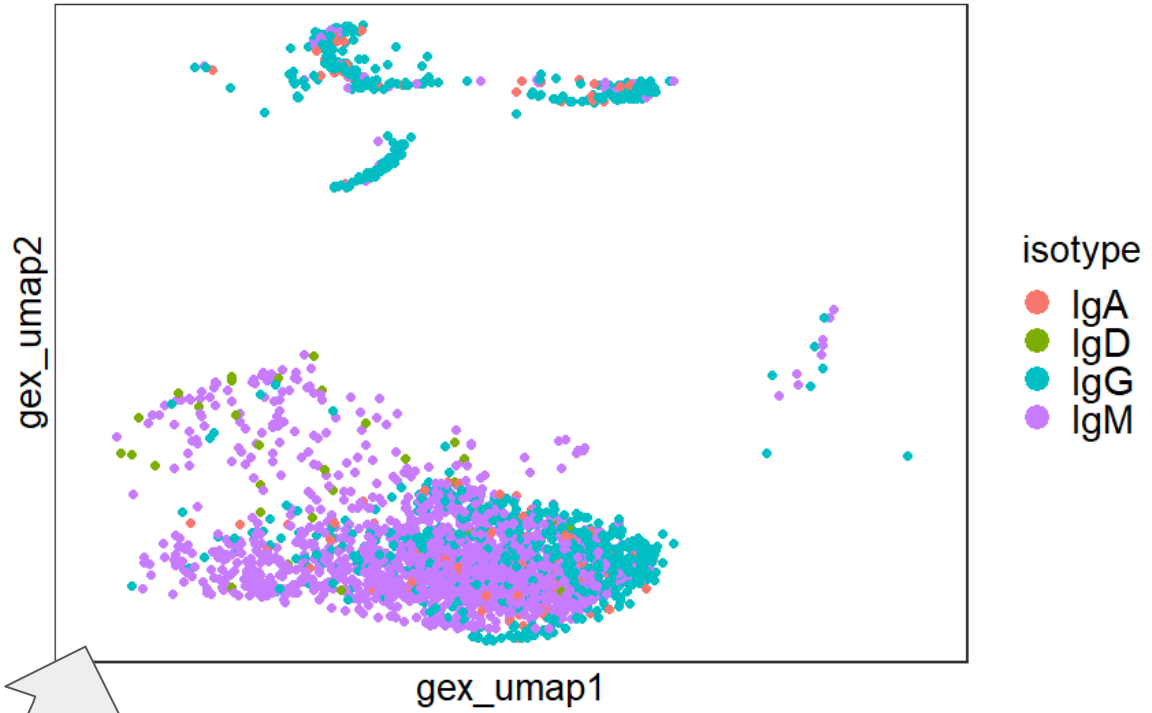
```
library(ggplot2)

# Set up color palette for annotations
col_anno = c(
"GC B"="dodgerblue2",
"PB"="firebrick2",
"ABC"="seagreen",
"Naive B"="darkgoldenrod2",
"RMB"="plum2",
"Germline"="black")

# Plot UMAP from bcr_db
ggplot(bcr_db) +
geom_point(aes(x = gex_umap1, y = gex_umap2,
       color = gex_annotation)) +
       scale_colour_manual(values=col_anno) +
            theme_bw()
```



Add GEX data to BCR object

```
# Plot isotype on UMAP
ggplot(bcr_db) +
      geom_point(aes(x=gex_umap1,
      y = gex_umap2,
      color = isotype)) +
      theme_bw()
```

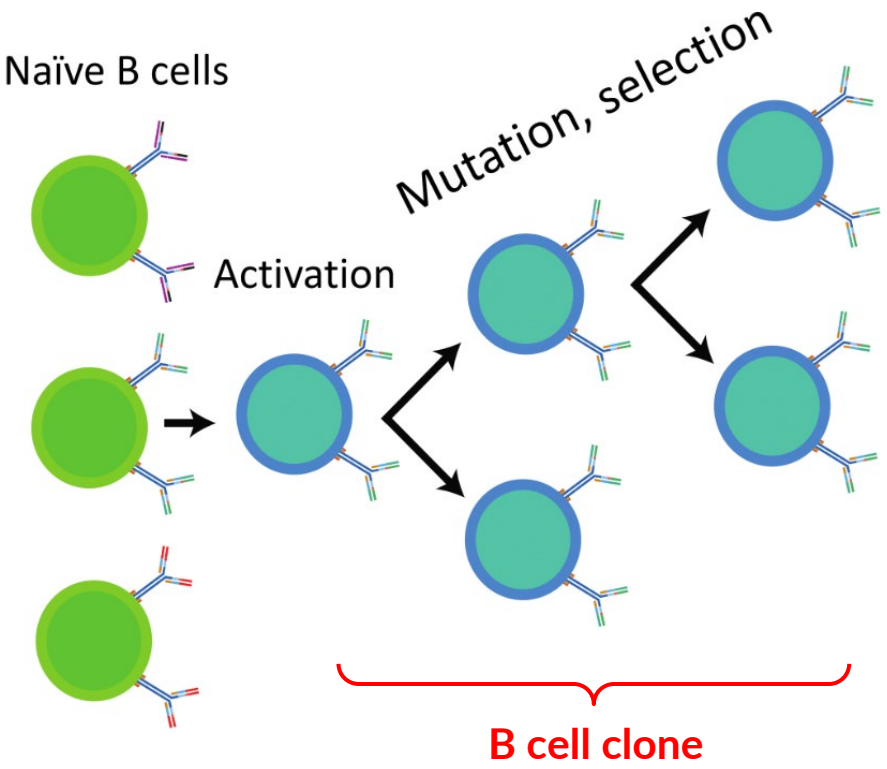




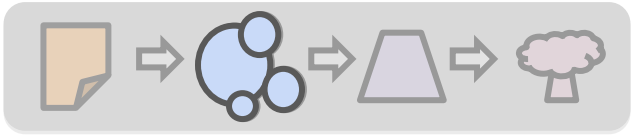
Identifying clonal clusters

Cells in B cell clonal clusters:
- Descend from common V(D)J rearrangement
- Can differ by mutations
- Must be identified before building trees

Our algorithm:
a) Group cells by common heavy chain V-gene, J-gene, and junction length
b) Pick a threshold for CDR3 distance
c) Assign cells within each group into clonal cluster by single-linkage hierarchical clustering

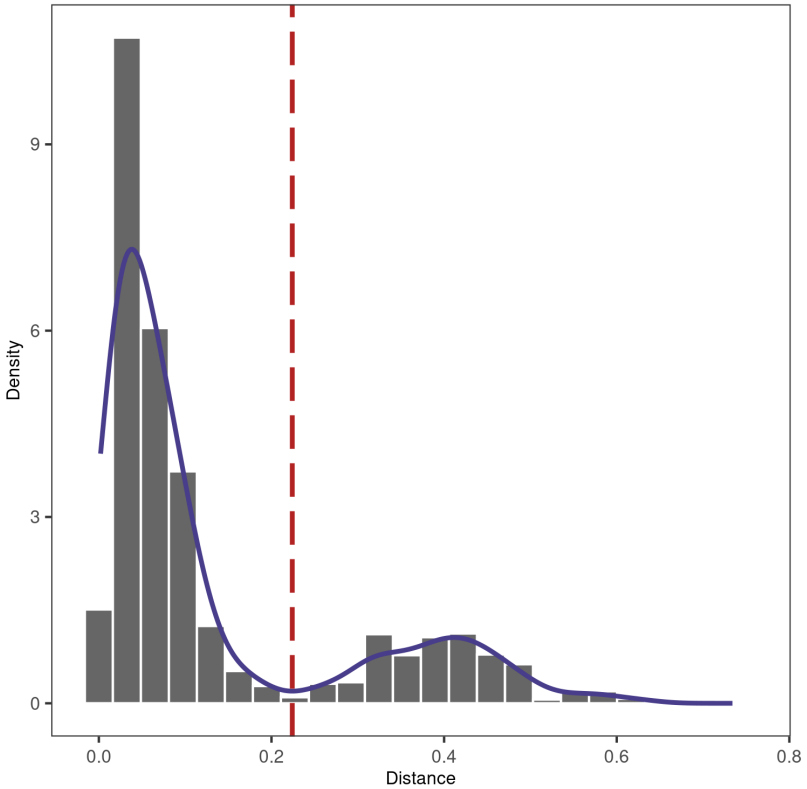


Gupta et al. (2017, JI)
Nouri and Steven H Kleinstein (2018, PLoS CB)



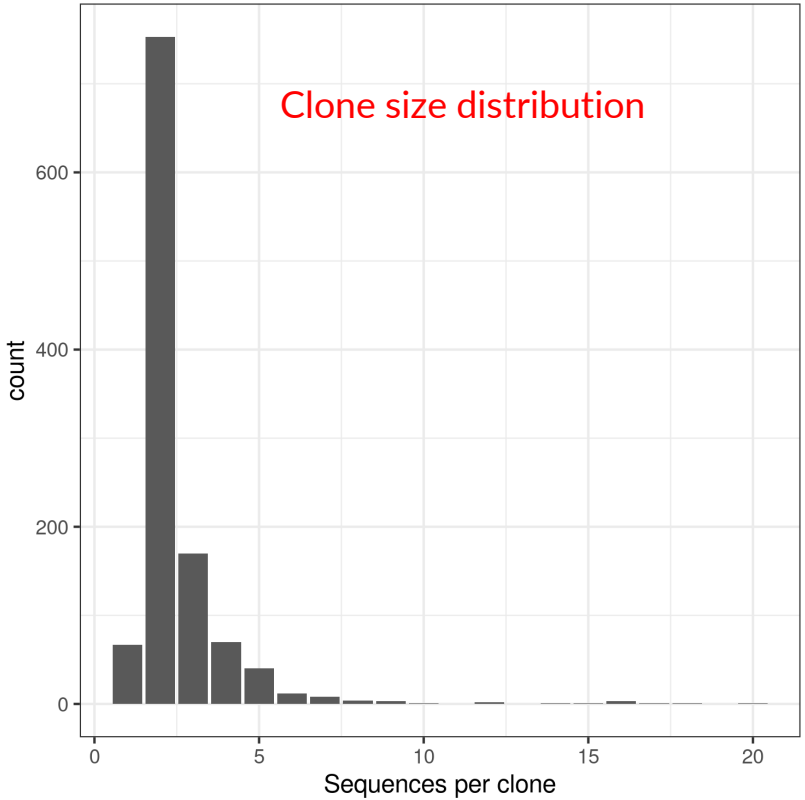
Picking a threshold using *shazam*

```
library(shazam)
library(dplyr)


# Find threshold using heavy chains
dist_ham <- distToNearest(filter(bcr_db,
    locus=="IGH"))

output <- findThreshold(dist_ham$dist_nearest)
threshold <- output@threshold


# Plot distance to each nearest neighbor
plotDensityThreshold(output)
```



Performing clustering using *scoper*

```
library(scoper)
library(dplyr)
library(alakazam)

# Assign clonal clusters
results <- hierarchicalClones(dist_ham,
    threshold=threshold)

results_db <- as.data.frame(results)

# get clone sizes using dplyr functions
clone_sizes <- countClones(results_db)

# Plot cells per clone
ggplot(clone_sizes, aes(x=seq_count))+
    geom_bar() + theme_bw() +
    xlab("Sequences per clone")
```

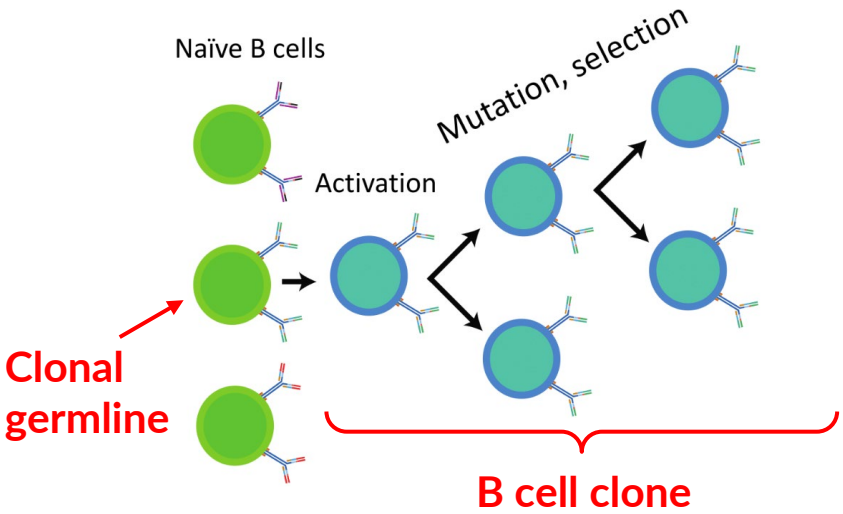


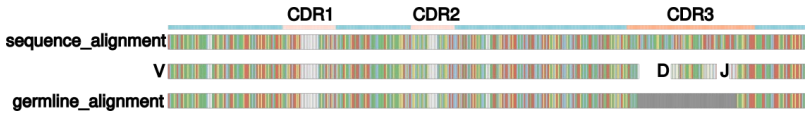
Note: will print out “running in bulk mode” because example data has only heavy chains, other options available if light chains included

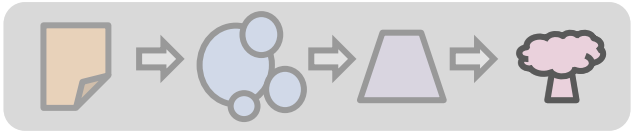
Reconstruct germlines using *dowser*

```
# Get IMGT reference database outside of Docker
# Enter commands in terminal inside tutorial_data/
git clone https://bitbucket.org/kleinstein/immcantation
immcantation/scripts/fetch_imgtdb.sh
```

```
library(dowser)

# read in IMGT data if downloaded on your own (above)
references = readIMGT(dir = "human/vdj/")

# read in IMGT data if using in Docker image
references = readIMGT(dir =
     "/usr/local/share/germlines/imgt/human/vdj")

# Reconstruct germlines
results_db = createGermlines(results_db, references)

# Check output column
results_db$germline_alignment_d_mask[1]
[1] "CAGGTTCAGCTGGTGCAGTCTGGAGCT...GAGGTGAAGAAGCCTGGG"
```



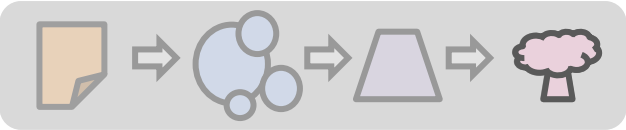




Formatting clones with *dowser*

```
library(dowser)

# Make clone objects with aligned, processed sequences
# collapse identical sequences unless differ by trait
# add up duplicate_count column for collapsed sequences
# store day, isotype, gex_annotation
# discard clones with < 5 distinct sequences
clones = formatClones(results_db,
    traits = c("day", "isotype", "gex_annotation"),
    num_fields=c("duplicate_count"), minseq=5)

clones
# A tibble: 66 x 4
   clone_id data        locus  seqs
   <chr>    <list>      <chr> <int>
 1 344      <airrClon> IGH          16
 2 1018     <airrClon> IGH          15
 3 395      <airrClon> IGH          15
 4 526      <airrClon> IGH          15
 5 1105     <airrClon> IGH          14
 6 477      <airrClon> IGH          13
 7 384      <airrClon> IGH          12
 8 1026     <airrClon> IGH          9
 9 1080     <airrClon> IGH          9
10 380      <airrClon> IGH          9
# … with 56 more rows
```

```
Formal class 'airrClone' [package "dowser"] with 12 slots
  ..@ data    :'data.frame':    16 obs. of  9 variables:
  .. ..$ sequence_id      : chr [1:16] "CGTCACTCACCTGG
  .. ..$ sequence         : chr [1:16] "CAGGTTCACCTGGT
  .. ..$ duplicate_count  : num [1:16] 4 2195 6 61 14 ...
  .. ..$ day              : num [1:16] 5 12 0 0 28 0
  .. ..$ isotype          : chr [1:16] "IgG" "IgG" "IgG"
  .. ..$ gex_annotation   : chr [1:16] "Naive B" "PB"
  .. ..$ lsequence        : chr [1:16] "" "" "" "" ...
  .. ..$ hlsequence       : chr [1:16] "CAGGTTCACCTGGTGC
  .. ..$ collapse_count   : num [1:16] 1 1 1 1 2 1 1 1
  ..@ clone      : chr "344"
  ..@ germline   : chr "CAGGTTCAGCTGGTGCAGTCTGGAGCTGAGGT
  ..@ lgermline  : chr ""
  ..@ hlgermline: chr "CAGGTTCAGCTGGTGCAGTCTGGAGCT
  ..@ v_gene     : chr "IGHV1-18"
  ..@ j_gene     : chr "IGHJ4"
  ..@ junc_len   : num 54
  ..@ locus      : chr [1:372] "IGH" "IGH" "IGH" "IGH" ...
  ..@ region     : chr [1:372] "fwr1" "fwr1" "fwr1" "fwr1"
  ..@ phylo_seq  : chr "sequence"
  ..@ numbers    : int [1:372] 1 2 3 4 5 6 7 8 9 10 ...
```



Constructing trees

Maximum parsimony

Minimize number of mutations.

Pros: Fast, simple, easy to interpret.

Cons: Positively misleading when parallel mutations are likely (long branch lengths)

Example programs: dnapars (PHYLIP), phangorn (R)

Maximum likelihood (standard)

Markov model of mutations
Maximize likelihood of data.

Pros: Less biased than parsimony in some cases.

Cons: Sensitive to poor model choice.

Example programs: dnaml (PHYLIP), RAxML, PhyML, codonPhyML, phangorn (R)

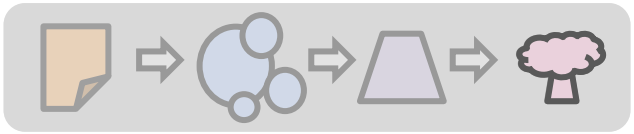
Maximum likelihood (B cell specific)

Incorporate B cell specific mutation biases

Pros: Less bias in parameter estimates from data.

Cons: Slower.

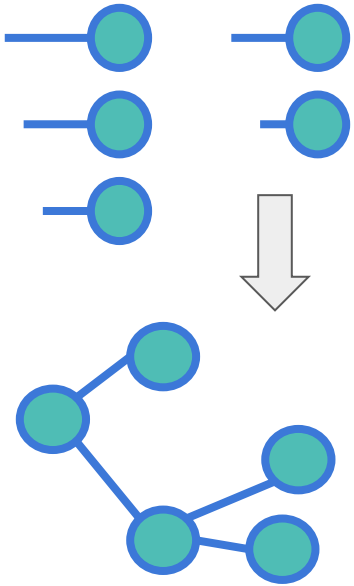
Example programs: IgPhyML

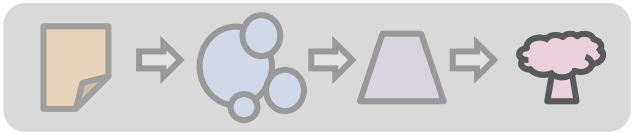


Tree building with *dowser*

```
# Two options for maximum parsimony trees
trees = getTrees(clones)
trees = getTrees(clones, build="dnapars",
      exec="/usr/local/bin/dnapars")

# Two options for standard maximum likelihood trees
trees = getTrees(clones, build="pml", sub_model="GTR")
trees = getTrees(clones, build="dnaml",
       exec="/usr/local/bin/dnaml")

# B cell specific maximum likelihood with IgPhyML
trees = getTrees(clones, build="igphyml",
      exec="/usr/local/share/igphyml/src/igphyml", nproc=2)

trees
# A tibble: 66 x 5
   clone_id data           locus  seqs trees
   <chr>     <list>        <chr> <int> <list>
 1 344       <airrClon> IGH             16 <phylo>
 2 1018      <airrClon> IGH             15 <phylo>
 3 395       <airrClon> IGH             15 <phylo>
 4 526       <airrClon> IGH             15 <phylo>
 5 1105      <airrClon> IGH             13 <phylo>
 6 477       <airrClon> IGH             13 <phylo>
 7 384       <airrClon> IGH             12 <phylo>
# … with 56 more rows
```

Locations here in Docker container
For native installations, go to:
IgPhyML
dnapars/dnaml

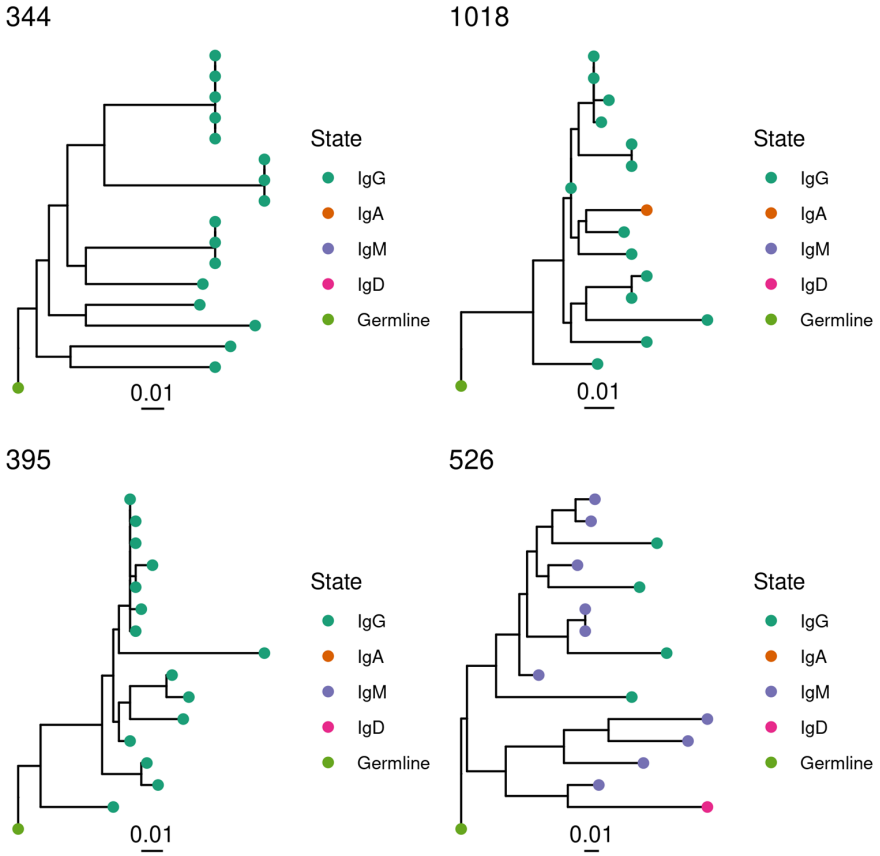




Plotting trees with *dowser* and *ggtree*

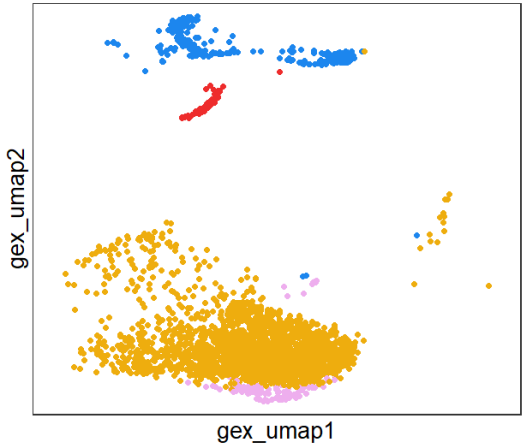
All tree building methods are plotted using the same method in *dowser*

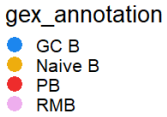
```
# Plot all trees
plots = plotTrees(trees, tips="isotype",
    tipsize=2)


# Plot the largest tree
plots[[1]]


# Save PDF of all trees
treesToPDF(plots, file="final_data_trees.pdf",
    nrow=2, ncol=2)
```

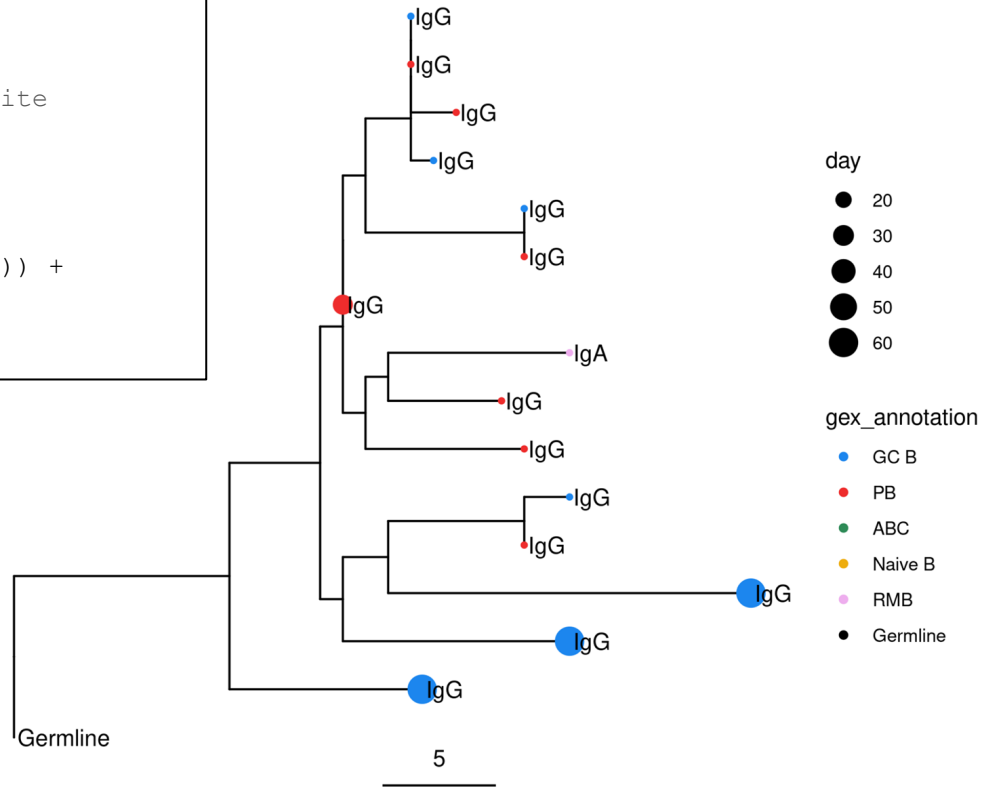


More elaborate tree plots

```
library(ggtree)

# Scale branches to mutations rather than mutations/site
trees = scaleBranches(trees)

# Make fancy tree plot of second largest tree
plotTrees(trees, scale=5)[[2]] +
    geom_tippoint(aes(colour=gex_annotation, size=day)) +
     geom_tiplab(aes(label=isotype), offset=0.002) +
     scale_colour_manual(values = col_anno)
```

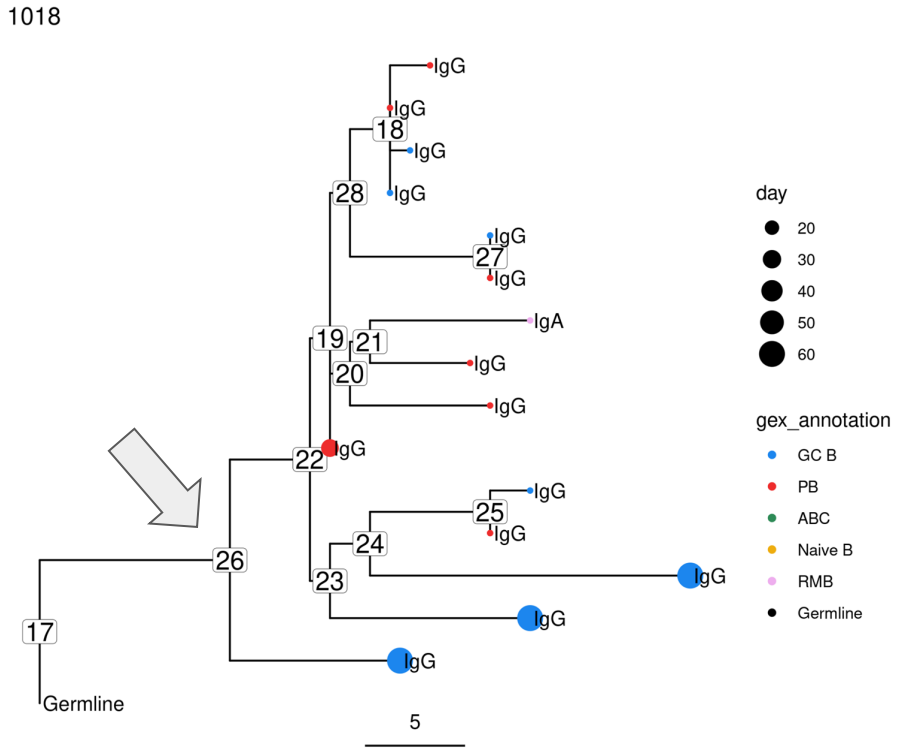






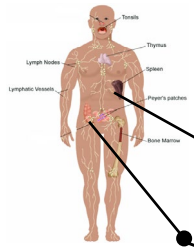
Reconstruct intermediate sequences

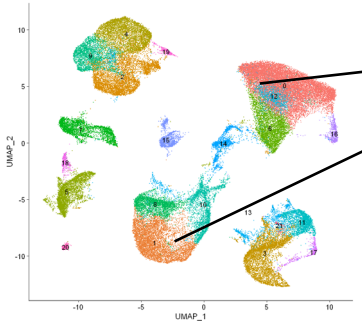
```
library(ggtree)

# Collapse nodes with identical sequences
trees = collapseNodes(trees)

# node_nums=TRUE labels each internal node
p = plotTrees(trees, node_nums=TRUE,
    labelsize=6, scale=5)[[2]] +
    geom_tippoint(aes(colour=gex_annotation,
    size=day)) + geom_tiplab(aes(label=isotype),
    offset=0.002) + scale_colour_manual(values =
    col_anno)

# Get sequence at node 26
getSeq(trees, clone=trees$clone_id[2], node=26)
                          IGH
"GAGGTGCAGCTGGTGGAGTCTGGGGGA...GGCTTGGTCCAGCCTGGGGGGTCCCTGAGACTCTCCTGTGC
      AGCCTCTVCATTCATCCTT............AGTRGCTTTTCGATGAGCTGGGTCCGCCAGGCTC
      CAGGGAAGGGGCTGGAGTGGGTGGCCAACATAAAGCAAGAT......GGAAGTGAGAAATACTAT
      GTGGACTCTGTGAAG...GGCCGATTCACCATCTCCAGAGACAACGCCAAGAACTCACTGTTTCT
      GCAAATGAACAGCCTGAGAGCCGAGGACACGGCTGTGTATTACTGTGCGAGAGATCGTTATGART
      ACGATTGGCGGAGTTTTCGACAAGATGAACCCYCCTACTACTTCCACATGGACGTCTGGGGCAAA
      GGGACCACGGTCACCGTCTCCTCA"
```



Trees link sources of B cell diversity

Immunological connectivity between tissues





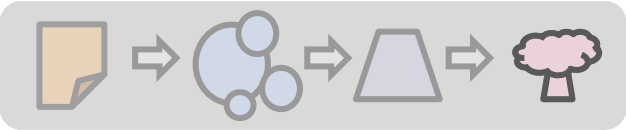
Time

Cell type transitions

Persistence or re-activation

Tissue/cell type connections:
Hoehn et al. (2020) *Phylogenetic analysis of migration, differentiation, and class switching in B cells. bioRxiv* https://doi.org/10.1101/2020.05.30.124446

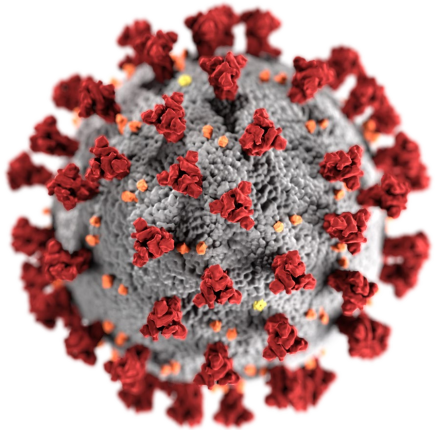
Evolution over time:
Hoehn et al. (2021) *Human B cell lineages engaged by germinal centers following influenza vaccination are measurably evolving. bioRxiv* https://doi.org/10.1101/2021.01.06.425648



Are lineages measurably evolving?

Measurably evolving population are those “from which molecular sequences can be taken at different points in time, among which there are a statistically significant number of genetic differences.” (Drummond et al. 2003)

Measurably evolving over one year



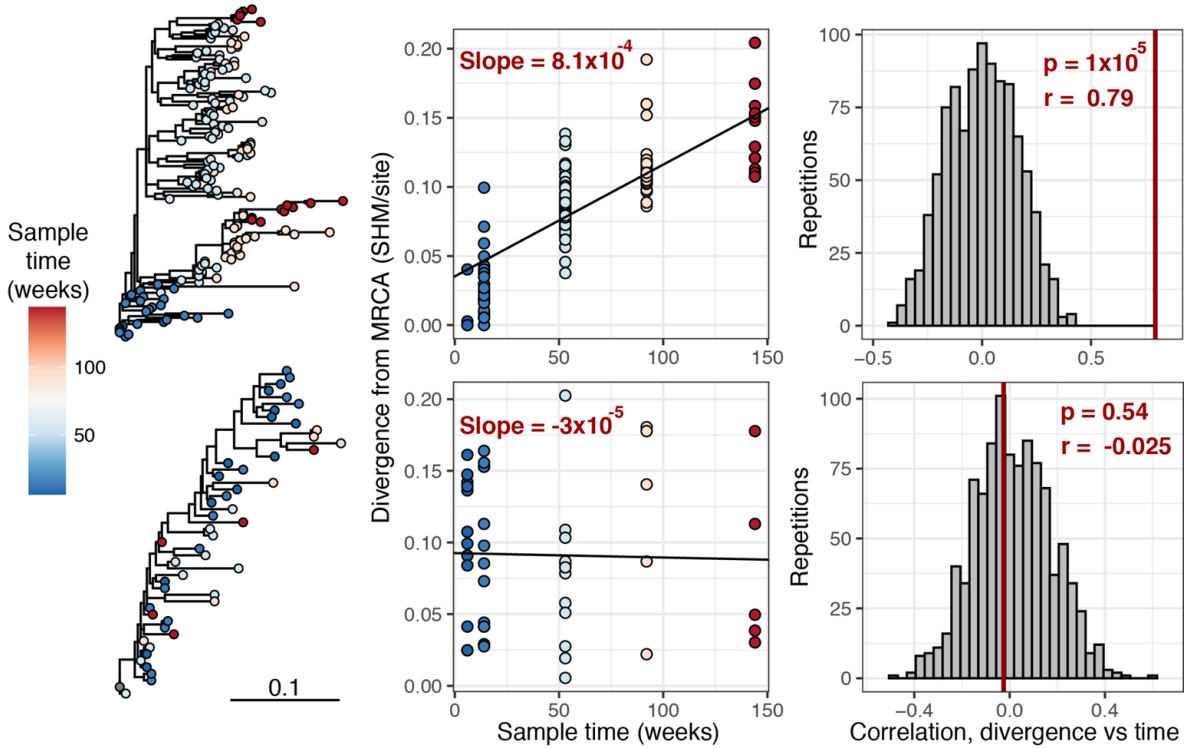
Probably not measurably evolving over one year

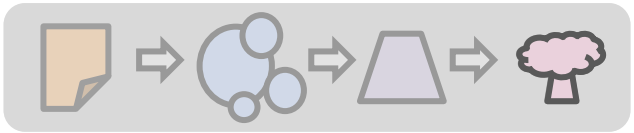
Detecting measurable evolution

Are B cell lineages accumulating additional mutations between sampled timepoints?

Measurably evolving

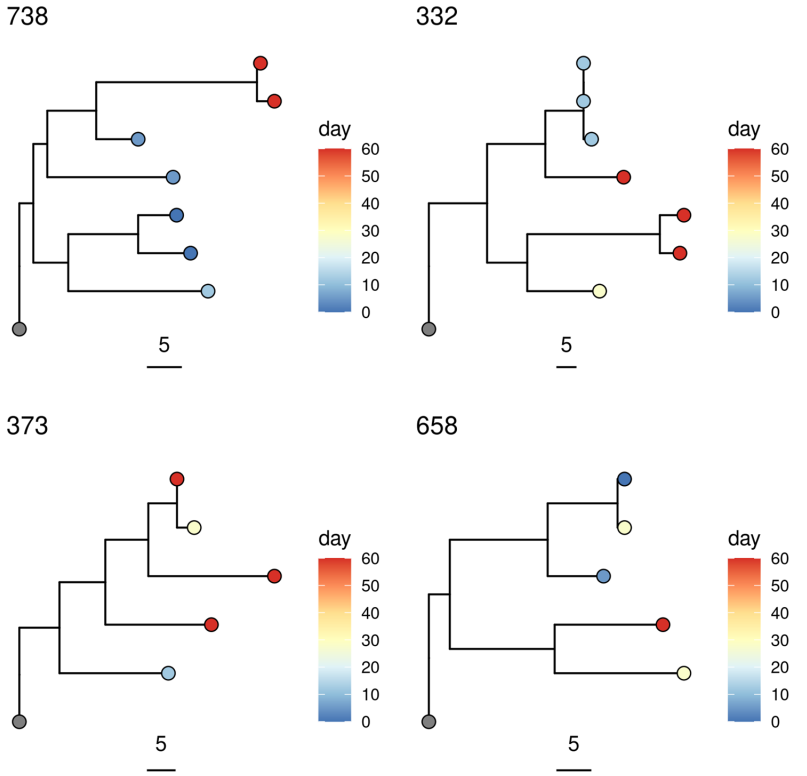
Not measurably evolving





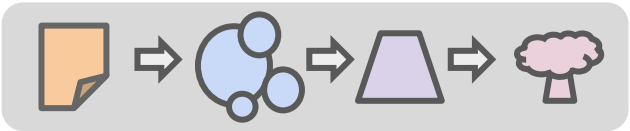
Correlation tests with *dowser*

```
# Correlation test
trees = correlationTest(trees, time="day")

# remove trees with one timepoint, order by p value
trees = filter(trees, !is.na(p))
trees = trees[order(trees$p),]

# Fancy tree plots
p = plotTrees(trees)
p = lapply(p, function(x)
    x + geom_tippoint(aes(fill=day),shape=21, size=3)+
    scale_fill_distiller(palette="RdYlBu"))

treesToPDF(p,file="time_data_trees.pdf")

select(trees, clone_id, slope, correlation, p)
# A tibble: 53 x 4
   clone_id  slope correlation      p
   <chr>     <dbl>       <dbl>  <dbl>
 1 738       0.232       0.908 0.0789
 2 332       0.402       0.889 0.0879
 3 373       0.180       0.553 0.162
 4 658       0.127       0.645 0.174
```



Material covered

1. Immcantation setup.
2. B cell phylogenetics background.
3. Combining gene expression and BCR sequences.
4. Identifying clonal clusters, reconstruct germlines.
5. Building and visualizing trees.
6. Tree analysis, detecting ongoing evolution.



Immcantation

Start to finish suite of programs for BCR and TCR repertoires: immcantation.org
Sequence QC, genotyping, clonal clustering, SHM and phylogenetic analysis.

Vignettes

Detailed usage documentation and tutorials for each individual tool in Immcantation are provided in the main documentation pages for each tool. The following list of shortcuts cover common analyses. Note, each link will leave the Immcantation portal page.

- Assembling raw reads from simple Illumina sequencing protocols with pRESTO
- Assembling raw reads from 5'RACE UMI barcoded Illumina sequencing protocols with pRESTO
- Processing 10x Genomics Cell Ranger data with Change-O
- Processing IgBLAST data with Change-O
- Processing IMGT/HighV-QUEST data with Change-O
- Building lineage trees with IgPhyML
- Assigning clonal groups with SCOPer
- Basic gene usage analysis with Alakazam
- Clonality and diversity analysis with Alakazam
- Mutational load analysis with SHazaM
- Selection pressure analysis with SHazaM
- Building SHM targeting models with SHazaM
- Novel allele detection and genotyping with TIgGER

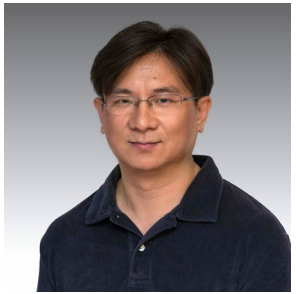
Software documentation

Immcantation: immcantation.org
dowser: dowser.readthedocs.io
IgPhyML: igphyml.readthedocs.io

Ken's email:
kenneth.hoehn@yale.edu

Immcantation software help:
immcantation@googlegroups.com

Noah Yann Lee



Dr. Hailong Meng

Prof. Steven Kleinstein

dowser

Hoehn et al. (2021) *Human B cell lineages engaged by germinal centers following influenza vaccination are measurably evolving.* bioRxiv. https://doi.org/10.1101/2021.01.06.425648

Hoehn et al. (2020) *Phylogenetic analysis of migration, differentiation, and class switching in B cells.* bioRxiv. https://doi.org/10.1101/2020.05.30.124446

Review of B cell phylo

Hoehn et al. (2016) *The diversity and molecular evolution of B-cell receptors during infection.* MBE. https://doi.org/10.1093/molbev/msw015

IgPhyML

Hoehn et al. (2019) *Repertoire-wide phylogenetic models of B cell molecular evolution reveal evolutionary signatures of aging and vaccination.* PNAS. https://doi.org/10.1073/pnas.1906020116

We’re hiring!

Prof. Steven Kleinstein

Looking for postdocs and bioinformaticians interested in computational and systems immunology. Great experimental & clinical collaborators. scRNA-seq + BCR + TCR, multi-omics & more. Develop visualizations & notebooks to support analytical pipelines.

Email: steven.kleinstein@yale.edu