TCREMP supplementary

M.S.

2025-01-09

Properties of TCREMP distances

First, lets show that TCREMP distances for CDR3β region behave as expected. Given sequence similarities s_{ij} one can obtain a metric $d_{ij} = s_{ii} + s_{jj} - 2s_{ij}$ which can be also computed on-the-fly by transforming substitution scoring (e.g. BLOSUM matrix with gaps for linear gaps) appropriately.

We'll first analyze CDR3β distances for $n = 3000$ prototypes mapped to themselves and answer two questions:

- **Q1**: What is the distribution of alignment scores d_{ij} and pairwise Euclidean distances D_{ij} in embedding space
- **Q2**: The properties of Euclidean distances are well-known, but are the alignment scores additive (so PCA can be applied)?
- **Q3**: How does pairwise distances in embedding space agree with actual alignment scores

Load data and compute alignment metric

```
data.1 <- read_tsv("p1000_p1000.txt.gz") |>
  rename(from = id) |>
  mutate(from = as.character(from)) |>
  melt() |>
  filter(grepl("cdr3", variable)) |>
  mutate(to = str_split_fixed(variable, "_", 2)[,1]) |>
  select(-variable)
```

```
## Rows: 994 Columns: 3001
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## dbl (3001): id, 0_v_score, 0_j_score, 0_cdr3_score, 1_v_score, 1_j_score, 1_...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
## Using from as id variables
```

```
ids <- intersect(data.1$from, data.1$to)

data.1 <- data.1 |>
  filter(from %in% ids, to %in% ids)

data.1 <- data.1 |>
  rename(Sij = value) |>
  group_by(from) |>
  mutate(Sii = Sij[from == to]) |>
  group_by(to) |>
  mutate(Sjj = Sij[from == to]) |>
  ungroup() |>
  mutate(Dij = Sii + Sjj - 2 * Sij)

glimpse(data.1)
```

```
## Rows: 988,036
## Columns: 6
## $ from <chr> "0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12~
## $ Sij  <dbl> 730, -250, 150, -80, 10, 0, -250, -140, -230, -140, 10, 10, -240,~
## $ to   <chr> "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", ~
## $ Sii  <dbl> 730, 400, 550, 410, 610, 500, 270, 250, 350, 280, 670, 580, 340, ~
## $ Sjj  <dbl> 730, 730, 730, 730, 730, 730, 730, 730, 730, 730, 730, 730, 730, ~
## $ Dij  <dbl> 0, 1630, 980, 1300, 1320, 1230, 1500, 1260, 1540, 1290, 1380, 129~
```

Compute embedding metric, append values

```
data.1m <- data.1 |>
  dcast(from ~ to, value.var = "Dij")

rownames(data.1m) <- data.1m$from
data.1m$from <- NULL
data.1m <- as.matrix(data.1m)
data.1d <- dist(data.1m) |>
  as.matrix() |>
  melt()
colnames(data.1d) <- c("from", "to", "DDij")
data.1d$from <- as.character(data.1d$from)
data.1d$to <- as.character(data.1d$to)
data.1 <- left_join(data.1, data.1d)
```

```
## Joining with `by = join_by(from, to)`
```

```
glimpse(data.1)
```

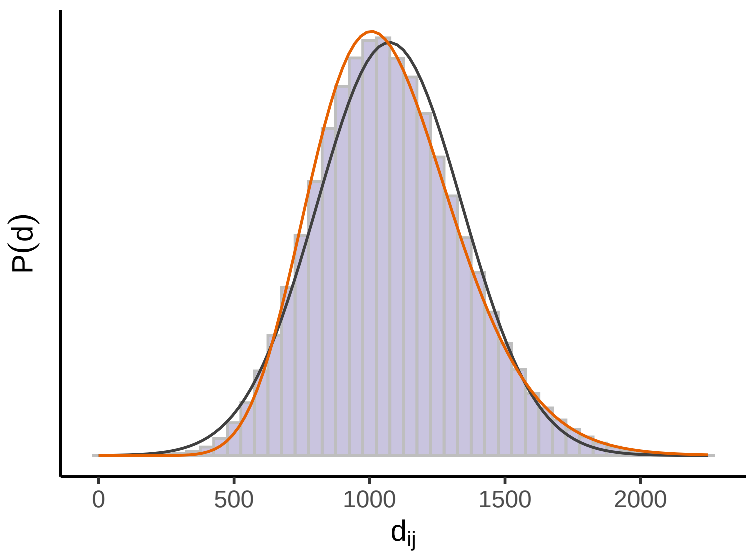
```
## Rows: 988,036
## Columns: 7
## $ from <chr> "0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12~
## $ Sij  <dbl> 730, -250, 150, -80, 10, 0, -250, -140, -230, -140, 10, 10, -240,~
## $ to   <chr> "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", ~
## $ Sii  <dbl> 730, 400, 550, 410, 610, 500, 270, 250, 350, 280, 670, 580, 340, ~
## $ Sjj  <dbl> 730, 730, 730, 730, 730, 730, 730, 730, 730, 730, 730, 730, 730, ~
## $ Dij  <dbl> 0, 1630, 980, 1300, 1320, 1230, 1500, 1260, 1540, 1290, 1380, 129~
## $ DDij <dbl> 0.000, 9756.895, 8095.134, 12501.840, 6352.008, 9569.922, 11944.4~
```

We provide two fits for d_{ij}, first is $\mathcal{N}(\mu, \sigma)$ and the second one that recaptures the right-skewness is $\Gamma(\alpha = \mu^2/\sigma^2, \lambda = \mu/\sigma^2)$, the Gamma distribution. This is in line with Pang, H., Tang, J., Chen, SS. et al. Statistical distributions of optimal global alignment scores of random protein sequences. BMC Bioinformatics 6, 257 (2005). A thorough theoretical proof for this is left for the reader. We will just note that if, for first half of

CDR3β sequence $d_{ij}^l \sim \Gamma(\cdot)$ then $d_{ij} = d_{ij}^l + d_{ij}^r \sim \Gamma(\cdot)$ due to the nature of Gamma distribution. Also note that number of matches between two random strings of amino acids can be modeled as a Poisson process which leads to Gamma distribution.

> N.B. We use Gamma distribution instead of Erlang distribution as sequence alignment scores may be non-integer

```r
data.1diag <- data.1 |>
  filter(as.integer(from) < as.integer(to))

mu <- mean(data.1diag$Dij)
sigma <- sd(data.1diag$Dij)
alpha <- mu * mu / sigma / sigma
lambda <- alpha / mu

fig1 <- data.1diag |>
  ggplot(aes(x = Dij)) +
  geom_histogram(aes(y = after_stat(density)),
                 binwidth = 50,
                 color = "grey",
                 fill = "#b2abd2",
                 alpha = 0.7) +
  stat_function(fun = dnorm,
                args = list(mean = mu, sd = sigma),
                color = "grey25") +
  stat_function(fun = dgamma,
                args = list(shape = alpha, rate = lambda),
                color = "#e66101") +
  xlab(TeX("$d_{ij}$")) + ylab(TeX("$P(d)$")) +
  theme_classic() +
  theme(axis.text.y = element_blank(),
        axis.ticks.y = element_blank())
fig1
```



```r
pdf("figs_aux_1A.pdf")
fig1
dev.off()
```

```
## pdf
##   2
```

We can fit distribution of Euclidean distances D_{ij} using a generalized extreme value (GEV) distribution $P_{GEV}(\mu, \sigma, \xi)$. It is said, that GEV distribution is often used as an approximation to model the maxima of long (finite) sequences of random variables. Namely, we aim to fit the Fréchet distribution aka type II GEV. We will first scale $D_{ij} \to \frac{D_{ij} - \mathrm{E}[D_{ij}]}{\mathrm{SD}[D_{ij}]}$ to simplify computations.

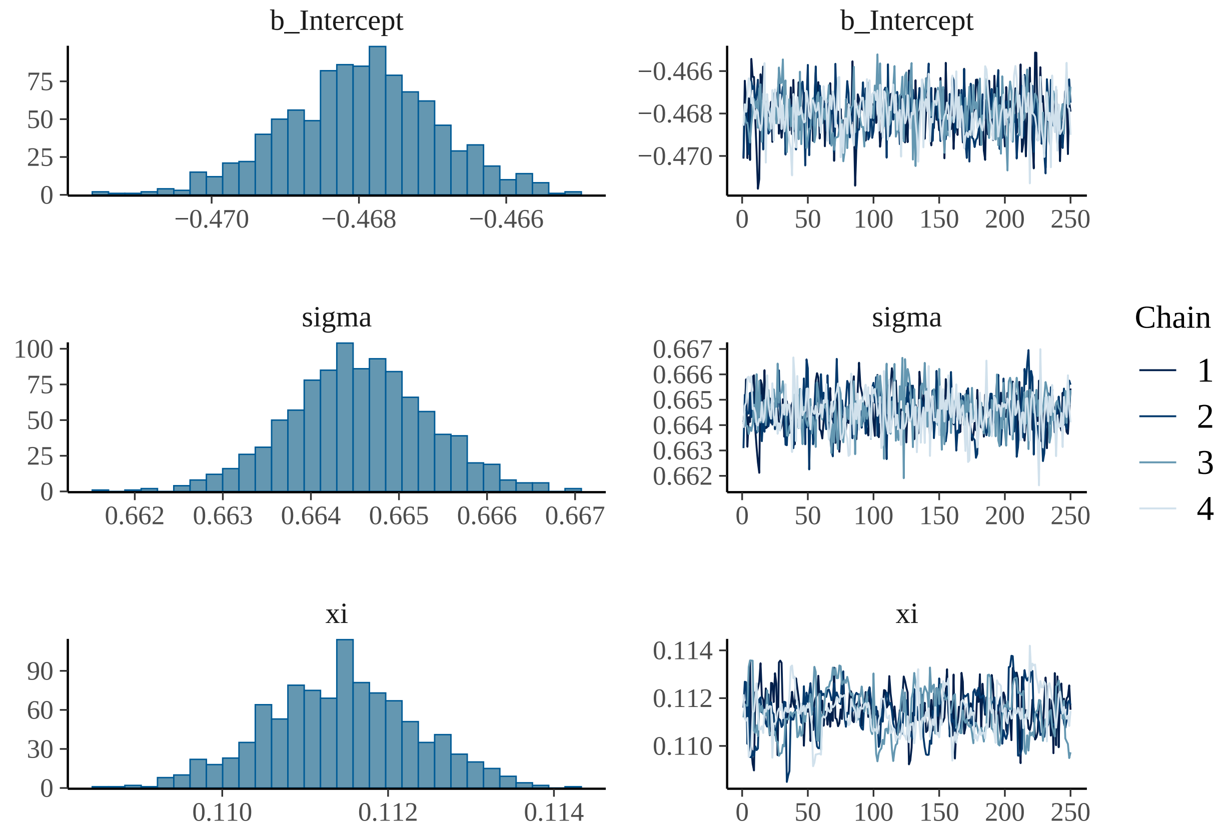
```r
DDij.mean <- mean(data.1diag$DDij)
DDij.sd <- sd(data.1diag$DDij)
ffit <- brm(formula = y ~ 1,
            data = data.1diag |>
              mutate(y = (DDij - DDij.mean) / DDij.sd),
            family = "gen_extreme_value",
            cores = 10,
            seed = 42,
            iter = 500)
```

```
## Warning: The 'gen_extreme_value' family is deprecated and will be removed in
## the future.

## Compiling Stan program...

## Trying to compile a simple C file

## Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
## using C compiler: ‘Apple clang version 16.0.0 (clang-1600.0.26.6)’
## using SDK: ‘MacOSX15.2.sdk’
## clang -arch arm64 -I"/Library/Frameworks/R.framework/Resources/include" -DNDEBUG   -I"/Library/Frame
## In file included from <built-in>:1:
## In file included from /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/library/StanHeade
## In file included from /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/library/RcppEigen
## In file included from /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/library/RcppEigen
## /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/library/RcppEigen/include/Eigen/src/Cor
##   679 | #include <cmath>
##       |          ^~~~~~~
## 1 error generated.
## make: *** [foo.o] Error 1

## Start sampling

## Warning: Bulk Effective Samples Size (ESS) is too low, indicating posterior means and medians may be
## Running the chains for more iterations may help. See
## https://mc-stan.org/misc/warnings.html#bulk-ess

## Warning: Tail Effective Samples Size (ESS) is too low, indicating posterior variances and tail quant
## Running the chains for more iterations may help. See
## https://mc-stan.org/misc/warnings.html#tail-ess
```

```r
fvars <- summary(ffit)
print(fvars)
```

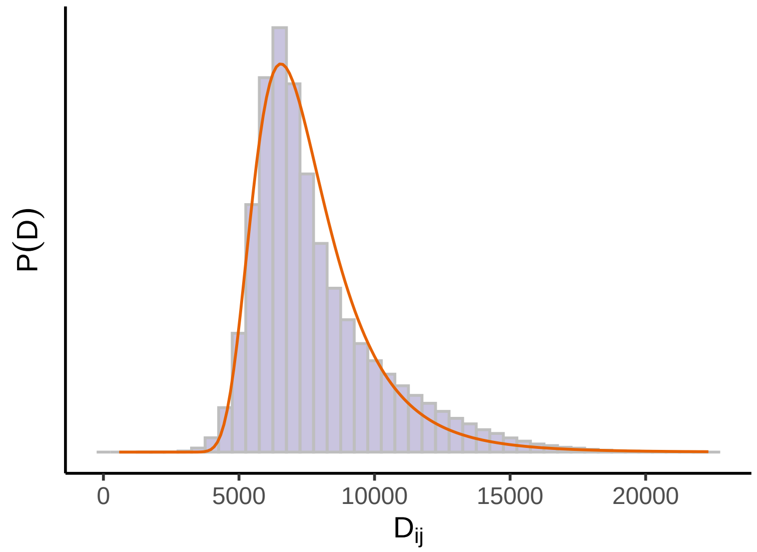
```
##  Family: gen_extreme_value
##   Links: mu = identity; sigma = identity; xi = identity
## Formula: y ~ 1
##    Data: mutate(data.1diag, y = (DDij - DDij.mean)/DDij.sd) (Number of observations: 493521)
##   Draws: 4 chains, each with iter = 500; warmup = 250; thin = 1;
##          total post-warmup draws = 1000
##
## Regression Coefficients:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept    -0.47      0.00    -0.47    -0.47 1.00      918      710
##
## Further Distributional Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     0.66      0.00     0.66     0.67 1.01      794      769
## xi        0.11      0.00     0.11     0.11 1.01      221      339
##
## Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
## in 2:
## Chain 2:
## Chain 3: Iteration:   1 / 500 [  0%]  (Warmup)
## Chain 2: Iteration:   1 / 500 [  0%]  (Warmup)
## Chain 4: Iteration:   1 / 500 [  0%]  (Warmup)
## Chain 1: Iteration:   1 / 500 [  0%]  (Warmup)
## Chain 2: Iteration:  50 / 500 [ 10%]  (Warmup)
## Chain 1: Iteration:  50 / 500 [ 10%]  (Warmup)
## Chain 3: Iteration:  50 / 500 [ 10%]  (Warmup)
## Chain 4: Iteration:  50 / 500 [ 10%]  (Warmup)
## Chain 1: Iteration: 100 / 500 [ 20%]  (Warmup)
## Chain 2: Iteration: 100 / 500 [ 20%]  (Warmup)
## Chain 3: Iteration: 100 / 500 [ 20%]  (Warmup)
## Chain 1: Iteration: 150 / 500 [ 30%]  (Warmup)
## Chain 2: Iteration: 150 / 500 [ 30%]  (Warmup)
## Chain 4: Iteration: 100 / 500 [ 20%]  (Warmup)
## Chain 2: Iteration: 200 / 500 [ 40%]  (Warmup)
## Chain 1: Iteration: 200 / 500 [ 40%]  (Warmup)
## Chain 1: Iteration: 250 / 500 [ 50%]  (Warmup)
## Chain 1: Iteration: 251 / 500 [ 50%]  (Sampling)
## Chain 3: Iteration: 150 / 500 [ 30%]  (Warmup)
## Chain 4: Iteration: 150 / 500 [ 30%]  (Warmup)
## Chain 2: Iteration: 250 / 500 [ 50%]  (Warmup)
## Chain 2: Iteration: 251 / 500 [ 50%]  (Sampling)
## Chain 1: Iteration: 300 / 500 [ 60%]  (Sampling)
## Chain 2: Iteration: 300 / 500 [ 60%]  (Sampling)
## Chain 1: Iteration: 350 / 500 [ 70%]  (Sampling)
## Chain 3: Iteration: 200 / 500 [ 40%]  (Warmup)
## Chain 4: Iteration: 200 / 500 [ 40%]  (Warmup)
## Chain 1: Iteration: 400 / 500 [ 80%]  (Sampling)
## Chain 2: Iteration: 350 / 500 [ 70%]  (Sampling)
## Chain 1: Iteration: 450 / 500 [ 90%]  (Sampling)
## Chain 3: Iteration: 250 / 500 [ 50%]  (Warmup)
## Chain 3: Iteration: 251 / 500 [ 50%]  (Sampling)
## Chain 4: Iteration: 250 / 500 [ 50%]  (Warmup)
## Chain 4: Iteration: 251 / 500 [ 50%]  (Sampling)
## Chain 1: Iteration: 500 / 500 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 120.096 seconds (Warm-up)
## Chain 1:                99.759 seconds (Sampling)
## Chain 1:                219.855 seconds (Total)
## Chain 1:
## Chain 2: Iteration: 400 / 500 [ 80%]  (Sampling)
## Chain 3: Iteration: 300 / 500 [ 60%]  (Sampling)
## Chain 2: Iteration: 450 / 500 [ 90%]  (Sampling)
## Chain 4: Iteration: 300 / 500 [ 60%]  (Sampling)
## Chain 2: Iteration: 500 / 500 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 128.297 seconds (Warm-up)
## Chain 2:                156.051 seconds (Sampling)
## Chain 2:                284.348 seconds (Total)
## Chain 2:
## Chain 3: Iteration: 350 / 500 [ 70%]  (Sampling)
## Chain 4: Iteration: 350 / 500 [ 70%]  (Sampling)
## Chain 4: Iteration: 400 / 500 [ 80%]  (Sampling)
## Chain 3: Iteration: 400 / 500 [ 80%]  (Sampling)
## Chain 4: Iteration: 450 / 500 [ 90%]  (Sampling)
## Chain 3: Iteration: 450 / 500 [ 90%]  (Sampling)
## Chain 4: Iteration: 500 / 500 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 221.874 seconds (Warm-up)
## Chain 4:                174.613 seconds (Sampling)
## Chain 4:                396.487 seconds (Total)
## Chain 4:
## Chain 3: Iteration: 500 / 500 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 203.095 seconds (Warm-up)
## Chain 3:                207.413 seconds (Sampling)
## Chain 3:                410.508 seconds (Total)
## Chain 3:
```

```r
plot(ffit)
```



Plot results

```
fmu <- fvars$fixed$Estimate
fsigma <- fvars$spec_pars$Estimate[1]
fxi <- fvars$spec_pars$Estimate[2]

fig2 <- data.1diag |>
  ggplot(aes(x = DDij)) +
  geom_histogram(aes(y = after_stat(density)),
                 binwidth = 500,
                 color = "grey",
                 fill = "#b2abd2",
                 alpha = 0.7) +
  geom_line(data = tibble(x = (-60:120 / 20) * DDij.sd + DDij.mean,
                          y = dgen_extreme_value(-60:120 / 20,
                                                 mu = fmu,
                                                 sigma = fsigma * 0.9,
                                                 xi = fxi
                                               ) / DDij.sd
                          ),
                          aes(x, y),
                color = "#e66101") +
  scale_x_continuous(TeX("$D_{ij}$")) +
  ylab(TeX("$P(D)$")) +
  theme_classic() +
  theme(axis.text.y = element_blank(),
        axis.ticks.y = element_blank())
fig2
```



```
pdf("figs_aux_1B.pdf")
fig2
dev.off()
```

```
## pdf
##   2
```

Q1 : $d_{ij} \sim \Gamma, D_{ij} \sim \text{GEV} \square$

Check approx. additivity and triangle rule, circle through triplicates

```
data.1tri <- expand_grid(a = 0:999, b = 0:999, c = 0:999) |>
  filter(a < b, b < c) |>
  mutate(a = as.character(a),
         b = as.character(b),
         c = as.character(c)) |>
  left_join(data.1 |>
              select(a = from, b = to, Dab = Dij, DDab = DDij) |>
              filter(a < b)) |>
  left_join(data.1 |>
              select(a = from, c = to, Dac = Dij, DDac = DDij) |>
              filter(a < c)) |>
  left_join(data.1 |>
              select(b = from, c = to, Dbc = Dij, DDbc = DDij) |>
              filter(b < c))
```

```
## Joining with `by = join_by(a, b)`
## Joining with `by = join_by(a, c)`
## Joining with `by = join_by(b, c)`
```

Plot them

```
fig3 <- data.1tri |>
  sample_n(100000) |>
  ggplot(aes(x = (Dab + Dbc) / 2, y = Dac)) +
  geom_hex(bins = 30) +
  scale_fill_distiller(palette = "Purples", direction = 1, guide = F) +
  geom_abline(slope = 1, intercept = 0,
              linetype = "dashed", color = "grey25") +
  scale_x_continuous(TeX("$(d_{ij} + d_{jk})/2$"),
                     limits = c(100, 2000)) +
  scale_y_continuous(TeX("$d_{ik}$"),
                     limits = c(100, 2000)) +
  theme_classic() +
  theme(aspect.ratio = 1)

with(data.1tri,
     cor.test(Dab + Dbc, Dac))
```

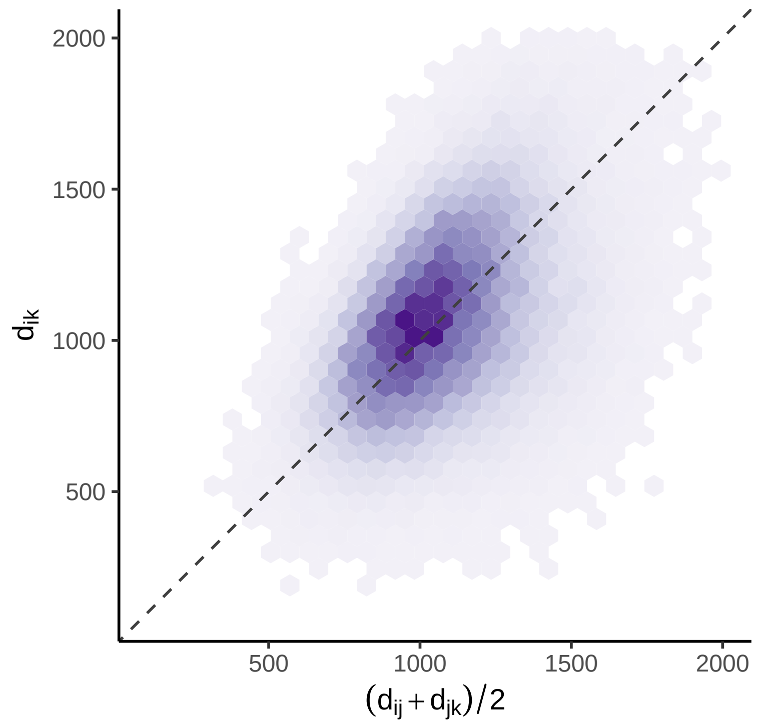
```
##
##  Pearson's product-moment correlation
##
## data:  Dab + Dbc and Dac
## t = 5873.1, df = 134308859, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.4519066 0.4521757
## sample estimates:
##       cor
## 0.4520411
```

```
fig3
```

```
## Warning: Removed 19262 rows containing non-finite outside the scale range
## (`stat_binhex()`).
```

```
## Warning: The `guide` argument in `scale_*()` cannot be `FALSE`. This was deprecated in
## ggplot2 3.3.4.
## i Please use "none" instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## Warning: Removed 1 row containing missing values or values outside the scale range
## (`geom_hex()`).
```



```
pdf("figs_aux_2A.pdf")
fig3
```

```
## Warning: Removed 19262 rows containing non-finite outside the scale range
## (`stat_binhex()`).
## Removed 1 row containing missing values or values outside the scale range
## (`geom_hex()`).
```

```
dev.off()
```

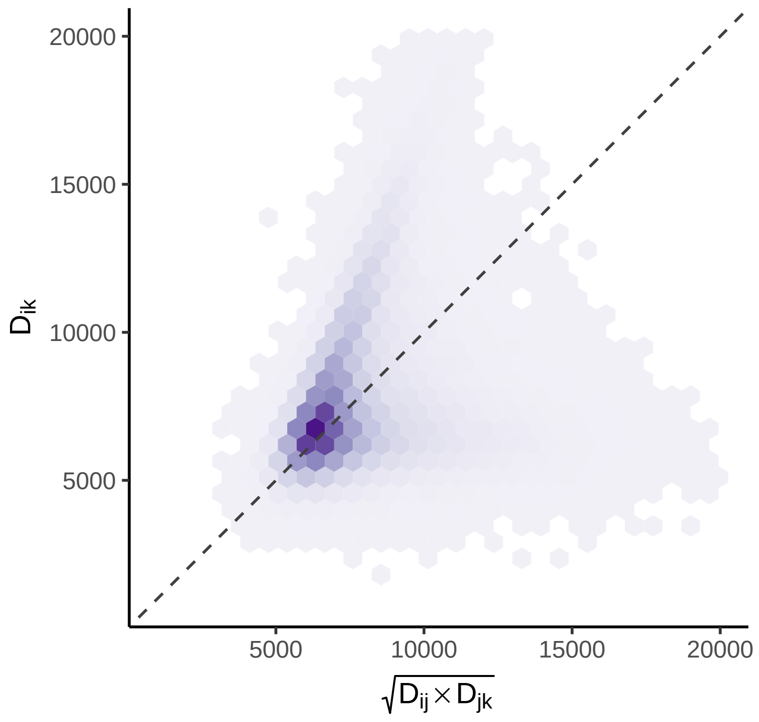
```
## pdf
##   2
```

```
fig4 <- data.1tri |>
  sample_n(100000) |>
  ggplot(aes(x = sqrt(DDab * DDbc), y = DDac)) +
  geom_hex(bins = 30) +
  scale_fill_distiller(palette = "Purples", direction = 1, guide = F) +
  geom_abline(slope = 1, intercept = 0,
              linetype = "dashed", color = "grey25") +
  scale_x_continuous(TeX("$\\sqrt{D_{ij} \\times D_{jk}}$"),
                     limits = c(1000, 20000)) +
  scale_y_continuous(TeX("$D_{ik}$"), limits = c(1000, 20000)) +
  theme_classic() +
  theme(aspect.ratio = 1)

with(data.1tri,
     cor.test(sqrt(DDab * DDbc), DDac))
```

```
##
##  Pearson's product-moment correlation
##
## data:  sqrt(DDab * DDbc) and DDac
## t = 2798.4, df = 134308859, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.2345626 0.2348822
## sample estimates:
##       cor
## 0.2347224
```

```
fig4
```

```
## Warning: Removed 19068 rows containing non-finite outside the scale range
## (`stat_binhex()`).
```



```
pdf("figs_aux_2B.pdf")
fig4
```

```
## Warning: Removed 19068 rows containing non-finite outside the scale range
## (`stat_binhex()`).
```

```
dev.off()
```

```
## pdf
##   2
```

Q2 : $d_{i,j} \sim d_{i,\cdot} + d_{\cdot,j}, \log D_{i,j} \sim \log D_{i,\cdot} + \log D_{\cdot,j}$ □

Compare alignment and embedding distances; also check that dissimilarity scores have good negative correlation with similarity scores

```
fig5 <- data.1diag |>
  ggplot(aes(x = Dij, y = Sij)) +
  geom_hex(bins = 30) +
  scale_fill_distiller(palette = "Purples", direction = 1, guide = F) +
  geom_smooth(fill = NA, color = "#e66101") +
  scale_x_continuous(TeX("$d_{ij}$"), limits = c(0, 2000)) +
  scale_y_continuous(TeX("$s_{ij}$"), limits = c(0, 400)) +
  theme_classic() +
  theme(aspect.ratio = 1)
with(data.1diag,
     cor.test(Dij, Sij))
```

```
##
##  Pearson's product-moment correlation
##
## data:  Dij and Sij
## t = -698.48, df = 493519, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.7064695 -0.7036635
## sample estimates:
##        cor
## -0.7050693
```

```
with(data.1diag,
     cor.test(Sij, DDij))
```

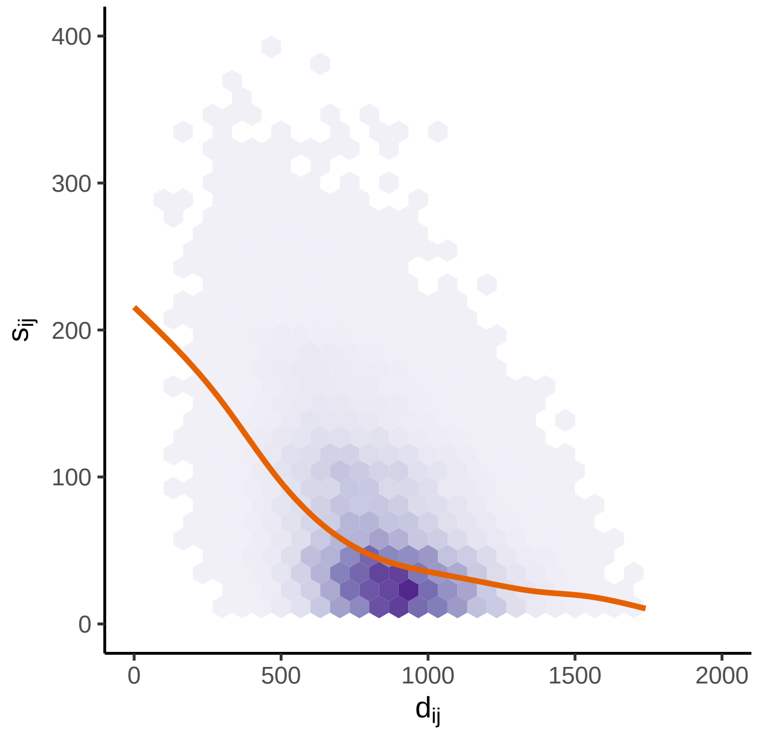
```
##
##  Pearson's product-moment correlation
##
## data:  Sij and DDij
## t = -301.88, df = 493519, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.3971552 -0.3924450
## sample estimates:
##        cor
## -0.3948027
```

```
fig5
```

```
## Warning: Removed 358400 rows containing non-finite outside the scale range
## (`stat_binhex()`).
```

```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```

```
## Warning: Removed 358400 rows containing non-finite outside the scale range
## (`stat_smooth()`).
## Warning: Removed 24 rows containing missing values or values outside the scale range
## (`geom_hex()`).
```



```
pdf("figs_aux_2C.pdf")
fig5
```

```
## Warning: Removed 358400 rows containing non-finite outside the scale range
## (`stat_binhex()`).

## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'

## Warning: Removed 358400 rows containing non-finite outside the scale range
## (`stat_smooth()`).
## Removed 24 rows containing missing values or values outside the scale range
## (`geom_hex()`).
```

```
dev.off()
```

```
## pdf
##   2
```

```
fig6 <- data.1diag |>
  ggplot(aes(x = Dij, y = DDij)) +
  geom_hex(bins = 30) +
  scale_fill_distiller(palette = "Purples", direction = 1, guide = F) +
  geom_smooth(fill = NA, color = "#e66101") +
  scale_x_continuous(TeX("$d_{ij}$"), limits = c(100, 2000)) +
  scale_y_continuous(TeX("$D_{ij}$"), limits = c(1000, 20000)) +
  theme_classic() +
  theme(aspect.ratio = 1)
with(data.1diag,
     cor.test(Dij, DDij))
```

```
##
##  Pearson's product-moment correlation
##
## data:  Dij and DDij
## t = 469.74, df = 493519, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.5539145 0.5577704
## sample estimates:
##       cor
## 0.5558455
```

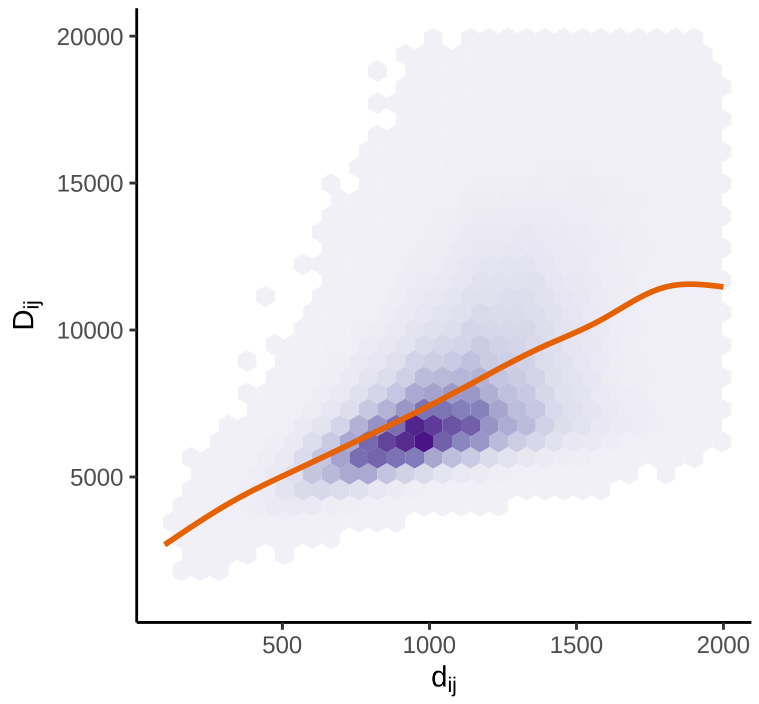
```
fig6
```

```
## Warning: Removed 360 rows containing non-finite outside the scale range
## (`stat_binhex()`).
```

```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```

```
## Warning: Removed 360 rows containing non-finite outside the scale range
## (`stat_smooth()`).
```

```
## Warning: Removed 13 rows containing missing values or values outside the scale range
## (`geom_hex()`).
```



```
pdf("figs_aux_2D.pdf")
fig6
```

```
## Warning: Removed 360 rows containing non-finite outside the scale range
## (`stat_binhex()`).
```

```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```

```
## Warning: Removed 360 rows containing non-finite outside the scale range
## (`stat_smooth()`).
## Removed 13 rows containing missing values or values outside the scale range
## (`geom_hex()`).
```

```
dev.off()
```

```
## pdf
##   2
```

Q3 : $D_{i,j} \sim d_{i,j}$ □

Check difference between generated and real prototypes

Load similarity scores and compute Euclidean distances for prototypes obtained from VDJ rearrangement model ("Murugan") or real-world repertoires ("Britanova"). Note that here we use CDR3β scores and omit Vβ, Jβ as they are quite predictable/discrete and are subject to batch effect/bias.

```r
read_proto <- function(prefix = "v_p", sz = 3000) {
  tmp <- read_tsv(paste0(prefix, sz, ".txt.gz")) |>
    select(cloneId, matches("b_\\d+_cdr3$")) |>
    melt(id.vars = "cloneId") |>
    dcast(cloneId ~ variable, mean)
  rownames(tmp) <- as.character(tmp$cloneId)
  tmp$cloneId <- NULL
  tmp |>
    as.matrix() |>
    dist() |>
    as.matrix() |>
    melt() |>
    rename(from = Var1, to = Var2, Dp = value)
}
v_p3000_d <- read_proto("v_p", 3000)
```

```
## Rows: 954 Columns: 18009
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr     (7): a_cdr3aa, a_v, a_j, b_cdr3aa, b_v, b_j, antigen.epitope
## dbl (18002): tcremp_id, cloneId, a_0_v, a_0_j, a_0_cdr3, a_1_v, a_1_j, a_1_c...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
v_b3000_d <- read_proto("v_b", 3000) |> rename(Db = Dp)
```

```
## Rows: 954 Columns: 18009
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr     (7): a_cdr3aa, a_v, a_j, b_cdr3aa, b_v, b_j, antigen.epitope
## dbl (18002): tcremp_id, cloneId, a_0_v, a_0_j, a_0_cdr3, a_1_v, a_1_j, a_1_c...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
v_bp3000_d <- merge(v_p3000_d, v_b3000_d)
```

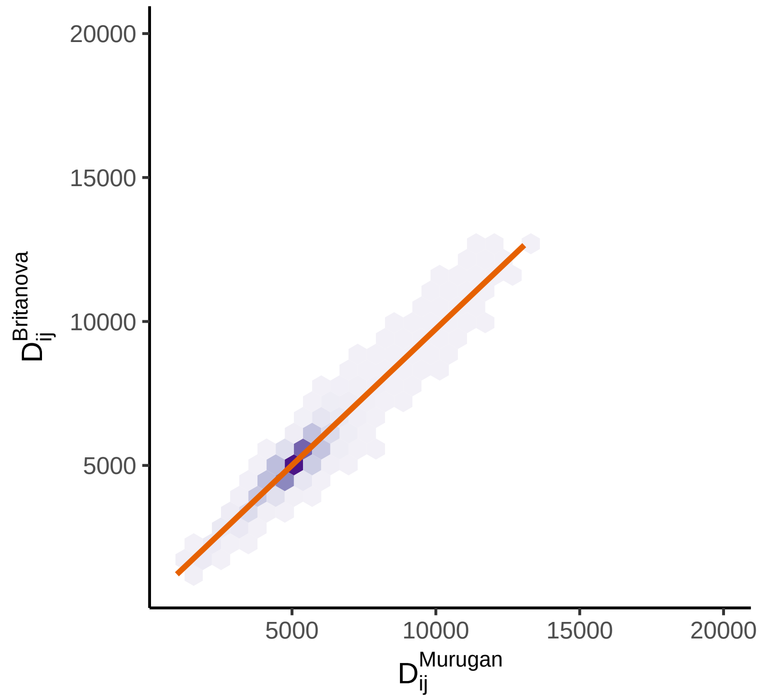
Plot and correlate

```r
fig7 <- v_bp3000_d |>
  ggplot(aes(x = Dp, y = Db)) +
  geom_hex(bins = 30) +
  scale_fill_distiller(palette = "Purples", direction = 1, guide = F) +
  geom_smooth(method = "lm", fill = NA, color = "#e66101") +
  scale_x_continuous(TeX("$D_{ij}^{Murugan}$"), limits = c(1000, 20000)) +
  scale_y_continuous(TeX("$D_{ij}^{Britanova}$"), limits = c(1000, 20000)) +
  theme_classic() +
  theme(aspect.ratio = 1)
with(v_bp3000_d,
     cor.test(Dp, Db))
```

```
##
##  Pearson's product-moment correlation
##
## data:  Dp and Db
## t = 2484.8, df = 560999, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.9572298 0.9576657
## sample estimates:
##       cor
## 0.9574483
```

```
fig7
```

```
## Warning: Removed 7917 rows containing non-finite outside the scale range
## (`stat_binhex()`).

## `geom_smooth()` using formula = 'y ~ x'

## Warning: Removed 7917 rows containing non-finite outside the scale range
## (`stat_smooth()`).

## Warning: Removed 1 row containing missing values or values outside the scale range
## (`geom_hex()`).
```



```r
pdf("figs_aux_3.pdf")
fig7
```

```
## Warning: Removed 7917 rows containing non-finite outside the scale range
## (`stat_binhex()`).

## `geom_smooth()` using formula = 'y ~ x'

## Warning: Removed 7917 rows containing non-finite outside the scale range
## (`stat_smooth()`).
## Removed 1 row containing missing values or values outside the scale range
## (`geom_hex()`).
```

```r
dev.off()
```

```
## pdf
##   2
```

Q4 : $D_{ij}^{Model} \sim D_{ij}^{RepSeq}$ □

Check effect fom the number of prototypes

Here we will again operate with CDR3β, look at the behavior of embeddings with $n_{proto} \in (100, 1000, 2000, 3000)$ prototypes.

```r
v_px <- bind_rows(
  read_proto("v_p", 100) |> mutate(n_prot = 100),
  read_proto("v_p", 1000) |> mutate(n_prot = 1000),
  read_proto("v_p", 2000) |> mutate(n_prot = 2000),
  read_proto("v_p", 3000) |> mutate(n_prot = 3000)
) |>
  left_join(read_tsv("v_p100.txt.gz") |>
              select(from = cloneId, ag = antigen.epitope) |>
              unique())
```

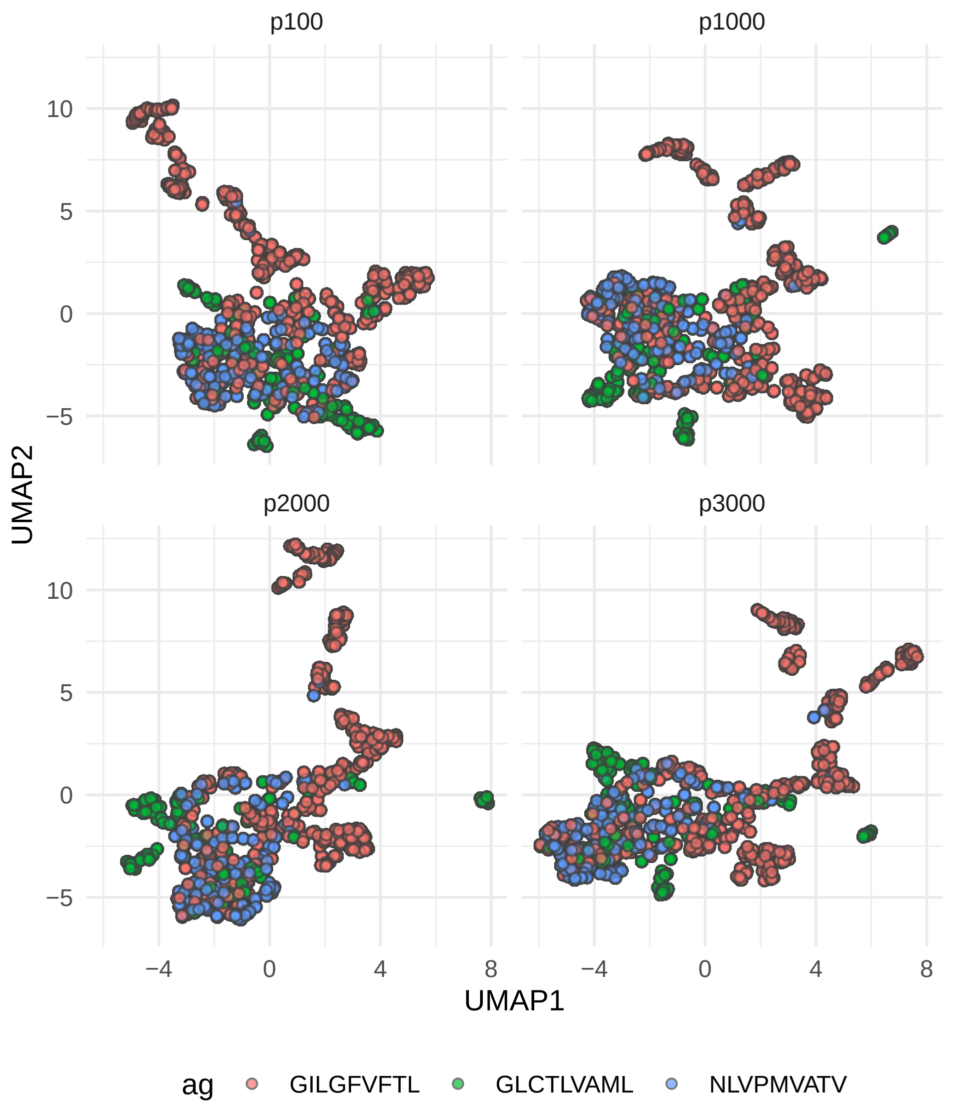
```
## Rows: 954 Columns: 609
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr   (7): a_cdr3aa, a_v, a_j, b_cdr3aa, b_v, b_j, antigen.epitope
## dbl (602): tcremp_id, cloneId, a_0_v, a_0_j, a_0_cdr3, a_1_v, a_1_j, a_1_cdr...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
## Rows: 954 Columns: 6009
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr    (7): a_cdr3aa, a_v, a_j, b_cdr3aa, b_v, b_j, antigen.epitope
## dbl (6002): tcremp_id, cloneId, a_0_v, a_0_j, a_0_cdr3, a_1_v, a_1_j, a_1_cd...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
## Rows: 954 Columns: 12009
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr     (7): a_cdr3aa, a_v, a_j, b_cdr3aa, b_v, b_j, antigen.epitope
## dbl (12002): tcremp_id, cloneId, a_0_v, a_0_j, a_0_cdr3, a_1_v, a_1_j, a_1_c...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
## Rows: 954 Columns: 18009
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr     (7): a_cdr3aa, a_v, a_j, b_cdr3aa, b_v, b_j, antigen.epitope
## dbl (18002): tcremp_id, cloneId, a_0_v, a_0_j, a_0_cdr3, a_1_v, a_1_j, a_1_c...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
## Rows: 954 Columns: 609
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr   (7): a_cdr3aa, a_v, a_j, b_cdr3aa, b_v, b_j, antigen.epitope
## dbl (602): tcremp_id, cloneId, a_0_v, a_0_j, a_0_cdr3, a_1_v, a_1_j, a_1_cdr...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
## Joining with `by = join_by(from)`
```

Do UMAP

```r
do_umap <- function(data) {
  config <- umap.defaults
  config$n_neighbors <- config$n_neighbors * 3
  config$min_dist <- config$min_dist * 3
  emb <- data |>
    dcast(from ~ to, value.var = "Dp") |>
    umap(config = config)
  res <- emb$layout |>
    as.tibble()
  colnames(res) <- c("UMAP1", "UMAP2")
  cbind(data |> select(from, ag), res)
}

v_px_umap <- v_px |>
  group_by(n_prot) |>
  group_modify(~ do_umap(.x))
```

```
## Warning: `as.tibble()` was deprecated in tibble 2.0.0.
## i Please use `as_tibble()` instead.
## i The signature and semantics have changed, see `?as_tibble`.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.

## Warning: The `x` argument of `as_tibble.matrix()` must have unique column names if
## `.name_repair` is omitted as of tibble 2.0.0.
## i Using compatibility `.name_repair`.
## i The deprecated feature was likely used in the tibble package.
##   Please report the issue at <https://github.com/tidyverse/tibble/issues>.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
fig8 <- v_px_umap |>
  mutate(n_prot = paste0("p", n_prot)) |>
  group_by(n_prot) |>
  mutate(sgn_GIL_1 = sign(sum(UMAP1 * startsWith(ag, "GIL"))),
         sgn_GIL_2 = sign(sum(UMAP2 * startsWith(ag, "GIL")))) |>
  ungroup() |>
  ggplot(aes(x = UMAP1 * sgn_GIL_1, y = UMAP2 * sgn_GIL_2, fill = ag)) +
  geom_point_rast(shape = 21,
                  color = "grey25",
                  alpha = 0.7) +
  facet_wrap(~n_prot) +
  scale_color_brewer(palette = "PuOr") +
  xlab("UMAP1") + ylab("UMAP2") +
  theme_minimal() +
  theme(aspect.ratio = 1,
        legend.position = "bottom")
fig8
```



```
pdf("figs_aux_4.pdf")
fig8
dev.off()
```

```
## pdf
##   2
```

Q5 : UMAP stabilizes for $n_{proto} \geq 1000$ ☐

```
print("             ")
```

```
## [1] "             "
```