In silico R4IDs simulation

Aim: establish likely performance.

Methods

A python script was used to simulate nanopore R9 sequencing runs from published **Cammelinae** genomes (~/Documents/all_work/programming/oddjects-sandbox/R4IDs/In-silico-genome-skimming-by-args.py). This used a mutation rate of 1/20 e.g. 5%. Uniformly-selected sites for mutation were assigned a substitution (equal base frequencies), point deletion, or point insertion (homopolymer created by inserting **n**th base as **n+1**th)

These read sets (intensities 10, 100, 1000 and 10000 reads) were used to simulate both R4IDs set-up and ID resequencing runs as for the empirical sci-fest data, and analysed in the same way.

Input reference genomes

Species	File	Mbp	# contigs
A. thaliana	()	120	7
Capsella rubella	file:////Users/joeparker/Downloads/ANNY01.1.fsa_nt.gz	129	7,067
A. halleri subsp. gemmifera	file:////Users/joeparker/Downloads/FJVB01.1.fsa_nt.gz	196	2,239
A. lyrata	()	208	3645
Capsella bursa-pastoris	file:////Users/joeparker/Downloads/MPGU01.1.fsa_nt.gz	268	8,186

Species	File	Mbp	# contigs
Camelina sativa	file:////Users/joeparker/Downloads/JFZQ01.1.fsa_nt.gz file:////Users/joeparker/Downloads/JFZQ01.2.fsa_nt.gz	641	37,780

Nanopore sequencing simulation

Analysis

```
library(lattice)
# old data
#in_silico = read.table('~/Documents/all_work/programming/oddjects-sandbox/R4IDs/manuscript-analyses/in

# new 2019 resimulated (more reps) data
in_silico = read.table('~/Documents/all_work/programming/oddjects-sandbox/R4IDs/manuscript-analyses/2019
```

How do stats' performance vary by species? List species in order.

First: *rate TP: total*

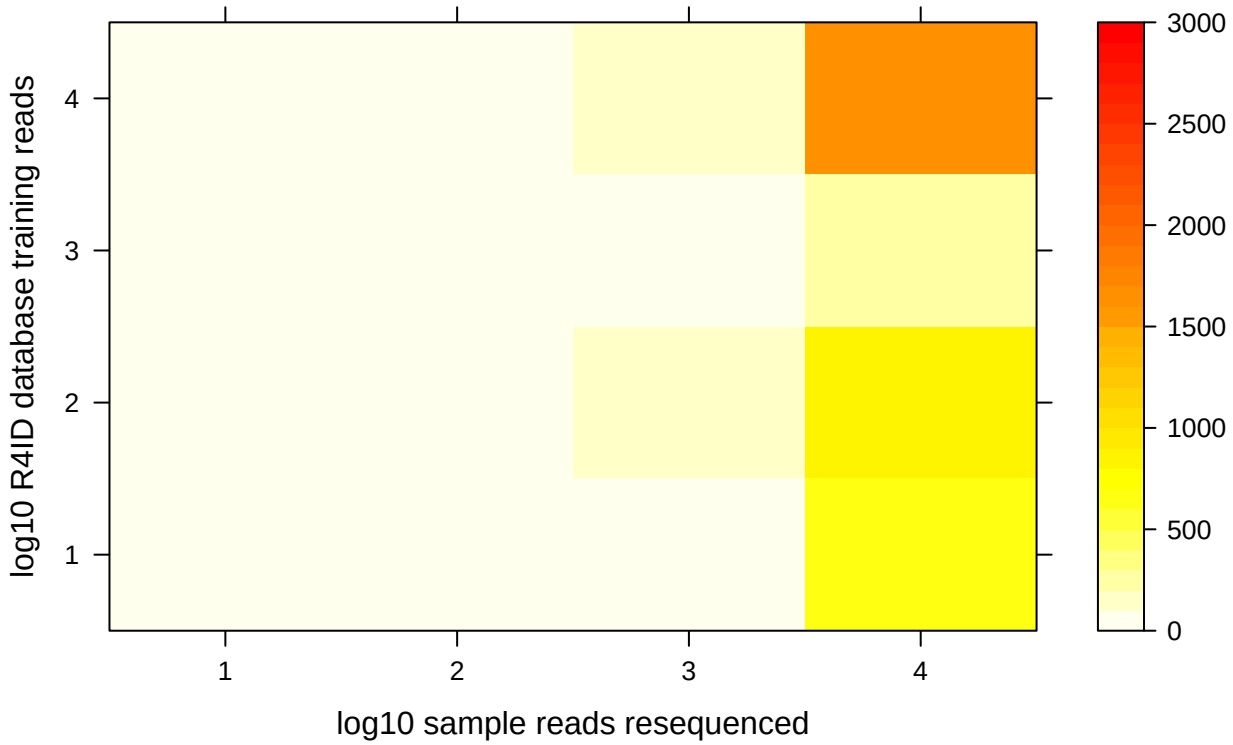
```
levelplot((one_way_TP + two_way_TP) / total_hits ~ log10(sample_DB_intensity)*log10(species_DB_intensity
levelplot((one_way_TP + two_way_TP) / total_hits ~ log10(sample_DB_intensity)*log10(species_DB_intensity
levelplot((one_way_TP + two_way_TP) / total_hits ~ log10(sample_DB_intensity)*log10(species_DB_intensity
levelplot((one_way_TP + two_way_TP) / total_hits ~ log10(sample_DB_intensity)*log10(species_DB_intensity
levelplot((one_way_TP + two_way_TP) / total_hits ~ log10(sample_DB_intensity)*log10(species_DB_intensity
levelplot((one_way_TP + two_way_TP) / total_hits ~ log10(sample_DB_intensity)*log10(species_DB_intensity
levelplot((one_way_TP + two_way_TP) / total_hits ~ log10(sample_DB_intensity)*log10(species_DB_intensity
```

Secondly *rate TP:FP*:

```
levelplot((one_way_TP + two_way_TP) / (one_way_FP + two_way_FP) ~ log10(sample_DB_intensity)*log10(spec
levelplot((one_way_TP + two_way_TP) / (one_way_FP + two_way_FP) ~ log10(sample_DB_intensity)*log10(spec
levelplot((one_way_TP + two_way_TP) / (one_way_FP + two_way_FP) ~ log10(sample_DB_intensity)*log10(spec
levelplot((one_way_TP + two_way_TP) / (one_way_FP + two_way_FP) ~ log10(sample_DB_intensity)*log10(spec
levelplot((one_way_TP + two_way_TP) / (one_way_FP + two_way_FP) ~ log10(sample_DB_intensity)*log10(spec
levelplot((one_way_TP + two_way_TP) / (one_way_FP + two_way_FP) ~ log10(sample_DB_intensity)*log10(spec
levelplot((one_way_TP + two_way_TP) / (one_way_FP + two_way_FP) ~ log10(sample_DB_intensity)*log10(spec
```

Thirdly *mean_bias*; first what is the aggregate length bias we can expect (as in the sci fest GUI)?

```
levelplot(two_way_rate*total_hits ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=in_sili
```




```
# boxplots by intensity to see confidence intervals
par(mfrow=c(1,2),oma=c(0,0,2,0))
boxplot(log10(two_way_rate*total_hits) ~ log10(sample_DB_intensity),data=in_silico,xlab='log10 sample r
```

```
## Warning in bplt(at[i], wid = width[i], stats = z$stats[, i], out =
## z$out[z$group == : Outlier (-Inf) in boxplot 1 is not drawn
```

```
## Warning in bplt(at[i], wid = width[i], stats = z$stats[, i], out =
## z$out[z$group == : Outlier (-Inf) in boxplot 2 is not drawn
```

```
## Warning in bplt(at[i], wid = width[i], stats = z$stats[, i], out =
## z$out[z$group == : Outlier (-Inf) in boxplot 3 is not drawn
```

```
## Warning in bplt(at[i], wid = width[i], stats = z$stats[, i], out =
## z$out[z$group == : Outlier (-Inf) in boxplot 4 is not drawn
```

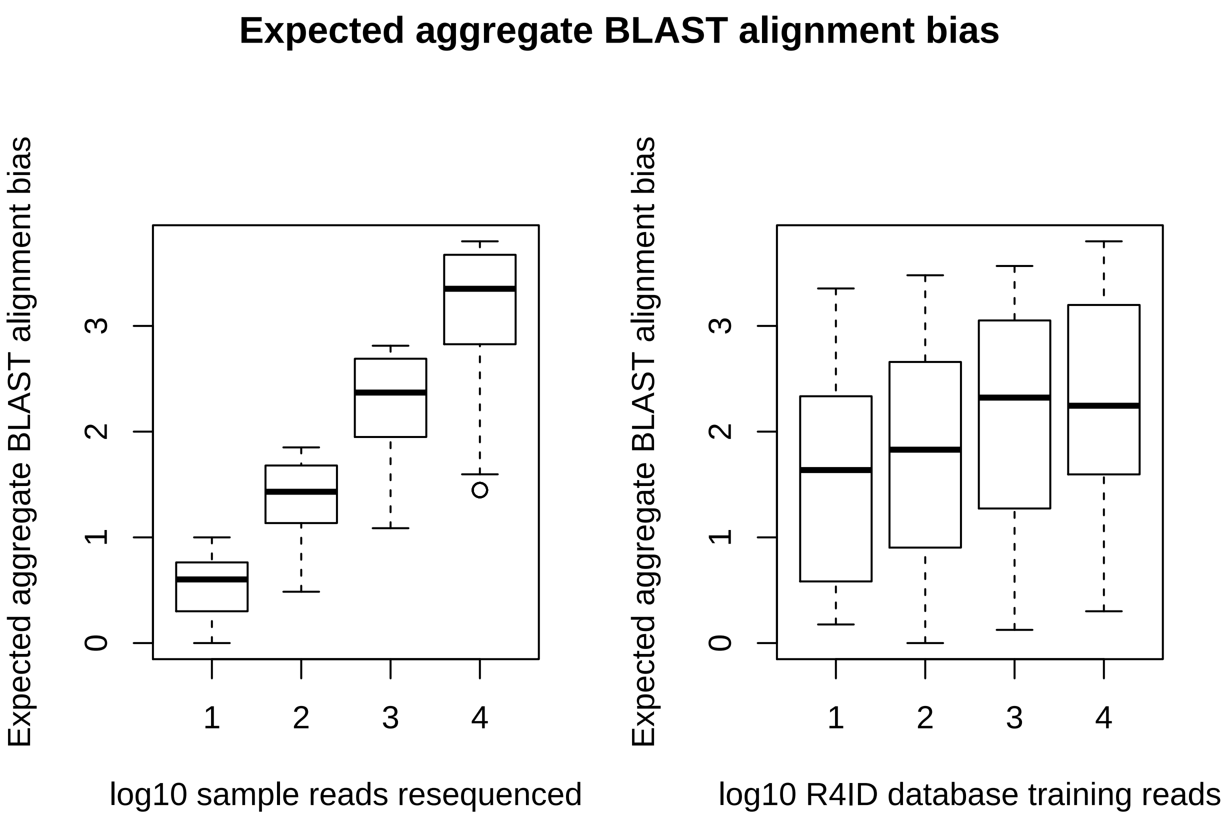
```
boxplot(log10(two_way_rate*total_hits) ~ log10(species_DB_intensity),data=in_silico,xlab='log10 R4ID da
```

```
## Warning in bplt(at[i], wid = width[i], stats = z$stats[, i], out =
## z$out[z$group == : Outlier (-Inf) in boxplot 1 is not drawn
```

```
## Warning in bplt(at[i], wid = width[i], stats = z$stats[, i], out =
## z$out[z$group == : Outlier (-Inf) in boxplot 2 is not drawn
```

```
## Warning in bplt(at[i], wid = width[i], stats = z$stats[, i], out =
## z$out[z$group == : Outlier (-Inf) in boxplot 3 is not drawn
```

```
title("Expected aggregate BLAST alignment bias",outer=T)
```



And the bias itself?

```
levelplot(log10(mean_bias) ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=in_silico, col
levelplot(mean_bias ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=in_silico[in_silico$TI
levelplot(mean_bias ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=in_silico[in_silico$TI
levelplot(mean_bias ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=in_silico[in_silico$TI
levelplot(mean_bias ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=in_silico[in_silico$TI
levelplot(mean_bias ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=in_silico[in_silico$TI
levelplot(mean_bias ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=in_silico[in_silico$TI
```

Fourth *hit rate in 2-way assignments*

```
levelplot(two_way_rate ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=in_silico, col.reg
levelplot(two_way_rate ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=in_silico[in_silic
levelplot(two_way_rate ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=in_silico[in_silic
levelplot(two_way_rate ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=in_silico[in_silic
levelplot(two_way_rate ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=in_silico[in_silic
levelplot(two_way_rate ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=in_silico[in_silic
levelplot(two_way_rate ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=in_silico[in_silic
```

Fifth *hit rate in 2-way assignments, with cutoff>50*

```
levelplot(two_way_rate_with_cutoff ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=in_sil
levelplot(two_way_rate_with_cutoff ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=in_sil
levelplot(two_way_rate_with_cutoff ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=in_sil
levelplot(two_way_rate_with_cutoff ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=in_sil
levelplot(two_way_rate_with_cutoff ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=in_sil
levelplot(two_way_rate_with_cutoff ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=in_sil
levelplot(two_way_rate_with_cutoff ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=in_sil
```

Finally *total hits*

```
levelplot(total_hits ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=in_silico, col.region
levelplot(total_hits ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=in_silico[in_silico$T
levelplot(total_hits ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=in_silico[in_silico$T
levelplot(total_hits ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=in_silico[in_silico$T
levelplot(total_hits ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=in_silico[in_silico$T
levelplot(total_hits ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=in_silico[in_silico$T
levelplot(total_hits ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=in_silico[in_silico$T
```

Okay, this is a little hard to interpret; now plot hit percentage (expectation) * total hits, e.g. number of expected positives:

```
#levelplot(total_hits * two_way_rate_with_cutoff ~ log10(sample_DB_intensity)*log10(species_DB_intensit
levelplot(total_hits * two_way_rate_with_cutoff ~ log10(sample_DB_intensity)*log10(species_DB_intensity)
levelplot(total_hits * two_way_rate_with_cutoff ~ log10(sample_DB_intensity)*log10(species_DB_intensity)
#levelplot(total_hits * two_way_rate_with_cutoff ~ log10(sample_DB_intensity)*log10(species_DB_intensit
levelplot(total_hits * two_way_rate_with_cutoff ~ log10(sample_DB_intensity)*log10(species_DB_intensity)
#levelplot(total_hits * two_way_rate_with_cutoff ~ log10(sample_DB_intensity)*log10(species_DB_intensit
levelplot(total_hits * two_way_rate_with_cutoff ~ log10(sample_DB_intensity)*log10(species_DB_intensity)
```

Formally analyse the relationship with a GLM:

```
# 2-way with cutoff, by species too
model_total_positives=glm(total_hits * two_way_rate_with_cutoff ~ (sample_DB_intensity)*(species_DB_inte
summary.glm(model_total_positives)
summary(aov(model_total_positives))
# two-way rate with cutoff, no species
model_total_positives_nosp=glm(total_hits * two_way_rate_with_cutoff ~ (sample_DB_intensity)*(species_DB
summary(aov(model_total_positives_nosp))
summary(model_total_positives_nosp)
# 2-way, by species too
model_total_positives=glm(total_hits * two_way_rate ~ (sample_DB_intensity)*(species_DB_intensity)*TP_sp
summary(aov(model_total_positives))
# two-way rate with cutoff, no species
model_total_positives_nosp=glm(total_hits * two_way_rate ~ (sample_DB_intensity)*(species_DB_intensity)
summary(aov(model_total_positives_nosp))
summary(model_total_positives_nosp)
```

And what we might really want to know is the expectation that a read in a sequencing run of length n will be a positive:

```
levelplot((total_hits * two_way_rate)/sample_DB_intensity ~ log10(sample_DB_intensity)*log10(species_DB_
levelplot((total_hits * two_way_rate_with_cutoff)/sample_DB_intensity ~ log10(sample_DB_intensity)*log10
levelplot(((total_hits * ((one_way_TP + two_way_TP)/(one_way_FP + two_way_FP+one_way_TP+two_way_TP)))/sa
levelplot((( ((one_way_TP + two_way_TP)/(one_way_FP + two_way_FP)))/sample_DB_intensity) ~ log10(sample_
levelplot((total_hits /sample_DB_intensity) ~ log10(sample_DB_intensity)*log10(species_DB_intensity),da
```

Let's break down the expected number of good hits a bit more simply:

```
par(mfrow=c(1,2))
boxplot((( ((one_way_TP + two_way_TP)))/sample_DB_intensity) ~ log10(species_DB_intensity),data=in_silic
boxplot((( ((one_way_TP + two_way_TP)))/sample_DB_intensity) ~ log10(sample_DB_intensity),data=in_silic
```

As above but using two-way hit rate:

```
par(mfrow=c(1,2))
boxplot((total_hits * two_way_rate)/sample_DB_intensity ~ log10(species_DB_intensity),data=in_silico, co
boxplot((total_hits * two_way_rate)/sample_DB_intensity ~ log10(sample_DB_intensity),data=in_silico, col
```

As above but using two-way hit rate, with cutoff > 50:

```
par(mfrow=c(1,2))
boxplot((total_hits * two_way_rate_with_cutoff)/sample_DB_intensity ~ log10(species_DB_intensity),data=
boxplot((total_hits * two_way_rate_with_cutoff)/sample_DB_intensity ~ log10(sample_DB_intensity),data=i
```

As above but accuracy e.g. [(TP:TP+FP) / total reads]. This is probably closest to the headline 'effective ID rate', e.g. expectation that a given sequenced read produces a BLAST hit which is accurately a true positive.

```
#par(mfrow=c(1,2))
# TP rate in red
boxplot(((total_hits * ((one_way_TP + two_way_TP)/(one_way_FP + two_way_FP+one_way_TP+two_way_TP)))/samp
# FP rate in blue
boxplot(((total_hits * ((one_way_FP + two_way_FP)/(one_way_FP + two_way_FP+one_way_TP+two_way_TP)))/samp
legend(0.5,0.9,'TP fraction',fill='pink')
legend(0.5,0.7,'FP fraction',fill='light blue')
```

Of course we should ask the reciprocal question e.g. [(FP:TP+FP) / total reads], expectation that a given sequenced read produces a BLAST hit which is inaccurately a false positive.

```
par(mfrow=c(1,2))
boxplot(((total_hits * ((one_way_FP + two_way_FP)/(one_way_FP + two_way_FP+one_way_TP+two_way_TP)))/samp
boxplot(((total_hits * ((one_way_FP + two_way_FP)/(one_way_FP + two_way_FP+one_way_TP+two_way_TP)))/samp
```