Krause/Niazi et al. DNA Data Analysis

Adnan M. Niazi & Maximillian Krause

5/13/2019

This document contains all the analyses done on ONT DNA data that was generated for the tailfindr paper. Knit this R markdown file after you have successfully run `drake::r_make()`.

Load the required libraries first:

```
pacman::p_load(dplyr, magrittr, ggplot2, drake, knitr, ggpubr, here, tidyverse)
```

Now load the data:

```
loadd(dna_kr_data)
```

Here is a description of columns of `dna_kr_data`:

```
knitr::kable(col_names_df)
```

Columns	Description
read_id	Read ID
tail_start_ff	tailfindr estimate of poly(A) start site based on flipflop basecalling
tail_end_ff	tailfindr estimate of poly(A) end site based on flipflop basecalling
samples_per_nt_ff	tailfindr estimate of read-specific translocation rate in units of samples per nucleotide bas
tail_length_ff	tailfindr estimate of poly(A) tail length based on flipflop basecalling
tail_start_st	tailfindr estimate of poly(A) start site from standard model basecalling
tail_end_st	tailfindr estimate of poly(A) end site from standard model basecalling
samples_per_nt_st	tailfindr estimate of read-specific translocation rate in units of samples per nucleotide fro
tail_length_st	tailfindr estimate of poly(A) tail length from standard model basecalling
read_type	Whether the read is poly(A) or poly(T) read
barcode	Expected poly(A)/(T) tail length from spikeins
replicate	Replicate No
file_path	Full file path (relevant only for use within Valen lab)
transcript_alignment_start_st	Location of tail end by eGFP sequence alignment (standard model basecalling)
transcript_alignment_start_ff	Location of tail end by eGFP sequence alignment (flipflop model basecalling

Data summary

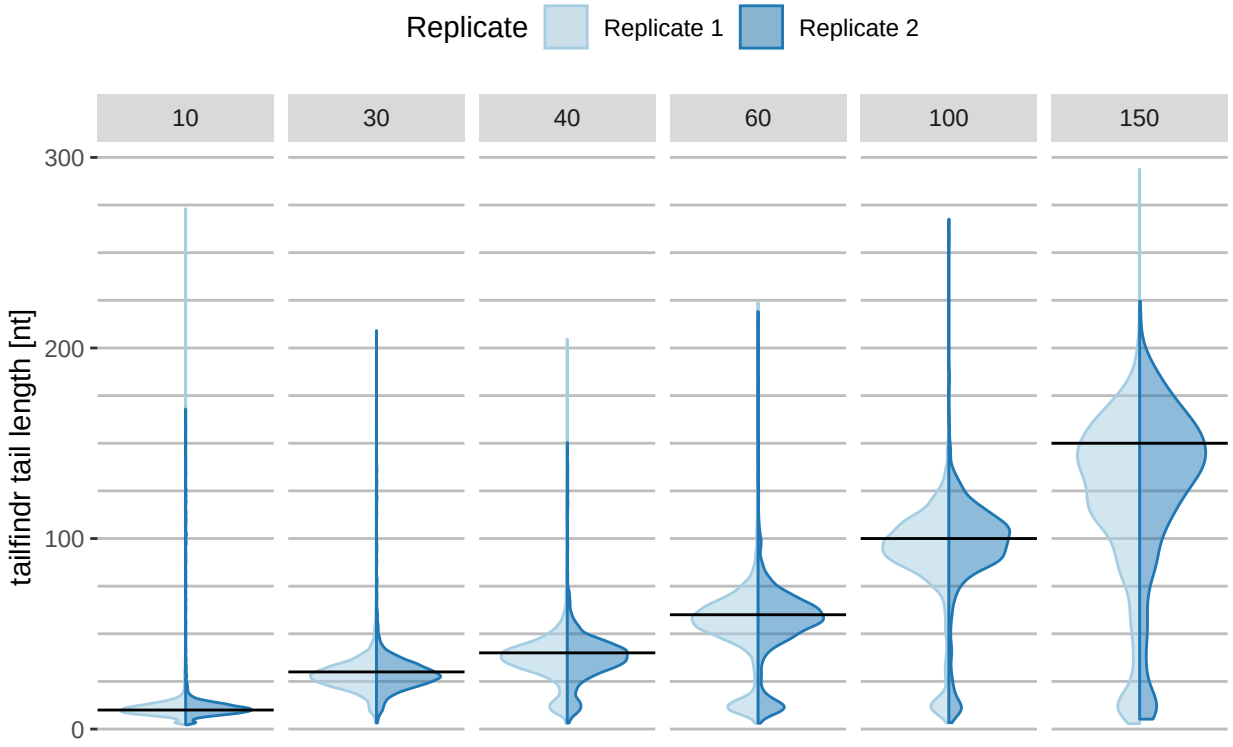
```
# define the function for computing standard error
std_err <- function(x) sd(x, na.rm = TRUE)/sqrt(length(x))

# summarize the data and display a table
summary_data <- dna_kr_data %>% group_by(barcode, read_type) %>%
  summarise(read_count = n(),
            mean = mean(tail_length_st, na.rm = TRUE),
            median = median(tail_length_st, na.rm = TRUE),
            std_dev = sd(tail_length_st, na.rm = TRUE),
            std_err = std_err(tail_length_st))
summary_data %<>% mutate(cof_var = std_dev/mean)
kable(summary_data)
```

barcode	read_type	read_count	mean	median	std_dev	std_err	cof_var
10	polyA	3850	12.83470	9.27000	15.25476	0.2458528	1.1885568
10	polyT	11072	14.27321	10.95000	17.32021	0.1646039	1.2134766
30	polyA	12858	28.42722	27.39000	10.49006	0.0925106	0.3690147
30	polyT	17087	29.92357	28.60000	14.00484	0.1071384	0.4680201
40	polyA	6826	36.56306	36.57500	12.83846	0.1553925	0.3511321
40	polyT	13811	37.35623	38.01000	16.03852	0.1364746	0.4293400
60	polyA	8065	53.56193	55.47000	16.73813	0.1863824	0.3125005
60	polyT	10073	51.64352	57.56465	24.32128	0.2423299	0.4709454
100	polyA	2959	89.93859	92.72000	22.34206	0.4107246	0.2484146
100	polyT	3167	89.88751	99.29374	34.44611	0.6120912	0.3832135
150	polyA	1693	121.62783	132.27000	39.53762	0.9609086	0.3250705
150	polyT	2535	117.56694	128.08000	49.71003	0.9873135	0.4228232

tailfindr tail length estimation across replicates

To find out how robust the tail length estimated by tailfindr is across technical replicates:

```
p <- ggplot(dna_kr_data, aes(x = barcode, y = tail_length_st,
                             color = replicate, fill = replicate)) +
  geom_two_sided_flat_violin(position = position_nudge(x = 0, y = 0), alpha = .5) +
  facet_grid(~barcode, scales = 'free') +
  geom_hline(aes(yintercept = as.numeric(as.character(barcode))))
```

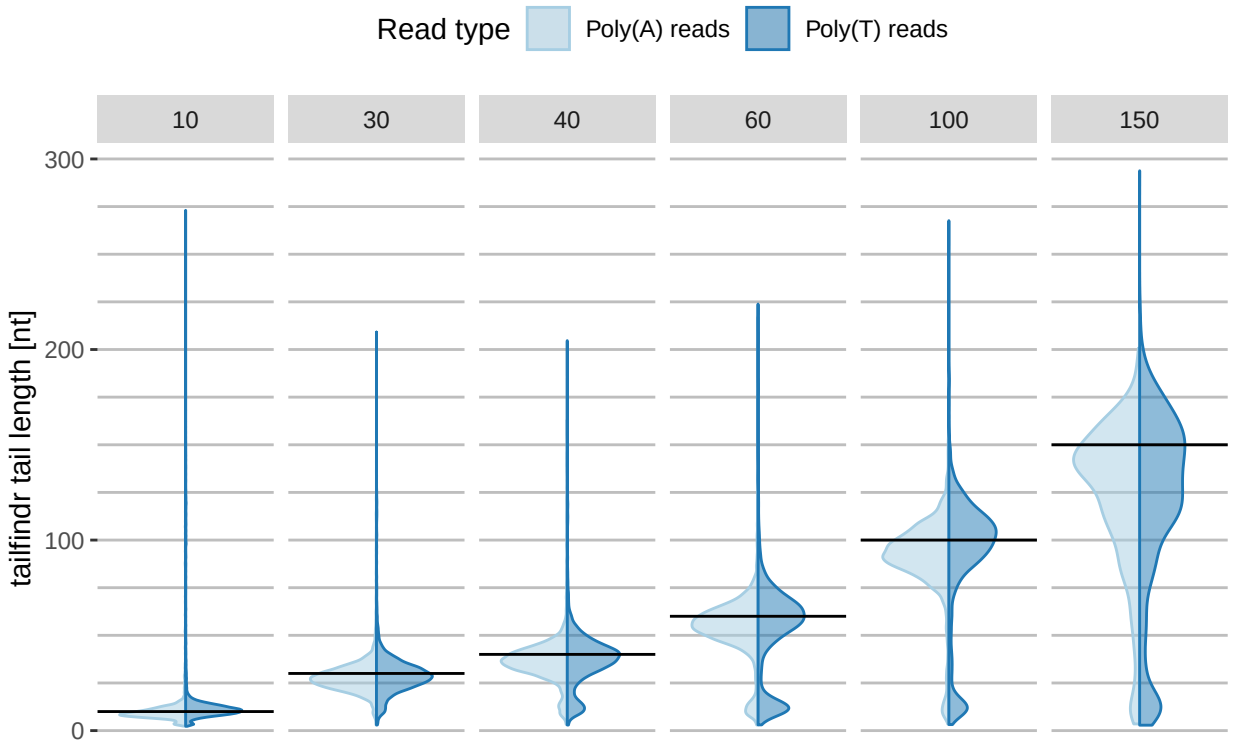


tailfindr tail length estimate is robust across technical replicates.
The black horizontal lines represent the expected tail length.

tailfindr tail length comparison between poly(A) and poly(T) read types

To address whether estimated tail lengths of poly(A) and poly(T) reads are comparable:

```
p <- ggplot(dna_kr_data, aes(x = barcode, y = tail_length_st,
                             color = read_type, fill = read_type)) +
  geom_two_sided_flat_violin(position = position_nudge(x = 0, y = 0), alpha = .5) +
  facet_grid(~barcode, scales = 'free') +
  geom_hline(aes(yintercept = as.numeric(as.character(barcode))))
```

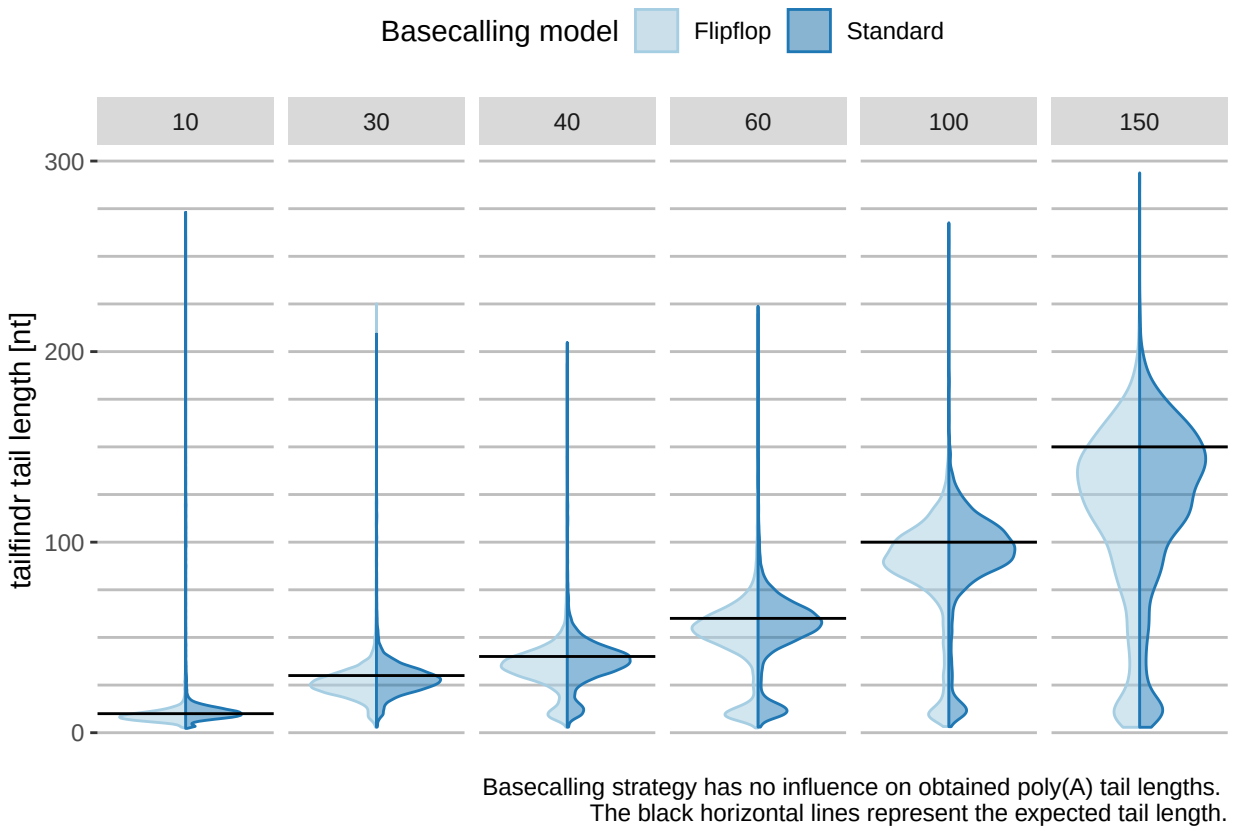


tailfindr tail length estimate is robust across read types.
The black horizontal lines represent the expected tail length.

tailfindr tail length comparison between flipflop and standard basecalling

To test whether basecalling strategy (standard model vs flip-flop model basecalling) has an influence on poly(A) length estimation:

```
# make data first
long_dna_data <- dna_kr_data %>% select(barcode, tail_length_ff, tail_length_st) %>%
  gather(key = 'basecaller', value = 'tail_length', tail_length_ff, tail_length_st)

p <- ggplot(long_dna_data, aes(x = barcode, y = tail_length,
                             color = basecaller, fill = basecaller)) +
  geom_two_sided_flat_violin(position = position_nudge(x = 0, y = 0), alpha = .5) +
  facet_grid(~barcode, scales = 'free') +
  geom_hline(aes(yintercept = as.numeric(as.character(barcode))))
```



Basecalling strategy has no influence on obtained poly(A) tail lengths.
The black horizontal lines represent the expected tail length.

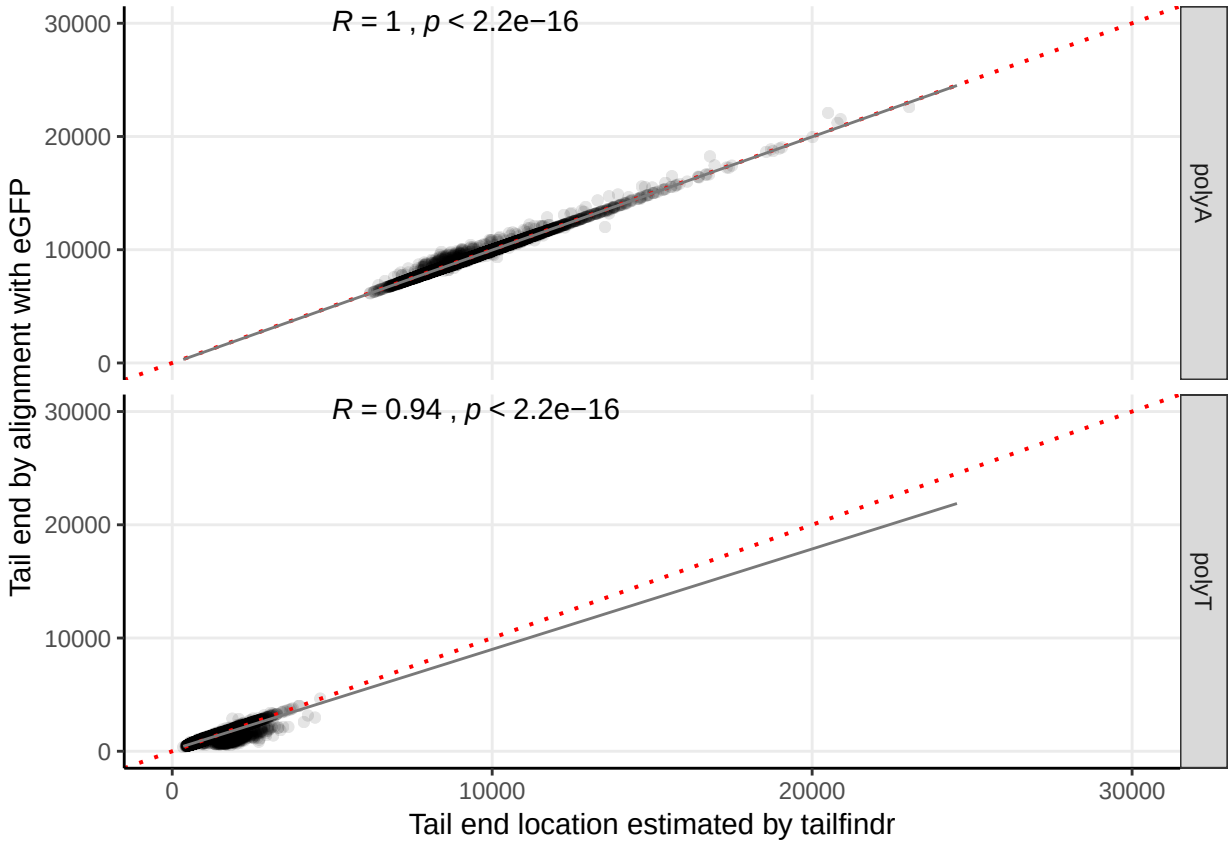
tailfindr tail end estimate vs. tail end obtained by alignment of eGFP

First a new column `transcript_end_tf`is produced in our dataset. This column holds the tailfindr boundary location which is adjacent to transcript:

```
dna_kr_data %<>%
  mutate(transcript_end_tf = ifelse(read_type == 'polyA',
                                    tail_start_ff, tail_end_ff))
```

To visualize whether the coordinates of the tail end estimated by tailfindr match up with those obtained from the alignment with eGFP sequence:

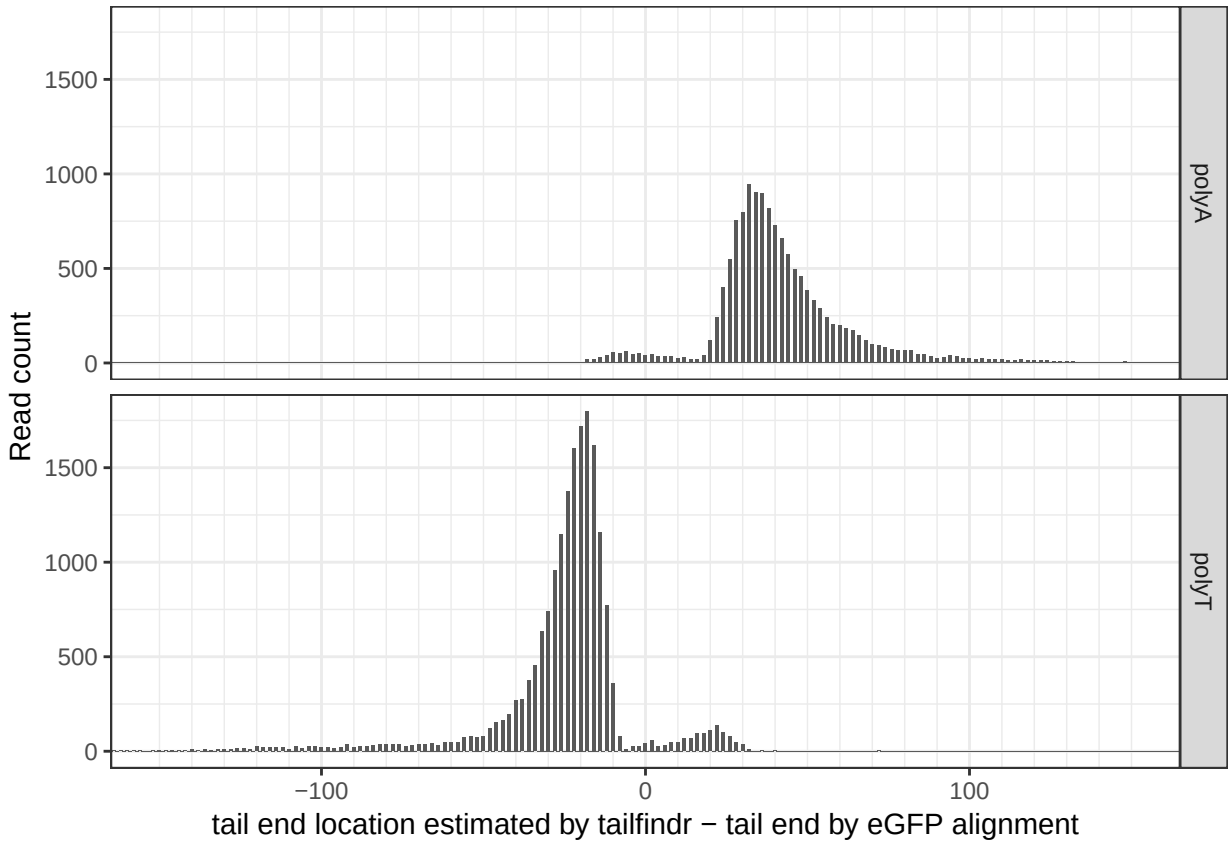
```
p <- ggplot(dna_kr_data, aes(x = transcript_end_tf, y = transcript_alignment_start_ff)) +
    geom_point(shape = 21, colour = 'black', fill = 'black', size = 2, stroke=0, alpha = 0.1) +
    geom_abline(intercept = 0, slope = 1, color="red", linetype = 'dotted', size = 0.7) +
    geom_smooth(method = 'lm',formula = y~x, color="#797979", fullrange = TRUE, se = FALSE, size = 0.5)
    stat_cor(method = "pearson", label.x = 5000, label.y = 30000) +
    coord_cartesian(xlim = c(0, 30000), ylim = c(0, 30000)) +
  facet_grid(read_type~.)
```



To better visulize the difference, the same information is plotted as histogram:

```
hist_data <- mutate(dna_kr_data, diff = transcript_end_tf - transcript_alignment_start_ff)
p <- ggplot(hist_data, aes(x = diff)) +
  geom_histogram(binwidth = 1) +
  facet_grid(read_type~.)
```

```
p <- p +
  theme_bw() +
  coord_cartesian(xlim = c(-150, 150)) +
  scale_x_continuous(minor_breaks = seq(-150, 150, 10)) +
  xlab('tail end location estimated by tailfindr - tail end by eGFP alignment') +
  ylab('Read count')
p
```



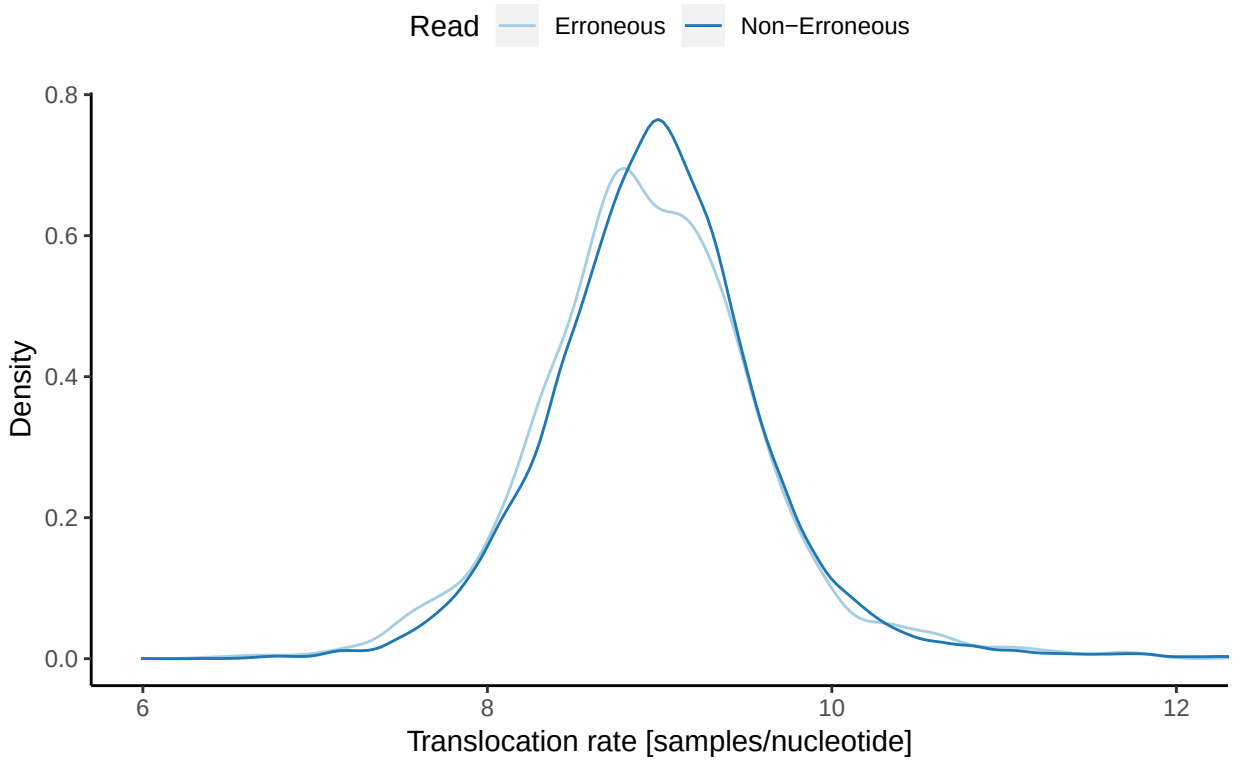
Analysis of spurious peak around 12 in the density plots

First, classify the reads into erroneous and non-erroneous read

```
# Erroneous reads have tail length between 9 and 15
spurious_peak_data <- dna_kr_data %>%
  filter(barcode == 60 | barcode == 100 | barcode == 150) %>%
  mutate(read_classification =
           if_else(tail_length_st >= 9 & tail_length_st <= 15,
                   'erroneous',
                   'non-erroneous'))
```

a. Read rate density from erroneous vs non-erroneous reads

```
p <- ggplot(spurious_peak_data, aes(x = samples_per_nt_st,
                                    color = read_classification)) +
  geom_line(stat = 'density')
```

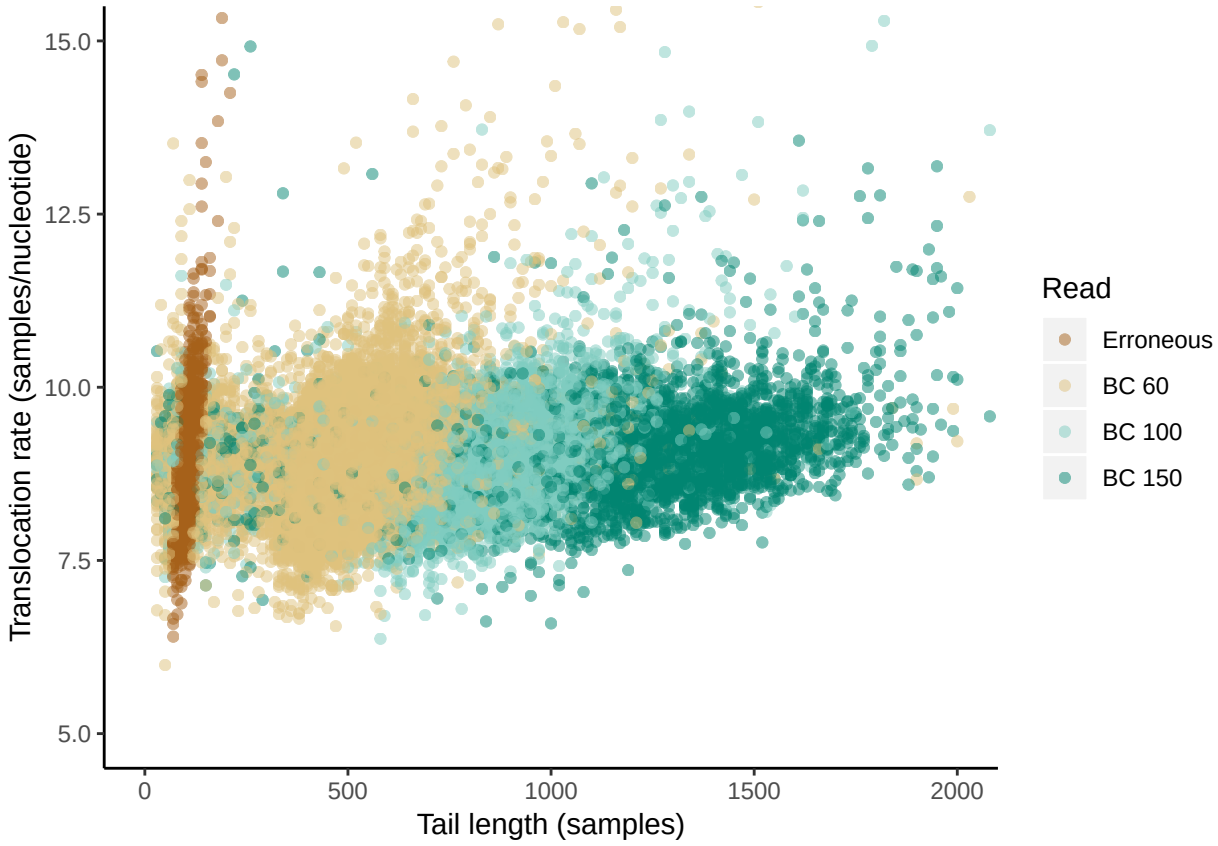


Both erroneous and non−erroneous reads have the same nucleotide translocation rate profiles

b. Scatter plot X axis raw tail length vs normaliser y axis

```
spurious_peak_data %<>%
  mutate(tail_length_in_samples_st = tail_end_st - tail_start_st) %>%
  mutate(barcode = as.character(barcode)) %>%
  mutate(barcode = ifelse(read_classification == 'erroneous', 'erroneous', barcode)) %>%
  mutate(barcode = fct_relevel(barcode, "erroneous", "60", "100", "150"))

p <- ggplot(spurious_peak_data, aes(x = tail_length_in_samples_st,
                                    y = samples_per_nt_st,
                                    color = barcode)) +
  geom_point(alpha = 0.5, stroke = 0, size = 2)
```

```
lengend_name <- 'Read'
legend_labels <- c('Erroneous', 'BC 60', 'BC 100', 'BC 150')
p <- p + theme(
  panel.background = element_blank(),
  axis.line = element_line(colour = 'black', size = 0.5)) +
    scale_color_brewer(palette = "BrBG",
                       name = lengend_name,
                       labels = legend_labels) +
  xlab('Tail length (samples)') +
  ylab('Translocation rate (samples/nucleotide)') +
  coord_cartesian(xlim = c(0, 2000), ylim = c(5, 15))
p
```



c. eGFP alignment end vs tailfindr end on erroneoush vs non-erroneous reads

```
spurious_peak_data %<>%
  mutate(transcript_end_st = ifelse(read_type == 'polyA',
                                    tail_start_st,
                                    tail_end_st)) %>%
  mutate(diff = transcript_end_st - transcript_alignment_start_st) %>%
  mutate(
    read_classification =
      case_when(
        read_type == 'polyA' &
          read_classification == 'erroneous' ~ "erroneous_polya",
        read_type == 'polyA' &
          read_classification == 'non-erroneous' ~ "non-erroneous_polya",
        read_type == 'polyT' &
          read_classification == 'erroneous' ~ "erroneous_polyt",
        read_type == 'polyT' &
          read_classification == 'non-erroneous' ~ "non-erroneous_polyt"
      )
  ) %>%
  mutate(read_classification = fct_relevel(read_classification,
                                           "erroneous_polya",
                                           "non-erroneous_polya",
                                           "erroneous_polyt",
                                           "non-erroneous_polyt"))
```

```
p <- ggplot(spurious_peak_data, aes(x = diff,
                                    color = read_classification)) +
  geom_line(stat = 'density', size = 0.9)
```

```
lengend_name <- 'Read'
legend_labels <- c('Erroneous Poly(A)',
                   'Non-Erroneous Poly(A)',
                   'Erroneous Poly(T)',
                   'Non-Erroneous Poly(T)')
p <- p + theme(
  panel.background = element_blank(),
  axis.line = element_line(colour = 'black', size = 0.5)) +
    scale_color_brewer(palette = "Paired",
                       name = lengend_name,
                       labels = legend_labels) +
  xlab('tailfindr tail end - sequence end defined by eGFP alignment') +
  ylab('Density') +
  coord_cartesian(xlim = c(-100, 100))
p
```