Results

Table of contents


```
library(yaml)
library(here)
```

```
here() starts at /root/repo
```

```
fmt3 <- function(x) sprintf("%.3f", x)
fmt6 <- function(x) sprintf("%.6f", x)

cleaning <- yaml::read_yaml(here("outputs", "results", "cleaning.yml"))
base <- yaml::read_yaml(here("outputs", "results", "base_lm.yml"))
freq <- yaml::read_yaml(here("outputs", "results", "frequency_effect.yml"))
```

Cleaning

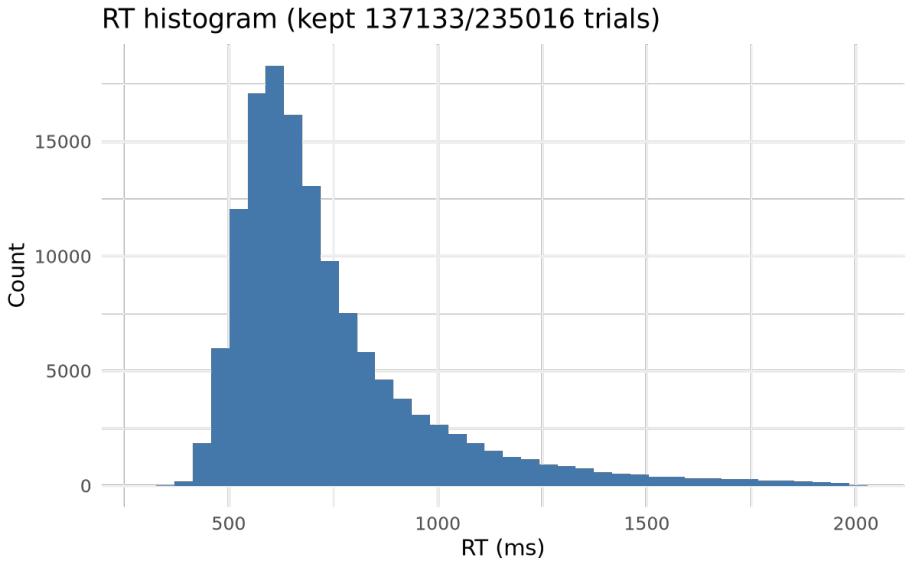
The pipeline kept 137133 of 235016 trials (dropped 97883). Settings: correct-only = TRUE, RT range = 200–2000 ms.

```
data.frame(
  setting = c(
    "correct_only",
    "rt_min_ms",
    "rt_max_ms",
    "total_trials",
    "kept_trials",
    "dropped_trials"
  ),
  value = c(
    as.character(cleaning$trimming$correct_only),
    cleaning$trimming$rt_min_ms,
    cleaning$trimming$rt_max_ms,
    cleaning$counts$total_trials,
```

```
    cleaning$counts$kept_trials,
    cleaning$counts$dropped_trials
  )
)
```

```
         setting  value
1   correct_only   TRUE
2      rt_min_ms    200
3      rt_max_ms   2000
4   total_trials 235016
5    kept_trials 137133
6 dropped_trials  97883
```

```
knitr::include_graphics(here("outputs", "figures", "rt_hist.png"))
```



Baseline model: frequency and strokes

```
data.frame(
  term = c("intercept", "log_freq", "strokes"),
  estimate = c(
    fmt6(as.numeric(base$coefficients$intercept)),
    fmt6(as.numeric(base$coefficients$log_freq)),
    fmt6(as.numeric(base$coefficients$strokes))
  )
)
```

```
       term  estimate
```

```
1 intercept  6.452355
2  log_freq -0.070823
3   strokes  0.013355
```

R^2 0.434; adjusted R^2 0.433; residual sigma 0.099. AIC -6851.160, BIC -6826.134.

Frequency effect on recognition speed

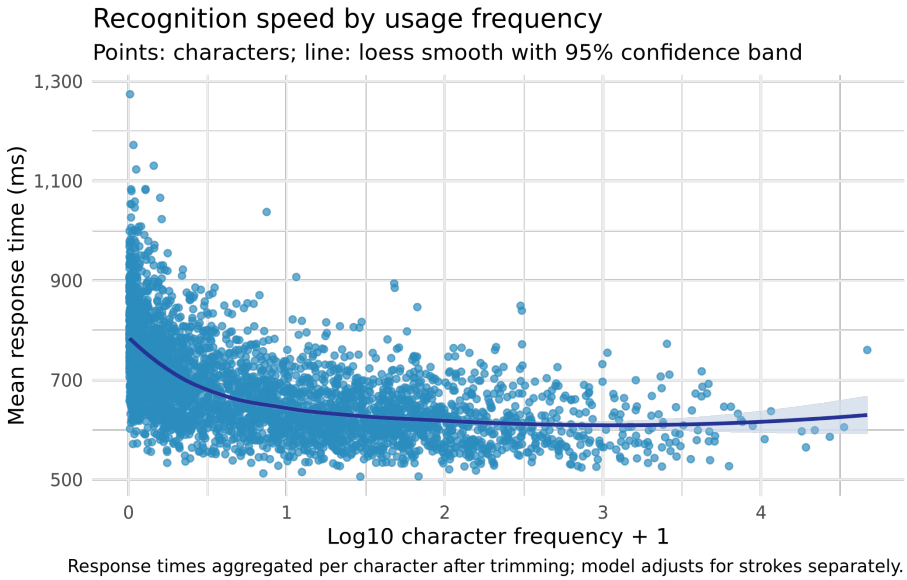
```
freq_terms <- names(freq$coefficients)
freq_df <- data.frame(
  term = freq_terms,
  estimate = sapply(freq$coefficients, function(x) fmt6(as.numeric(x$estimate))),
  conf_low = sapply(freq$coefficients, function(x) fmt6(as.numeric(x$conf_low))),
  conf_high = sapply(freq$coefficients, function(x) fmt6(as.numeric(x$conf_high)))
)
freq_df
```

```
                  term  estimate  conf_low conf_high
(Intercept) (Intercept)  6.452355  6.441170  6.463540
log_freq       log_freq -0.070823 -0.074644 -0.067002
strokes         strokes  0.013355  0.012417  0.014294
```

```
freq_effect <- freq$coefficients$log_freq
freq_multiplier <- exp(as.numeric(freq_effect$estimate))
freq_multiplier_low <- exp(as.numeric(freq_effect$conf_low))
freq_multiplier_high <- exp(as.numeric(freq_effect$conf_high))
freq_pct_drop <- (1 - freq_multiplier) * 100
freq_pct_drop_low <- (1 - freq_multiplier_high) * 100
freq_pct_drop_high <- (1 - freq_multiplier_low) * 100
pred_rt <- freq$reference$predicted_rt_ms
freq_quantiles <- freq$reference$log_freq_quantiles
```

A 1-unit increase in log10 frequency (roughly a tenfold usage jump) is associated with an estimated 6.837% faster recognition (95% CI 6.481% to 7.193%) when strokes are held at their median (10). This corresponds to predicted mean response times of approximately 723.216 ms (10th percentile frequency), 694.402 ms (median), and 624.486 ms (90th percentile). The marginal gains are largest through common characters—going from the median to the 90th percentile yields 69.916 ms faster responses—but they taper for the most extreme items: the 10th to 50th percentile shift saves about 28.814 ms, and the jump from the 90th to 95th percentile trims only 18.729 ms.

```
knitr::include_graphics(here(freq$figure))
```



A loess smooth over the character-level means reinforces the flattening: after log10 frequency around 2.104, the curve levels off with uncertainty bands that overlap tightly, suggesting diminishing recognition-speed returns for the very most frequent characters.