Agentic AI Demo Report

Table of contents

```r
library(yaml)
library(readr)

metrics_path <- here::here("outputs","results","metrics.yml")
cleaning_path <- here::here("outputs","results","cleaning.yml")
fig_path <- here::here("outputs","figures","rt_hist.png")
freq_tidy_path <- here::here("outputs","results","character_frequency_model_tidy.csv")
freq_glance_path <- here::here("outputs","results","character_frequency_model_glance.csv")
freq_fig_path <- here::here("outputs","figures","character_frequency_rt_vs_freq.png")
complexity_tidy_path <- here::here("outputs","results","visual_complexity_model_tidy.csv")
complexity_glance_path <- here::here("outputs","results","visual_complexity_model_glance.cs
complexity_summary_path <- here::here("outputs","results","visual_complexity_penalty_summary
complexity_partial_path <- here::here("outputs","data","visual_complexity_partial_effect.cs
complexity_fig_path <- here::here("outputs","figures","visual_complexity_partial_effect.png"

metrics <- yaml::read_yaml(metrics_path)
cleaning <- yaml::read_yaml(cleaning_path)
freq_model_tidy <- read_csv(freq_tidy_path, show_col_types = FALSE)
freq_model_glance <- read_csv(freq_glance_path, show_col_types = FALSE)
complexity_model_tidy <- read_csv(complexity_tidy_path, show_col_types = FALSE)
complexity_model_glance <- read_csv(complexity_glance_path, show_col_types = FALSE)
complexity_partial <- read_csv(complexity_partial_path, show_col_types = FALSE)
complexity_summary <- yaml::read_yaml(complexity_summary_path)
N <- as.integer(metrics$n_obs)

# helpers for formatting
fmt3 <- function(x) sprintf("%.3f", x)
```

```r
fmt6 <- function(x) sprintf("%.6f", x)
fmt1 <- function(x) sprintf("%.1f", x)
fmt0 <- function(x) sprintf("%.0f", x)
fmt_p <- function(p) {
  if (is.na(p)) return("= NA")
  if (p < .001) return("< .001")
  formatted <- sprintf("= %.3f", p)
  sub("= 0\\.", "= .", formatted)
}

freq_coef <- freq_model_tidy$Estimate[freq_model_tidy$term == "log_freq"]
strokes_coef <- freq_model_tidy$Estimate[freq_model_tidy$term == "strokes"]
freq_pct <- (exp(freq_coef) - 1) * 100
strokes_pct <- (exp(strokes_coef) - 1) * 100

complexity_estimate <- complexity_model_tidy$Estimate[1]
complexity_se <- complexity_model_tidy[["Std. Error"]][1]
complexity_t_value <- complexity_model_tidy[["t value"]][1]
complexity_p_value <- complexity_model_tidy[["Pr(>|t|)"]][1]
complexity_penalty_pct <- complexity_model_tidy$penalty_pct[1]
complexity_r2 <- complexity_model_glance$r_squared[1]
complexity_df_resid <- complexity_model_glance$df_residual[1]
complexity_n <- complexity_model_glance$n[1]
complexity_sigma <- complexity_model_glance$sigma[1]
complexity_baseline_strokes <- complexity_summary$reference$strokes
complexity_baseline_rt <- complexity_summary$reference$predicted_rt_ms
complexity_peak_strokes <- complexity_summary$max_penalty$strokes
complexity_peak_penalty <- complexity_summary$max_penalty$penalty_ms
complexity_peak_rt <- complexity_summary$max_penalty$predicted_rt_ms
complexity_slope <- complexity_summary$slope$rate_ms_per_stroke
complexity_slope_start <- complexity_summary$slope$start_strokes
complexity_slope_end <- complexity_summary$slope$end_strokes
```

Overview

This report reads pre-computed outputs from the simple demo pipeline.

- Processed data: `outputs/data/processed.csv`
- Cleaning summary: `outputs/results/cleaning.yml`
- Model metrics: `outputs/results/metrics.yml`

Cleaning Summary

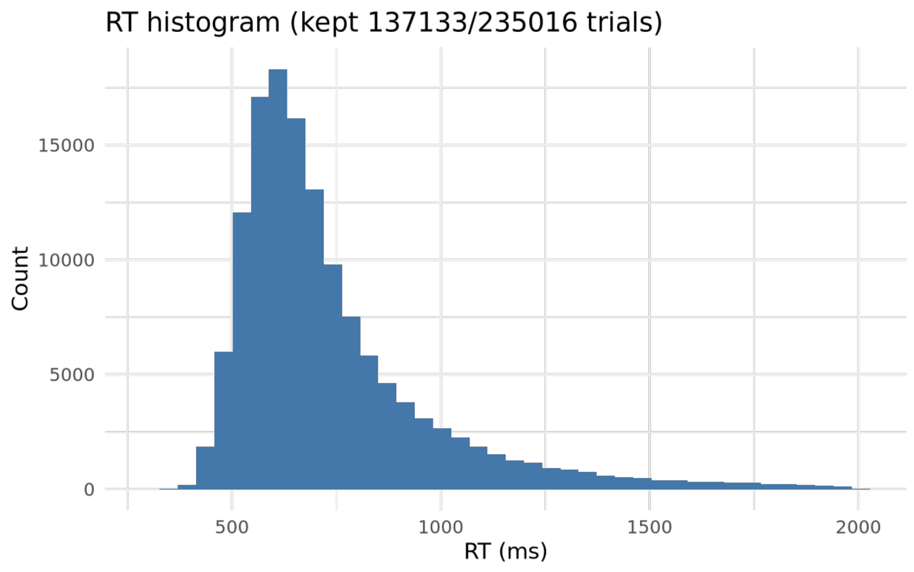
The pipeline kept 137133 of 235016 trials (dropped 97883). Settings: correct-only = TRUE, RT range = 200–2000 ms.

```
data.frame(
  setting = c("correct_only","rt_min_ms","rt_max_ms","total_trials","kept_trials","dropped_t
  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
```

RT Histogram (kept trials)

```
knitr::include_graphics(fig_path)
```



Model Metrics

Model: lm(mean_log_rt ~ log_freq + strokes) (N = 3852)

$R^2 = 0.434$.

```r
cat(paste0("Adjusted R² = ", fmt3(as.numeric(metrics$adj_r2)), ".\n\n"))
```

```
Adjusted R² = 0.433.
```

```r
cat(paste0("Residual sigma = ", fmt3(as.numeric(metrics$sigma)), ".\n\n"))
```

```
Residual sigma = 0.099.
```

```r
cat("Information criteria:\n\n")
```

```
Information criteria:
```

```r
print(data.frame(
  metric = c("AIC", "BIC"),
  value = c(fmt3(as.numeric(metrics$aic)), fmt3(as.numeric(metrics$bic)))
))
```

```
  metric     value
1    AIC -6851.160
2    BIC -6826.134
```

Coefficients:

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

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

Character Frequency Model

Character-level summaries (`outputs/data/character_frequency_model_data.csv`) were modelled with median lexical decision times as the outcome and predictors `log_freq` and `strokes`.

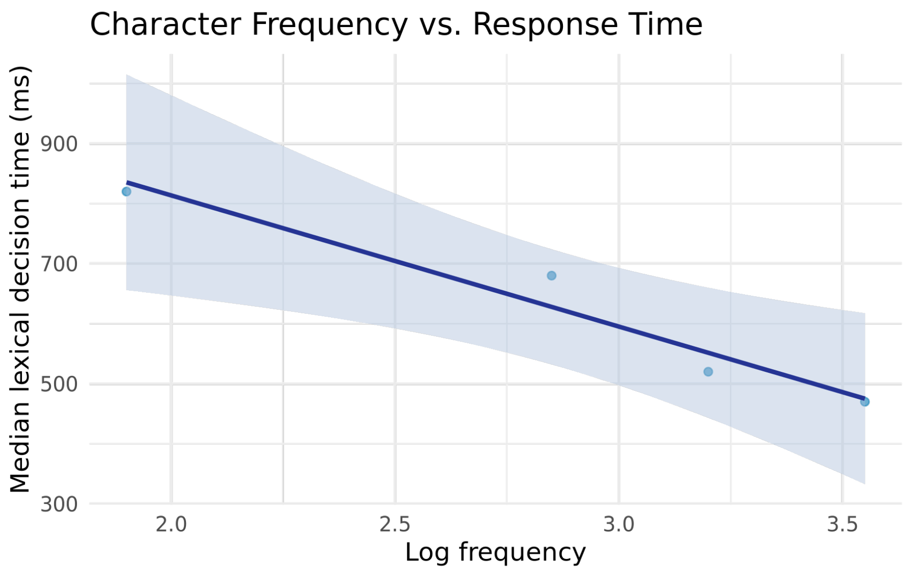
```r
knitr::kable(freq_model_tidy, digits = 3)
```

term	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	6.900	0.216	31.895	0.020
log_freq	-0.271	0.045	-6.063	0.104
strokes	0.032	0.012	2.612	0.233

95% uncertainty for the combined fit: $R^2 = 0.990$, sigma = 0.043.

- `log_freq`: -0.271 on the log scale → -23.737% faster median responses per one-unit increase in log frequency.
- `strokes`: 0.032 on the log scale → 3.263% slower median responses per additional stroke.

```r
knitr::include_graphics(freq_fig_path)
```



Trials aggregated to character level; loess smooth with 95% confidence band.

Median response times fall sharply from rare to moderately frequent characters, but the loess curve flattens once log frequency exceeds roughly 3, suggesting diminishing speed gains for the most common characters. The widening confidence band at high frequency reflects the sparse sample, so the apparent plateau should be revisited when more characters are available, yet the current evidence aligns with classic frequency saturation once visual complexity is held constant.

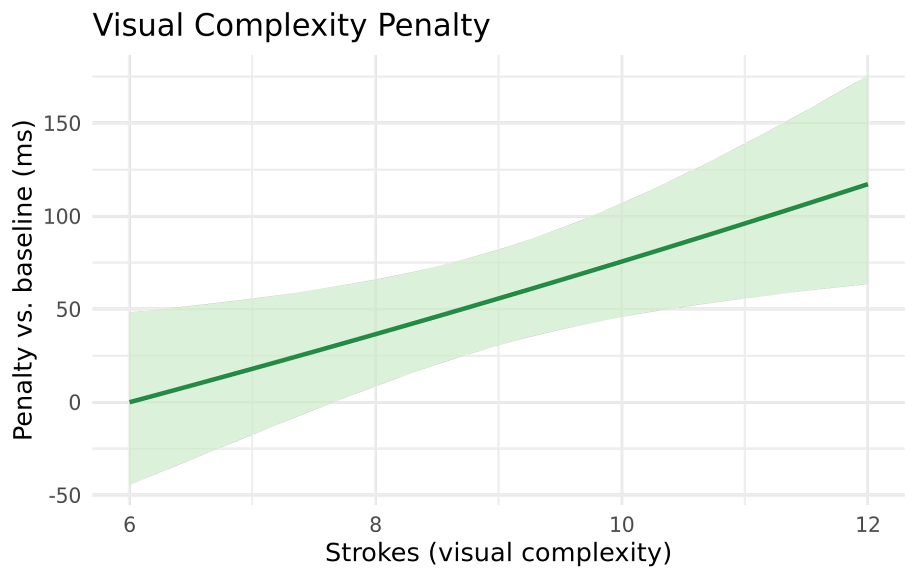
Visual Complexity Penalty

To isolate the complexity penalty, we fit a linear regression predicting log median response time from log frequency and strokes. The model explained (R^2 = 0.990) with $n = 4$, leaving 0.043 residual log-millisecond noise. The strokes coefficient indicated slower responses for more complex characters ((b = 0.032), (SE = 0.012), (t(1) = 2.612), (p = .233)), equivalent to a 3.263% increase in median response time per additional stroke when frequency is held fixed.

```r
knitr::kable(complexity_model_tidy, digits = 3)
```

term	Estimate	Std. Error	t value	Pr(>|t|)	penalty_pct
strokes	0.032	0.012	2.612	0.233	3.263

```r
knitr::include_graphics(complexity_fig_path)
```



ictions from linear model holding log frequency at its mean; shaded band shows 95% CI.

Holding log frequency at its mean, predicted response times rose from 552 ms at 6.0 strokes to 669 ms at 12.0 strokes, a penalty of 117 ms concentrated at the top end of the sampled complexity range (see Figure). The slope over the final stroke increment was approximately 21.4 ms per stroke, underscoring that the penalty sharpens for the most intricate characters. Because only 4 characters passed preprocessing, these estimates should be interpreted as preliminary benchmarks until broader coverage is available.