Metodología de la modelación conductual

Seminario Metodológico IV

Claudio Lavin

Profesor
Doctor en Neurociencias, PUC
claudio.lavin@mail.udp.cl

Roberto Garcia

Data Scientits
Lic. Psicologia, UDP
roberto.garciaa@mail.udp.cl

Escuela de Psicología, Doctorado en Neurociencias
Universidad Adolfo Ibáñez

Computational modeling

Computational modeling of behavior

Modelos matemáticos precisos para interpretar los datos del comportamiento. Los modelos computacionales instancian diferentes "hipótesis algorítmicas" sobre cómo se genera el comportamiento. (Wilson and Collins, 2019)

- Los datos del comportamiento suelen presentarse en forma de elecciones, pero también pueden ser tiempos de reacción, movimientos oculares u otros comportamientos fácilmente observables, e incluso datos neuronales.

Computational modeling

Two-Armed Bandit

Dos máquinas tragamonedas (bandits), cada una con una probabilidad de dar un premio (Reward). El objetivo es maximizar las ganancias sin conocer de antemano cuál es la probabilidad de ganar en cada una, mediante ensayo y error.

N-Armed Bandit es una tarea formalizada en Machine Learning (Reinforcement Learning aka RL) para problemas de toma de decisión en incertidumbre. Su adaptación como paradigma experimental en Neurociencias son tareas de aprendizaje probabilístico (Probabilistic Learning Task).

- States (S): Es el entorno en un momento dado. Describe una configuración del entorno. En términos experimentales, se puede entender como Contexto.
- Choices o Actions (A): Lo que el sujeto hace en un estado, una decisión que cambia (o no) el entorno.
- Rewards (R): La retroalimentación (outcome, feedback) que el agente recibe después de tomar una acción en un estado dado.

Computational modeling

Modelamiento Computacional

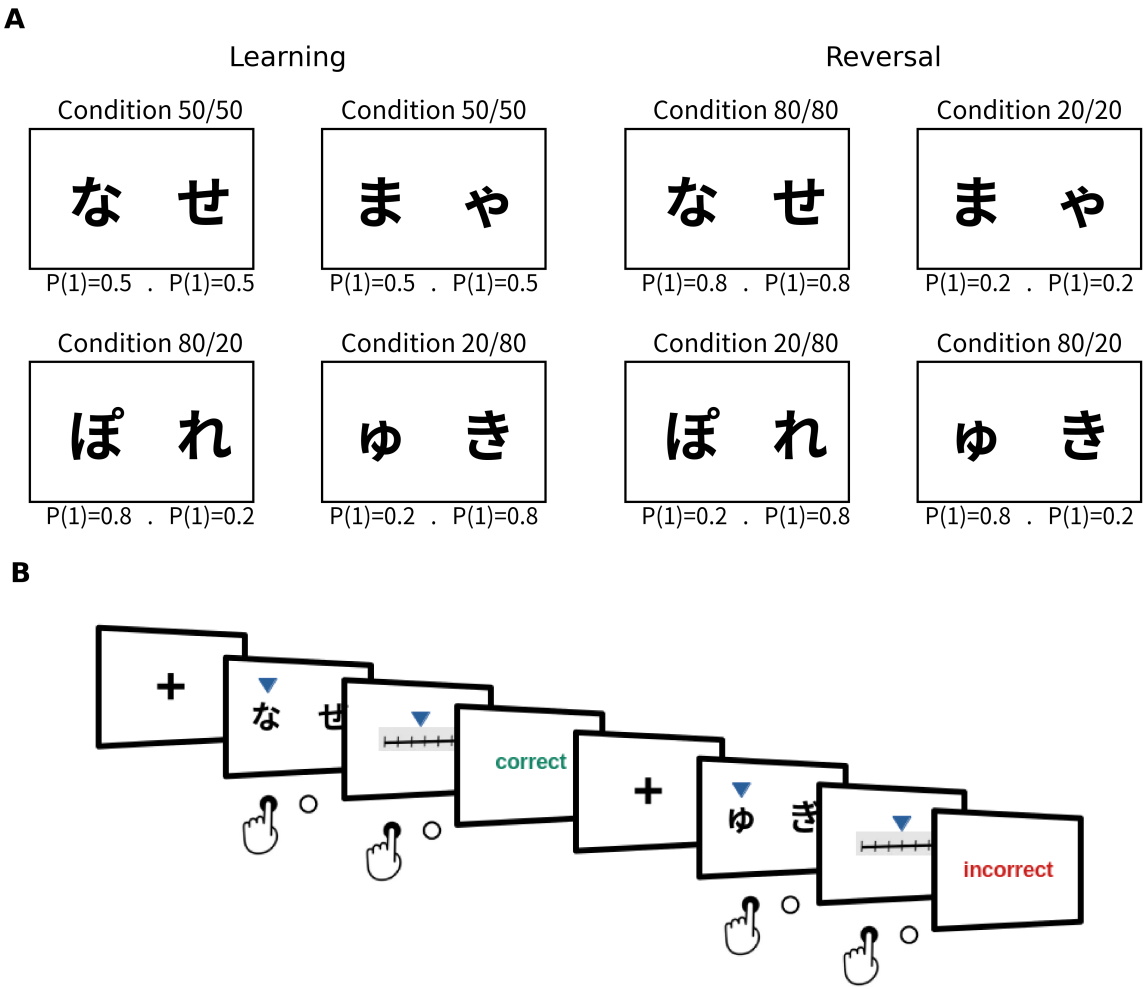
Two-Armed Bandit

Definimos una versión simple de la tarea con N Trials:

- States (S): Un solo estado o contexto estático, donde debe elegir entre 2 opciones o estímulos (LEFT or RIGHT options).
- Actions (A): Acción o elección que realiza en cada trial. Se pueden codificar como 0 o 1 para la acción de elegir LEFT o RIGHT respectivamente.
- Reward (R): Recompensa recibida. En este caso, la recompensa es probabilística:
  - Elegir RIGHT tiene probabilidad p=0.8 de entregar +1 de recompensa y 1-p=0.2 de no entregar recompensa.
  - Elegir LEFT tiene probabilidad p=0.2 de entregar +1 de recompensa y 1-p=0.8 de no entregar recompensa.

Computational modeling

Probabilistic Learning Task



(Lavín et al., 2024)

Computational modeling

Modelamiento Computacional

Procedimiento Tradicional en Modelo Lineal

Cuando ajustamos un modelo de **Regression Lineal** a un conjunto de datos, se realizan una serie de procedimientos que son transversales al modelamiento estadístico.

- Especificación e identificación del modelo.
- Estimación de parámetros y coeficientes.
- Evaluación de parámetros.
- Evaluación y/o comparación de modelos.

Nota: Cada etapa del procedimiento puede implementarse con diferentes métodos y enfoques estadísticos.

Computational modeling

Especificación del modelo

Modelo Lineal Simple

Al Especificar un modelo, definimos formalmente los parámetros, variables y relaciones entre ellas del modelo.

- En el LM, el modelo consiste en una función lineal con un término de error y una serie de supuestos (Gauss-Markov Theorem), formalmente:

$$\mathrm{Y}_i = \beta_0 + \beta_1 \mathrm{X}_i + \epsilon_i$$

Y_i: Valores observados de Y

β_0: Intercepto (Valor medio de Y cuando X=0)

β_1: Coeficiente de la variable X, refleja la influencia de X sobre Y controlando el resto de variables.

ϵ_i: Error, componente aleatorio, parte no explicada por el modelo.

Computational modeling

Especificación del modelo

Modelo Lineal Simple

- Reemplazando los coeficientes reales por los estimados ($\hat{\beta}_0$, $\hat{\beta}_1$), se puede predecir la variable dependiente de la muestra:

$$\widehat{\mathrm{Y}}_i = \hat{\beta}_0 + \hat{\beta}_1 \mathrm{X}_i$$

$$\hat{\epsilon_i} = \mathrm{Y}_i - \widehat{\mathrm{Y}}_i$$

$\widehat{\mathrm{Y}}_i$: Valores predichos por el modelo

$\hat{\beta}_0$, $\hat{\beta}_1$: coeficientes estimados

$\hat{\epsilon_i}$: Residuales: diferencia entre los valores reales y los predichos.

Computational modeling

Especificación del modelo

Modelo de elección binaria

- Los modelos toman la forma de ecuaciones matemáticas que vinculan las variables observables (ej, estímulos, recompensas, experiencias pasadas) con el comportamiento futuro inmediato.

  Por ejemplo, en un modelo de respuesta binaria, la probabilidad de que $\mathrm{Y} = 1$ se modela como:

$$P(\mathrm{Y} = 1|\mathrm{X}) = \frac{1}{1 + e^{-\beta \mathrm{X}}}$$

  Si estamos interesados en predecir la probabilidad de elegir un estímulo entre 2 opciones, la probabilidad de que Choice = Right se modela como:

$$P(\mathrm{Choice} = \mathrm{Right}|\mathrm{X}) = \frac{1}{1 + e^{-\beta \mathrm{X}}}$$

  Dónde X es una variable independiente y β es un parámetro libre que representa la influencia de X en la probabilidad de elección.

Computational modeling

Especificación del modelo

Modelo de elección binaria

- Supongamos que tenemos las valoraciones subjetivas de las opciones Left y Right trial-a-trial. Podemos resumirlas en un único valor que refleje la diferencia de las valoraciones ΔValue:

$$\Delta\text{Value} = \text{Value}\,(\text{Right}) - \text{Value}\,(\text{Left})$$

  Donde $\Delta\text{Value} > 0$ implica que:

$$\text{Value}\,(\text{Right}) > \text{Value}\,(\text{Left})$$

- Sin embargo, las valoraciones subjetivas son variables no observables, por lo tanto debemos estimarlas a partir de otros observables.

Computational modeling

Especificación del modelo

Modelo RL-Rescorla-Wagner

- El modelo de Rescorla-Wagner relaciona el valor esperado de dos opciones y el aprendizaje a través de los conceptos de predicción y actualización del valor.
- Esta teoría propone que actualizamos el valor de los estímulos en la medida que tenemos más información. Si la información sobre el valor es redundante, no hay aprendizaje. Es decir, para que exista aprendizaje, debe existir **sorpresa** o **error de predicción**.
- El error de predicción δ da cuenta de que tanto difiere lo que esperamos respecto de lo que recibimos. En términos de Valor y recompensa se define como:

$$\delta = \text{Reward} - \text{Value}\,(\text{Right})$$

Donde para cada trial, Value (Right) es el valor que esperamos recibir (Expected Value) para la opción **Right** y Reward es lo que recibimos. Esto se conoce como **delta rule**.

Computational modeling

Especificación del modelo

Modelo RL-Rescorla-Wagner

- La actualización del valor se realiza trial-to-trial para las opciones presentadas.
- Por ejemplo, si en el trial t elegimos el estímulo **Right**, actualizamos el valor para ese estímulo en el siguiente trial:

$$\text{Value}\,(\text{Right})_{t+1} = \text{Value}\,(\text{Right})_t + \alpha\delta$$

Donde δ es el **error de predicción** y α es un parámetro libre conocido como **learning rate** que refleja la capacidad de los sujetos para incorporar la información nueva y actualizar el valor.

- Por otro lado, la opción no elegida **Left**, no se actualiza, por lo que mantiene el valor en el siguiente trial:

$$\text{Value}\,(\text{Left})_{t+1} = \text{Value}\,(\text{Left})_t$$

Computational modeling

Especificación del modelo

Modelo RL-Rescorla-Wagner

En resumen, el algoritmo para estimar el valor de las opciones trial-to-trial seria:

- En el trial $t = 1$ el sujeto debe elegir entre 2 opciones **Left** y **Right**. Se asume que no existen valoraciones previas, por lo que se valoran igual, es decir $\text{Value}\,(\text{Left}) = \text{Value}\,(\text{Right}) = 0.5$
- Calculamos la probabilidad de elegir **Right** a partir de ΔValue con el modelo binario:

$$P(\text{Choice} = \text{Right}|\Delta\text{Value}) = \frac{1}{1 + e^{-\beta\Delta\text{Value}}}$$

- El sujeto elige la opción con mayor probabilidad.

Computational modeling

Especificación del modelo

Modelo RL-Rescorla-Wagner

- En el trial $t = 1$ el sujeto recibe recompensa por su elección. Con esto actualiza el valor con la **regla delta** para el siguiente trial $t = 2$. Suponiendo que su elección fue **Right**, y su recompensa fue $Reward = 1$, La actualización es:

$$\begin{aligned} \delta &= \text{Reward} - \text{Value}\,(\text{Right})_t \\ \text{Value}\,(\text{Right})_{t+1} &= \text{Value}\,(\text{Right})_t + \alpha\delta \\ \text{Value}\,(\text{Left})_{t+1} &= \text{Value}\,(\text{Left})_t \end{aligned}$$

- El trial $t = 1$ finaliza y pasamos al trial $t = 2$, en donde se repite el ciclo, hasta completar todos los trials.

Nota: Este algoritmo retorna variables latentes trial-to-trial que pueden ser de interés conductual y neurocientífico. Para cada trial, tendremos el **Valor esperado** de cada opción, el **Error de predicción** y la **probabilidad de elegir** de cada opción.

Computational modeling

Algorithm Modelo RL Rescorla-Wagner

Require: $T = 30$ (N trials), β (temperature), α (learning rate), $P_1 = 0.8$, $P_0 = 0.2$ (prob de recompensa para acción),

Ensure: V_0, V_1 (expectativas inicializadas en 0.5)

Inicializar $V_0 \leftarrow 0.5$, $V_1 \leftarrow 0.5$

for $t = 1$ **to** T **do**

Calcular probabilidad de elegir acción 1:

$$P(a = 1) = \frac{1}{1 + e^{-\beta(V_1 - V_0)}}$$

Seleccionar acción a:

if $a = 1$ **then**

Generar recompensa $R \sim \text{Bernoulli}(P_1)$

Actualizar $V_1 \leftarrow V_1 + \alpha \cdot (R - V_1)$

else

Generar recompensa $R \sim \text{Bernoulli}(P_0)$

Actualizar $V_0 \leftarrow V_0 + \alpha \cdot (R - V_0)$

end if

end for

Retornar V_0, V_1

Computational modeling

Variables computacionales

Choice Probability (Predicted probability)

- La probabilidad de la decisión es la función que vincula el Valor con la probabilidad binaria y corresponde a la predicción del modelo binario para las decisiones. Por lo que se espera que la diferencia entre la predicción del modelo, y las decisiones observadas (**error**) sea mínima.

δ (reward prediction error)

- δ refleja cuánto distan las **expectativas** de los **resultados obtenidos**.
- La intuición es que cuando **recibimos lo que esperamos, no existe aprendizaje**.

Computational modeling

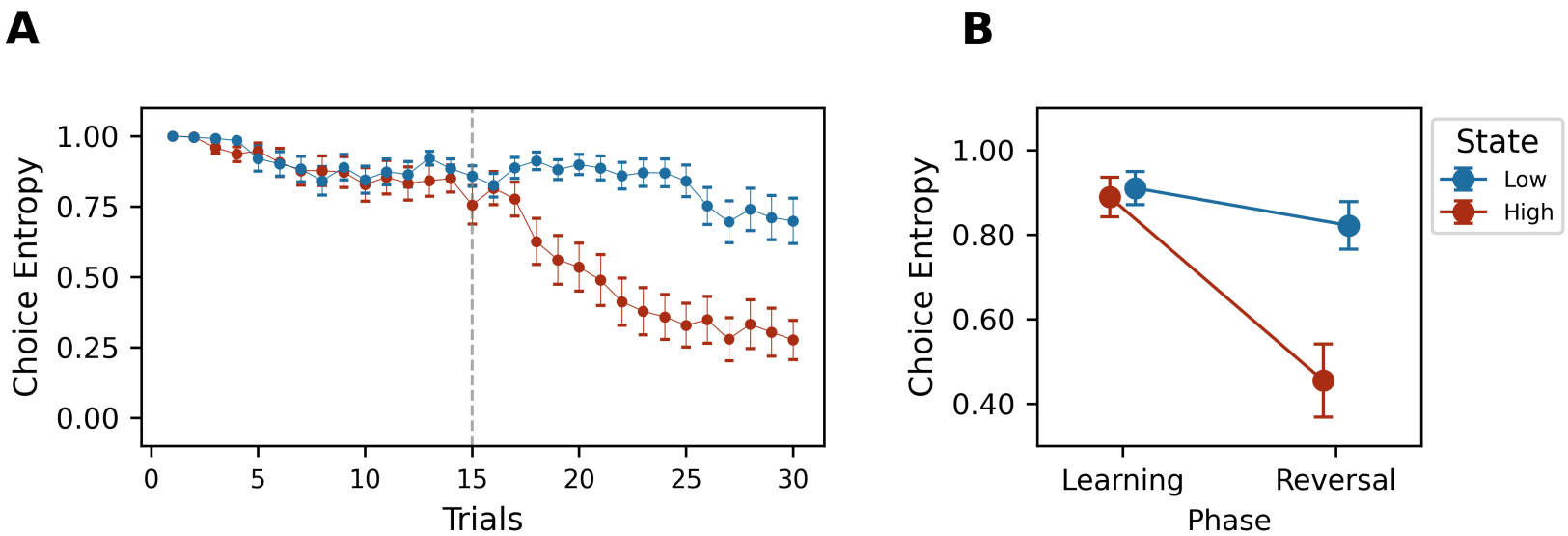
Variables computacionales

Valor (Expected Value)

- La variable computacional **Valor esperado** refleja la computación de valor subjetivo asociada a cada estímulo.
- Para el caso binario, la diferencia de las valoraciones de los estímulos (ΔValue) puede interpretarse como:
  - Resume en un valor, las **valoraciones de las opciones**. Esto es especialmente util si se quiere obtener la probabilidad de elegir a partir del valor.
  - $|\Delta\text{Value}|$ refleja la **dificultad** de discriminar el valor de dos opciones. Valores $|\Delta\text{Value}| \approxeq 0$ implica que las valoraciones son muy similares, por lo que **discriminar cuál tiene mayor valor**, es más difícil.

Computational modeling

Computational Variables



(Lavín et al., 2024)

Computational modeling

Parámetros computacionales

β (inverse temperature)

El parámetro β se conoce como **choice stochasticity** o **inverse temperature**, y controla la influencia del **Valor** en la toma de decisión.

- La intuición es que valores de $\beta \gg 1$ **aumentan** el efecto del **Valor**. Es decir la decisión se guía fuertemente por la opción con mayor **Valor**. Esto se conoce como **explotación** e implica una toma de decisión mas **determinista** o dependiente del valor.
- Por otro lado si $\beta \approxeq 0$ **disminuyen** el efecto del **Valor** de la opción. Es decir la decisión deja de guiarse por el **Valor**, permitiendo que otras opciones alternativas sean elegidas. Esto se conoce como **exploración aleatoria** e implica una toma de decisión menos **determinista** o independiente del valor.

Nota: El término de **inverse temperature** proviene de la física estadística: En sistemas termodinámicos, la **temperatura** aumenta la **entropía** del sistema.

Computational modeling

Parámetros computacionales

α (learning rate)

El parámetro α se conoce como **learning rate** y controla la influencia del **error de predicción** a la hora de actualizar el valor.

- La intuición es que los valores de $\alpha \approxeq 0$ **disminuyen** el efecto del **Error de predicción** en la **actualizacion del valor**, es decir, que los sujetos tienden a mantener sus valoraciones o son más resistentes a cambiarlas (**persistencia**).
- Por otro lado, si $\alpha \approxeq 1$, entonces el efecto del **Error de predicción** en la **actualización del valor aumenta**, por lo que el sujeto está más dispuesto a actualizar sus valoraciones incorporando la información nueva en cada trial (**adaptación**).

Computational modeling

Estimación de parámetros

- El método tradicional en LM es mínimos cuadrados ordinarios (ordinary least squares, OLS).
- El objetivo de este método es encontrar los parámetros que hacen que la función de error (Suma del Error Cuadrático, SSE) sea mínima. Por lo tanto el problema se reduce a encontrar el valor mínimo (Optimización)
- Bajo los supuestos de LM, la función de error es **cóncava**, por lo que tiene un único mínimo.
- Existen varios enfoques alternativos para la estimación de parámetros. Por conveniencia utilizaremos el método estimación por máxima verosimilitud (maximum likelihood estimation, MLE)

Nota: En el caso del modelo **Rescorla-Wagner**, no contamos con los supuestos, por lo que la función de error es desconocida, por lo que no es posible usar **OLS**.

Computational modeling

Estimación de parámetros

Maximum Likelihood Estimation, MLE

El objetivo de MLE es, dado un conjunto de datos observados $\mathrm{X} = \{x_1, x_2, \ldots, x_n\}$ y un modelo $f(x_i; \theta)$, encontrar los parámetros θ, que hacen que los datos observados sean los más probables:

$$\mathcal{L}(\theta|\mathrm{X}) = P(\mathrm{X}|\theta) = \prod_{i=1}^{n} f(x_i; \theta)$$

Maximizar la Verosimilitud es buscar el valor de $\hat{\theta}$ que maximiza la $\mathcal{L}(\theta|\mathrm{X})$ para nuestros datos:

$$\hat{\theta} = \arg\max_{\theta} \mathcal{L}(\theta|\mathcal{X})$$

La intuición es encontrar los parámetros que mejor explican los datos en términos probabilísticos).

Computational modeling

Minimize Negative LogLikelihood, NLL

Calcular Negative Log Likelihood se resume como:

- Trial-a-trial calcular el Likelihood, que equivale a calcular la probabilidad de elegir RIGHT
- Sumar todos los valores resultantes y multiplicar por -1 para obtener el negativo.

Encontrar el mínimo de los Negative Log Likelihood consiste:

- Calcular el Negative Log Likelihood con diferentes valores de θ
- Encontrar el valor de θ que tiene el valor mínimo de Negative Log Likelihood.

Computational modeling

```
def mle_rescorla_wagner(parms, states, rewards, choices):
    # Rescorla-Wagner Model (RW) (Rescorla & Wagner, 1972; Sutton & Barto, 1998)
    beta, alpha = parms[0], parms[1]
    n_trial = len(states)
    log_lik = [] #container to store the values
    V = np.zeros((2, n_trial+1), dtype=float)
    for t in range(n_trial):
        a_t, r_t = choices[t], rewards[t]
        dV = V[1, t] - V[0, t]
        cp_t = 1/(1+ np.exp(-beta*dV))
        log_lik_t = a_t * np.log(cp_t) + (1 - a_t) * np.log(1 - cp_t)
        log_lik.append(log_lik_t) #store current trial log-lik value
        pe = r_t - V[a_t, t]
        V[a_t, t+1] = V[a_t, t] + alpha*pe
        V[1-a_t, t+1] = V[1-a_t, t]
    neg_loglik = -np.sum(log_lik)  #Add all the values and calculate the negative
    return neg_loglik
```

Computational modeling

Minimize Negative LogLikelihood, NLL

La función *scipy.optimize.minimize* de Python implementa la optimización y retorna el valor de la función objetivo *mle_rescorla_wagner* y los valores de los parámetros estimados $\hat{\beta}$ y $\hat{\alpha}$

```
1  from scipy.optimize import minimize
2
3  beta = 0.1
4  alpha = 0.0
5  parms = [beta, alpha]
6  opt = minimize(mle_rescorla_wagner, parms, args=(states,
       rewards, choices), method='BFGS')
7  parms = opt.x
8  nll = opt.fun
9  bic = -2 * -nll + len(parms) * np.log(len(states))
10 aic = 2* len(parms) + 2* nll
```

Computational modeling

Estimación de parámetros

Minimize Negative LogLikelihood, NLL

- Para facilitar la computacion y el calculo, concluimos que Maximizar Likelihood es equivalente a Maximizar el Log Likelihood:

$$\ell(\theta|\mathrm{X}) = \log(\mathcal{L}(\theta|\mathrm{X})) = \sum_{i=1}^{n} \log(f(x_i;\theta))$$

$$\hat{\theta} = \arg\max_{\theta} \ell(\theta|\mathrm{X})$$

- Luego Maximizar el Log Likelihood es equivalente a Minimizar el Negative Log Likelihood (NLL):

$$-\ell(\theta|\mathrm{X}) = -\log(\mathcal{L}(\theta|\mathrm{X})) = -\sum_{i=1}^{n} \log(f(x_i;\theta))$$

$$\hat{\theta} = \arg\min_{\theta} -\ell(\theta|\mathrm{X})$$

Computational modeling

Estimación de parámetros

Maximum Likelihood Estimation, MLE

En el caso de un **modelo lineal simple**, el modelo es una funcion lineal

$$y_i = f(x_i; \beta_0, \beta_1, \sigma) = \beta_0 + \beta_1 x_i + \epsilon_i$$
$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i \quad \text{donde} \quad \epsilon_i \sim \mathcal{N}(0, \sigma^2)$$

donde los parámetros libres incluidos son β_0, β_1, σ (intercepto, pendiente y desviación estándar de los errores respectivamente).

La función Negative Log Likelihood a minimizar es:

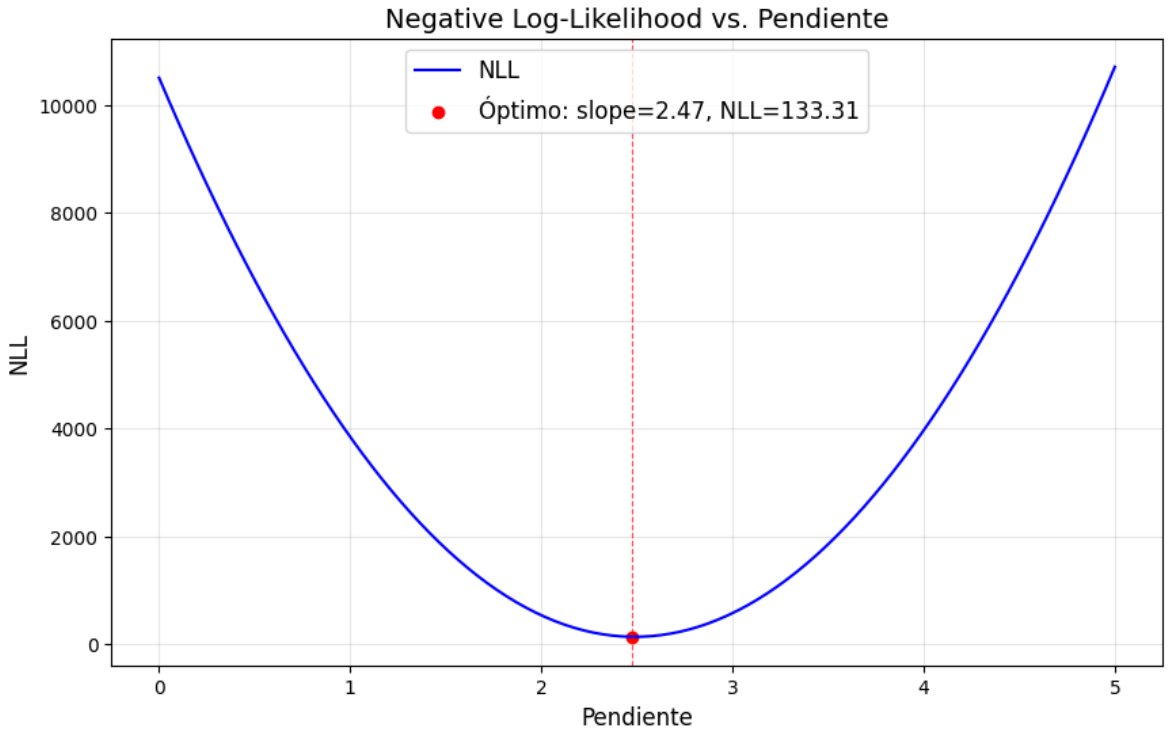
$$-\log(\mathcal{L}(\beta_0, \beta_1, \sigma|\mathrm{X})) = \frac{n}{2}\log(2\pi\sigma^2) + \frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - (\beta_0 + \beta_1 x_i))^2$$
$$\hat{\beta_0}, \hat{\beta_1}, \hat{\sigma} = \arg\min_{\beta_0, \beta_1, \sigma} -\mathcal{L}(\beta_0, \beta_1, \sigma|\mathcal{X})$$

Computational modeling

Estimación de parámetros

Maximum Likelihood Estimation, MLE

La optimización consiste en probar múltiples combinaciones de β_0, β_1, σ hasta encontrar el valor mínimo de $-\log(\mathcal{L}(\beta_0, \beta_1, \sigma|\mathrm{X}))$



Computational modeling

Estimación de parámetros

Maximum Likelihood Estimation, MLE

En el caso de un **modelo logístico simple**, el modelo es una función sigmoide:

$$P(y_i = 1 \mid x_i; \beta) = f(x_i; \beta) = \frac{1}{1 + e^{-(\beta x_i)}}$$

La función Negative Log Likelihood a minimizar es:

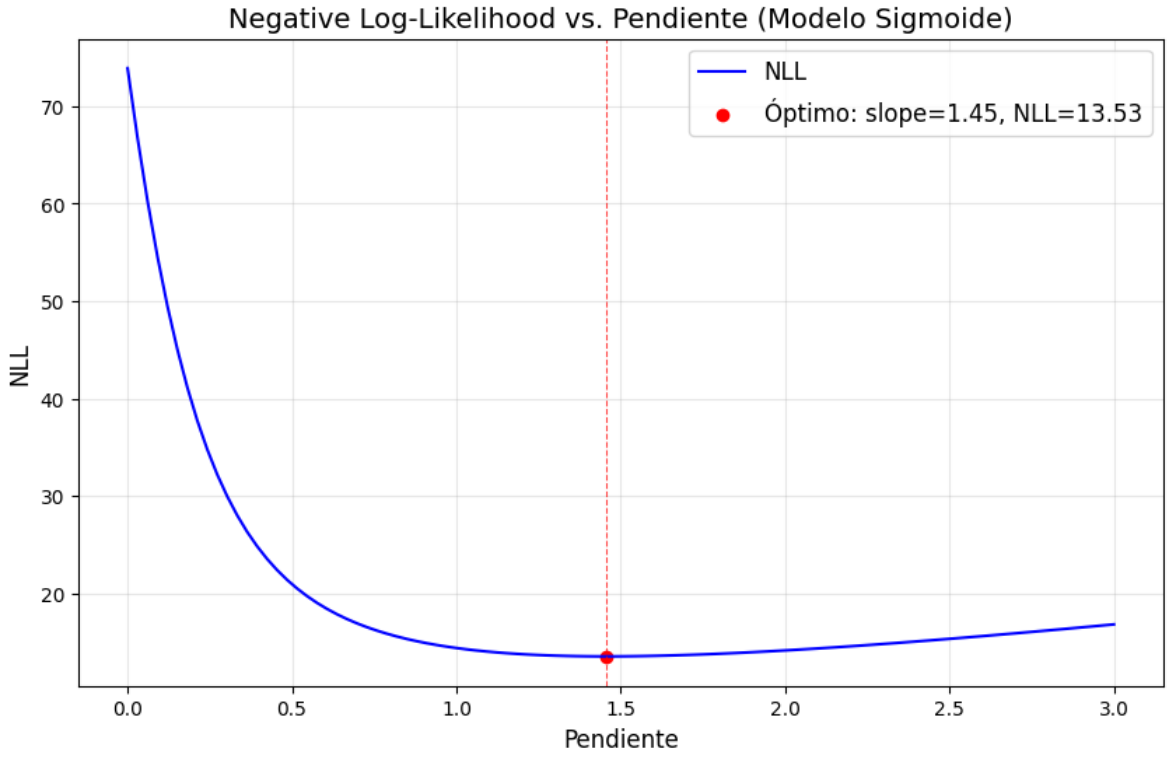
$$-\log(\mathcal{L}(\beta|\mathrm{X})) = -\sum_{i=1}^{n} \Big[y_i \log p(X_i, \beta) + (1 - y_i) \log(1 - p(X_i, \beta)) \Big]$$

Computational modeling

Estimación de parámetros

Maximum Likelihood Estimation, MLE

La optimización consiste en probar múltiples valores de β, hasta encontrar el valor mínimo de $-\log(\mathcal{L}(\beta|\mathrm{X}))$



Computational modeling

Estimación de parámetros

Maximum Likelihood Estimation, MLE

En el caso del modelo **Rescorla-Wagner**, $f(x_i; \beta, \alpha)$ es el modelo de probabilidad binaria dependiente del **Valor V(.)** :

$$\begin{aligned} f(\Delta \mathrm{V}_t; \beta, \alpha) =& p(\mathrm{choice}_t = \mathrm{right} \mid \Delta \mathrm{V}_t; \beta, \alpha) \\ =& \frac{1}{1 + e^{-(\beta \Delta \mathrm{V})}} \end{aligned}$$

$$\begin{aligned} \mathrm{V}\left(\mathrm{right}\right)_{t+1} &= \mathrm{V}\left(\mathrm{right}\right)_t + \alpha\delta \\ \delta_t &= \mathrm{reward}_t - \mathrm{V}\left(\mathrm{right}\right)_t \\ \Delta \mathrm{V}_t &= \mathrm{V}\left(\mathrm{right}\right)_t - \mathrm{V}\left(\mathrm{left}\right)_t \end{aligned}$$

La función Negative Log Likelihood a minimizar es:

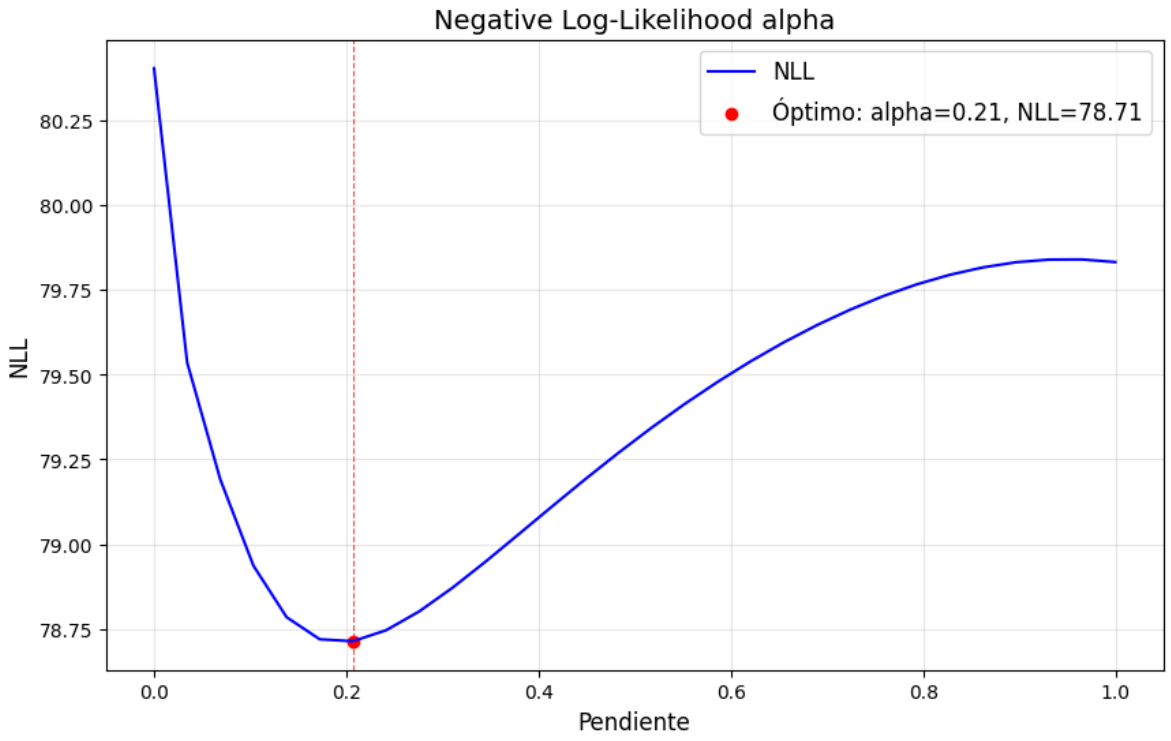
$$-\log(\mathcal{L}(\beta, \alpha | \Delta \mathrm{V_t})) = -\sum_{t=1}^{n} \left[\mathrm{choice}_t \log p(\Delta \mathrm{V}_t, \beta) + (1 - \mathrm{choice}_t) \log(1 - p(\Delta \mathrm{V}_t, \beta)) \right]$$

Computational modeling

Estimación de parámetros

Maximum Likelihood Estimation, MLE

La optimización consiste en probar múltiples combinaciones de β, α hasta encontrar el valor mínimo de $-\log(\mathcal{L}(\beta, \alpha|\Delta V_t))$



Computational modeling

Evaluación de parámetros

- Tradicionalmente, la evaluación de parámetros se resume un problema de toma de decisión estadística usando el test significancia de hipótesis nula (NHST) y depende de supuestos acerca de la distribución de $\hat{\beta}$ en muestras repetidas.
- NHST evalua si el parámetro es significativamente diferente de 0 donde $\hat{\beta}$ tiene distribución t-student o z dependiendo de la muestra. De esta forma:

$$t = \frac{\hat{\beta}_j - \beta_{j,0}}{\mathrm{SE}(\hat{\beta}_j)}$$

- En el caso del modelo **Rescorla-Wagner**, no contamos con los supuestos y desconocemos la distribución de los parámetros estimados, por lo que (NHST) no se puede implementar.
- Un método alternativo ampliamente usado en el modelamiento computacional es la **recuperación de parámetros**.

Computational modeling

Evaluación de parámetros

Parameter recovery

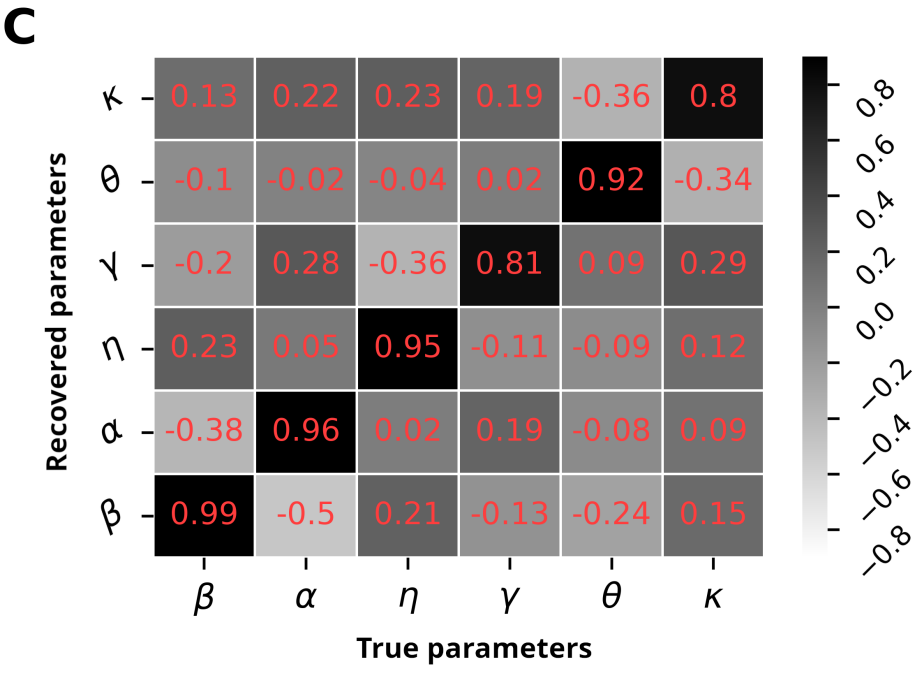
El procedimiento de **recuperación de parámetros** es el siguiente:

- Los **parámetros estimados** $(\hat{\beta}, \hat{\alpha})$ obtenidos al ajustar el modelo a datos reales por **MLE** para un sujeto serán los **parametros reales**. Por ejemplo, asumamos $\hat{\beta} = 5.1, \hat{\alpha} = 0.1$ como parámetros reales de un sujeto.
- **Simulamos** datos conductuales para la tarea, utilizando los **parámetros reales** y **Ajustamos el modelo** por medio de **MLE** para estimar los parámetros $\hat{\beta}$, $\hat{\alpha}$, pero esta vez sobre los **datos simulados**.
- Los parámetros estimados $(\tilde{\beta}, \tilde{\alpha})$ obtenidos al ajustar el modelo a datos simulados serán los **parámetros recuperados**.
- Se espera una alta correlación entre los parámetros estimados a partir de datos reales **true parameters** y los parámetros ajustados sobre datos simulados **recovered parameters**

Computational modeling

Evaluación de parámetros

Parameter Recovery Matrix



(Lavín et al., 2024)

Computational modeling

Evaluación de modelos (Goodness of fit)

Las métricas de Bondad de ajuste evalúan qué tan bien un modelo describe los datos observados.
En LM, la evaluación del modelo se realiza definiendo una métrica de bondad de ajuste (**Goodness of fit**) como **AIC**, **BIC** o $\mathbf{R}^2$, siendo esta ultima la mas utilizada.

- El coeficiente de determinación $\mathbf{R}^2$, evalúa la capacidad del modelo para explicar la variable dependiente (**Varianza explicada**
- Las métricas **AIC** y **BIC** se basan en teoría de la información y probabilidad bayesiana respectivamente, y son relativas a un modelo alternativo. :

$$\begin{aligned} \text{BIC} =& k \log(n) - 2 \log(\mathcal{L}) \\ \text{AIC} =& 2k - 2 \log(\mathcal{L}) \end{aligned}$$

donde n es el Tamaño de la muestra, k el Número de parámetros libres y $\log(\mathcal{L})$ es el Log-Likelihood.

Computational modeling

Evaluación de modelos (Goodness of fit)

- Las métricas de bondad de ajuste (como R^2 o Log-Likelihood) son sensibles al número de coeficientes (es decir, número de predictores o complejidad) en el modelo, es decir, un modelo con más predictores siempre tendrá mejor bondad de ajuste.
- Métricas de Bondad de ajuste como BIC, AIC o R^2_{adj} penalizan la complejidad del modelo.
- Existen procedimientos alternativos que provienen del Machine Learning para la validación de un modelo tales como la validación cruzada o la recuperación de modelos.

Computational modeling

Evaluación de modelos (Goodness of fit)

Modelo Lineal Simple

Comparación de modelos: Evalúa cuál de varios modelos candidatos explica mejor los datos observados.
Para el caso del **Modelo Lineal Simple**, podemos incorporar un nuevo predictor Z_i y comparar los modelos:

$$(1) \quad Y_i = \beta_0 + \beta_1 X_i + \epsilon_i,$$
$$(2) \quad Y_i = \beta_0 + \beta_1 X_i + \beta_2 Z_i + \epsilon_i$$

La comparación de los indicadores de Bondad de ajuste de los modelos arbitrará que modelo describe mejor los datos observados o si la incorporación de la nueva variable Z_i mejora el modelo.

Computational modeling

Evaluaciond de modelos (Goodness of fit)

Rescorla-Wagner

Para el caso del modelo **Rescorla-Wagner** la evaluación del modelo se implementa comparando las métricas de **goodness of fit** de un modelo objetivo con otros modelos candidatos (**Model comparison**).

- Definimos un modelo alternativo factible incluyendo una pequeña modificación al modelo **Rescorla-Wagner** previamente definido, el cual llamaremos **Asymmetric Rescorla-Wagner**
- Cada modelo debe representar una variante de la hipótesis computacional.

Computational modeling

Evaluación de modelos (Goodness of fit)

Asymmetric Rescorla-Wagner

Asymmetric Rescorla-Wagner es identico a **Rescorla-Wagner**, sin embargo, plantea la hipótesis de que los sujetos actualizan sus valoraciones de diferente manera dependiendo de si los resultados son **mejor o peor** de lo esperado.

Debido a que la actualización de las valoraciones está controlada por el **learning rate** (α), el modelo tendrá dos parámetros **learning rate**:

$$\alpha = \begin{cases} \alpha^{+} & \text{if } \delta > 0, \\ \alpha^{-} & \text{if } \delta \leq 0. \end{cases}$$

Donde δ es el **error de predicción** que define cuando los resultados son **mejor o peor** de lo esperado. Por lo tanto:

- α^{+} cuando los resultados son **mejor de lo esperado**
- α^{-} cuando los resultados son **peor de lo esperado**

Computational modeling

Evaluaciond de modelos (Goodness of fit)

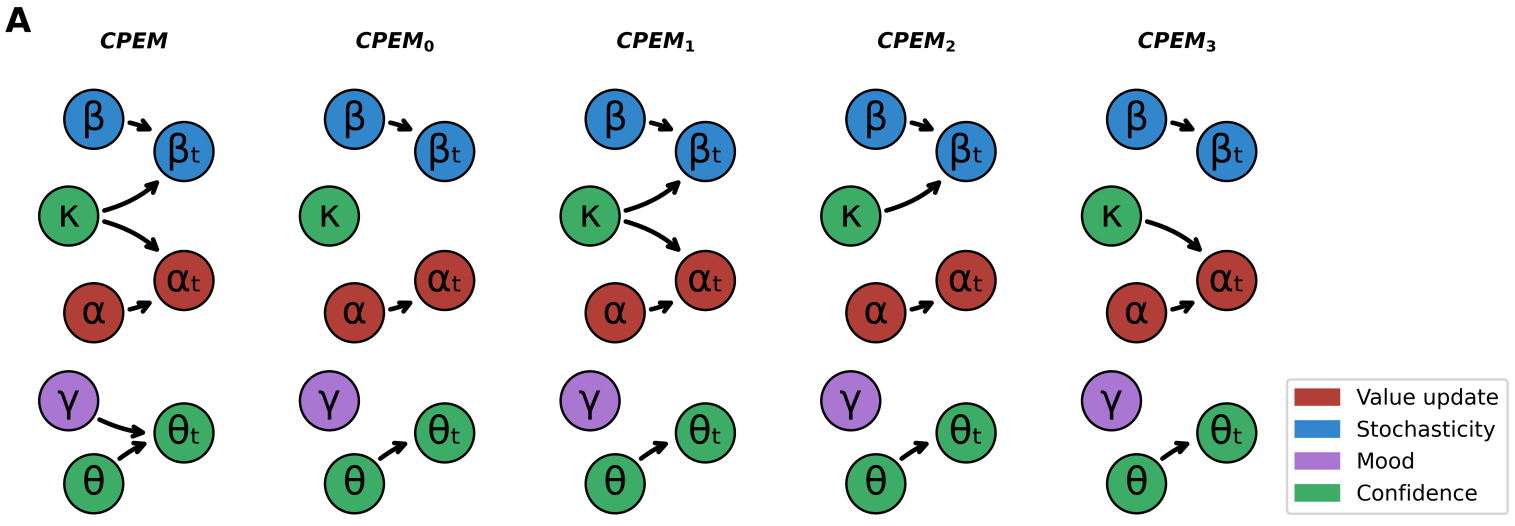
En resumen contamos con dos modelos candidatos que prueban hipótesis similares sobre el mecanismo de toma de decisión, pero difieren en un aspecto. (Lavín et al., 2024)

- **Modelo 1: Rescorla-Wagner** (β, α): Los sujetos aprenden por ensayo y error actualizando trial-to-trial el valor de las opciones en la medida de que las recompensas recibidas difieren de lo que esperábamos recibir.
- **Modelo 2: Asymmetric Rescorla-Wagner** (β, α^+, α^-): Los sujetos aprenden por ensayo y error actualizando tria-to-trial el valor de las opciones en la medida de que las recompensas recibidas difieren de lo que esperábamos recibir, sin embargo esta actualización es diferente si es **mejor de lo esperado,** α^+ o si es **peor de lo esperado,** α^-.

Nota: El modelo **Asymmetric Rescorla-Wagner** tiene un parámetro extra.

Computational modeling

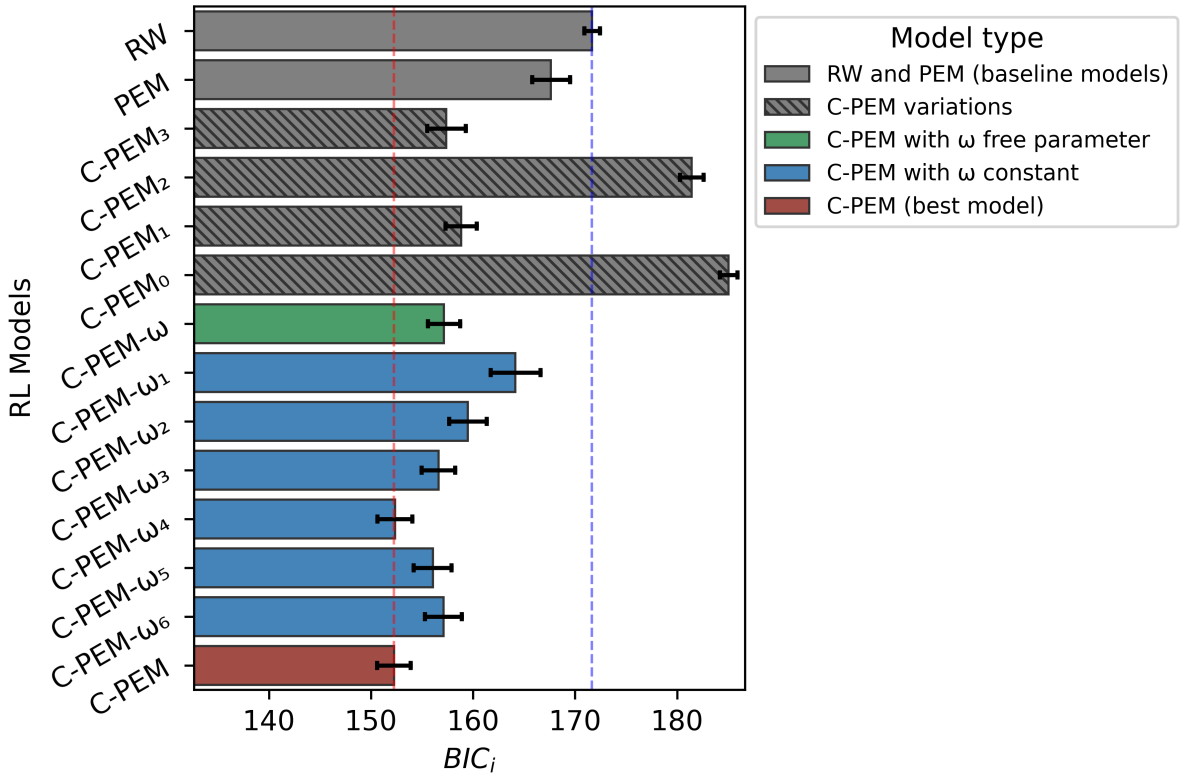
Evaluación de modelos (Goodness of fit)



(Lavín et al., 2024)

Computational modeling

Evaluación de modelos (Goodness of fit)



(Lavín et al., 2024)

Computational modeling

Fenotipado Computacional

Fenotipo computacional

Conjunto de parámetros, derivados de datos neurales y conductuales, que caracterizan los mecanismos cognitivos de un individuo.

- Caracterización mecanicista más explícita que formaliza los procesos cognitivos cuantitativamente.
- Reduce la dimensionalidad al comprimir el proceso objetivo en un conjunto de parámetros.
- Los parámetros estimados varían dentro y entre individuos, lo que permite examinar diferencias individuales.

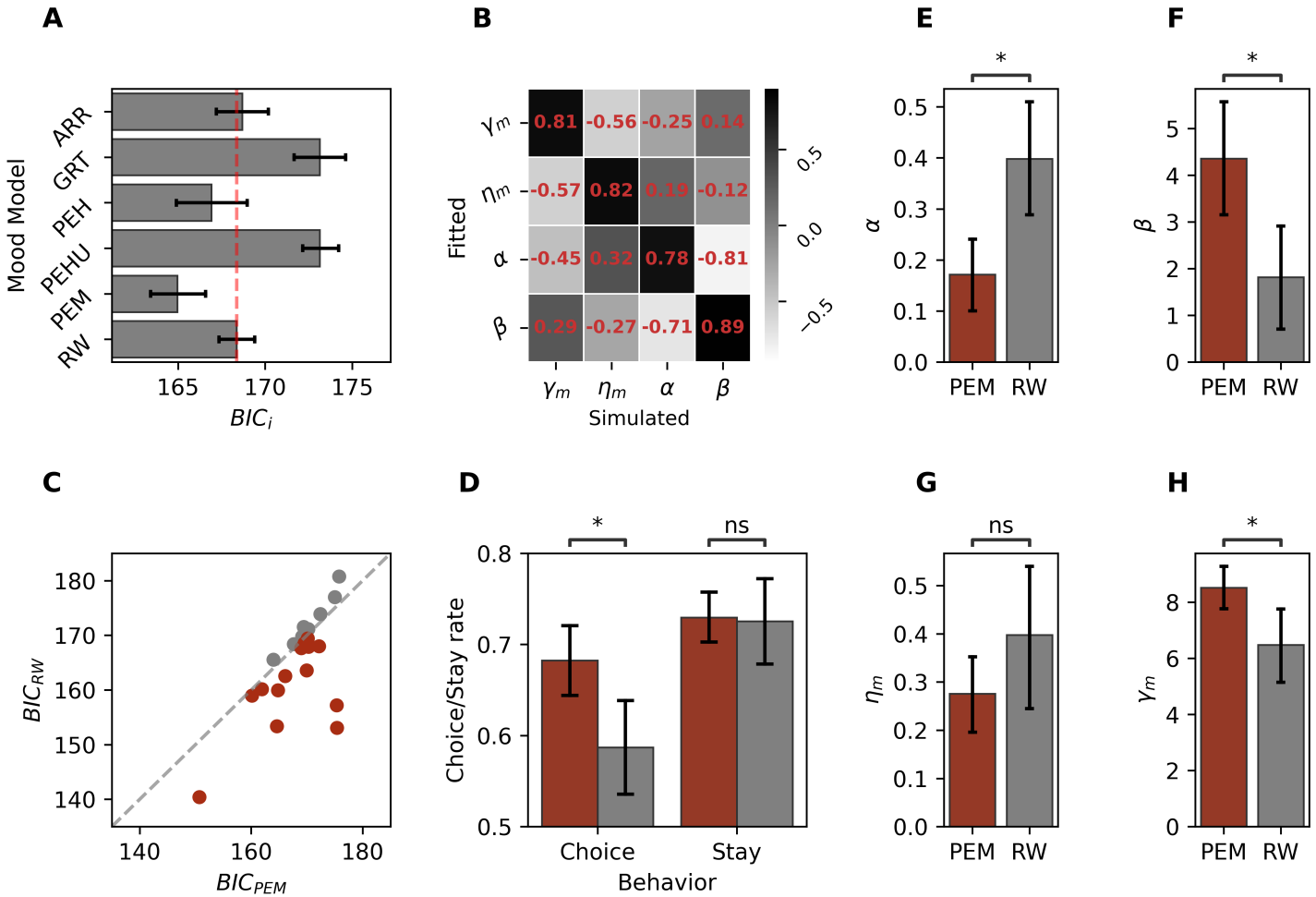
Computational modeling

Fenotipado Computacional

- Los enfoques tradicionales para el estudio de la personalidad, como el análisis factorial son eficaces para reducir el espacio de alta dimensión de la personalidad a constructos latentes.
- Los constructos tradicionales de personalidad son en gran medida agnósticos en cuanto a los mecanismos que conducen a diferencias en el comportamiento.
- Los procesos específicos se pueden operacionalizar, como por ejemplo por qué las personas con mayor conciencia buscan un comportamiento más orientado a objetivos.
- Identificar los fenotipos computacionales asociados con estos constructos de personalidad ofrece la oportunidad de vincular la validez predictiva del constructo con sus mecanismos subyacentes.

Computational modeling

Fenotipado Computacional



(Lavín et al., 2024)

Computational modeling

Sources

- Colab (Colab)
- Github (Github)
- Research (Research)

References

Lavín, C., García, R., and Fuentes, M. (2024). Navigating uncertainty: The role of mood and confidence in decision-making flexibility and performance. *Behavioral Sciences*, 14(12):1144.

Wilson, R. C. and Collins, A. G. (2019). Ten simple rules for the computational modeling of behavioral data. *Elife*, 8:e49547.