Four Data Science Curveballs

Kenneth W. Regan[1]
University at Buffalo (SUNY)

UP-STAT 2016

[1] Joint work with Tamal Tanu Biswas and with grateful acknowledgment to UB's Center for Computational Research (CCR)

Four Natural Expectations

1. Equal values yield equal behavior.

Four Natural Expectations

1. Equal values yield equal behavior.
2. Unbiased data-gathering yields unbiased data.

Four Natural Expectations

1. Equal values yield equal behavior.
2. Unbiased data-gathering yields unbiased data.
3. If Y is a continuous function of X, then a small change in X produces a small change in Y.

Four Natural Expectations

1. Equal values yield equal behavior.
2. Unbiased data-gathering yields unbiased data.
3. If Y is a continuous function of X, then a small change in X produces a small change in Y.
4. Factors whose insignificance you demonstrated will stay insignificant when you have 10x–100x data.

Four Natural Expectations

1. Equal values yield equal behavior.
2. Unbiased data-gathering yields unbiased data.
3. If Y is a continuous function of X, then a small change in X produces a small change in Y.
4. Factors whose insignificance you demonstrated will stay insignificant when you have 10x–100x data.
5. *OK, five:* Secondary aspects of standard library routines called by your data-gathering engines won't disturb the above expectations.

Four Natural Expectations

1. Equal values yield equal behavior.
2. Unbiased data-gathering yields unbiased data.
3. If Y is a continuous function of X, then a small change in X produces a small change in Y.
4. Factors whose insignificance you demonstrated will stay insignificant when you have 10x–100x data.
5. *OK, five:* Secondary aspects of standard library routines called by your data-gathering engines won't disturb the above expectations.

Key points: *Data points have histories*,

Four Natural Expectations

1. Equal values yield equal behavior.
2. Unbiased data-gathering yields unbiased data.
3. If Y is a continuous function of X, then a small change in X produces a small change in Y.
4. Factors whose insignificance you demonstrated will stay insignificant when you have 10x–100x data.
5. *OK, five:* Secondary aspects of standard library routines called by your data-gathering engines won't disturb the above expectations.

Key points: *Data points have histories*, *notionally* unbiased/ continuous/... need not imply *factually* unbiased/ continuous/...,

Four Natural Expectations

1. Equal values yield equal behavior.
2. Unbiased data-gathering yields unbiased data.
3. If Y is a continuous function of X, then a small change in X produces a small change in Y.
4. Factors whose insignificance you demonstrated will stay insignificant when you have 10x–100x data.
5. *OK, five:* Secondary aspects of standard library routines called by your data-gathering engines won't disturb the above expectations.

Key points: *Data points have histories*, *notionally* unbiased/ continuous/... need not imply *factually* unbiased/ continuous/..., and *zero-sigma* results can be artifacts too.

X and Y and Z

- $X = $ *values of chess moves* obtained by analyzing millions of chess positions with chess programs—called *engines*—with names like "Komodo" and "Stockfish" and "Rybka." Now vastly stronger than all human players even running on commodity hardware.

X and Y and Z

- X = *values of chess moves* obtained by analyzing millions of chess positions with chess programs—called *engines*—with names like "Komodo" and "Stockfish" and "Rybka." Now vastly stronger than all human players even running on commodity hardware.
- Y = *performance indicators* of (human) players:

X and Y and Z

- X = *values of chess moves* obtained by analyzing millions of chess positions with chess programs—called *engines*—with names like "Komodo" and "Stockfish" and "Rybka." Now vastly stronger than all human players even running on commodity hardware.
- Y = *performance indicators* of (human) players:
  - **MM%** = how often the player chose the move listed first by the engine in value order.
  - **EV%** = how often the player chose the first move or one of equal value, as happens in 8–10% of positions.
  - **ASD** = the average scaled difference in value between the player's chosen move m_i and the engine's first move m_1.

X and Y and Z

- X = *values of chess moves* obtained by analyzing millions of chess positions with chess programs—called *engines*—with names like "Komodo" and "Stockfish" and "Rybka." Now vastly stronger than all human players even running on commodity hardware.
- Y = *performance indicators* of (human) players:
  - **MM%** = how often the player chose the move listed first by the engine in value order.
  - **EV%** = how often the player chose the first move or one of equal value, as happens in 8–10% of positions.
  - **ASD** = the average scaled difference in value between the player's chosen move m_i and the engine's first move m_1.
- Z = *the players' chess Elo rating*: Adult beginner ≈ 600, club player 1400, master player 2200, human champs 2800, computers 3200+. Based on opponents' Elo ratings and results of the games.

A Predictive Analytic Model

1. Domain: A set T of decision-making situations t.
   Chess game turns
2. Inputs: Values v_i for every option at turn t.
   Computer values of moves m_i
3. Parameters: $s, c, \dots$ denoting skills and levels.
   Trained correspondence $P(s, c, \dots) \longleftrightarrow$ Elo rating E
4. Main Output: Probabilities p_i $(= p_{t,i})$ for $P(s, c, \dots)$ to select option i (at turn t).
5. The model's **Main Equation** entails $v_i = v_j \implies p_i = p_j$.
6. Derived Outputs:
   - **MM%, EV%, AE** and other aggregate statistics.
   - Projected confidence intervals for them—via Multinomial Bernoulli Trials plus an adjustment for correlation between consecutive turns.
   - **Intrinsic Performance Ratings** (IPRs) for the players.

Gathering Data With a GUI (note EV-tie at depths 12 and 13)

How the Model Operates

- Let v_1, v_i be values of the best move m_1 and ith-best move m_i.

How the Model Operates

- Let v_1, v_i be values of the best move m_1 and ith-best move m_i.
- Given $s, c, \dots$, the model computes $x_i = g_{s,c}(v_1, v_i) =$ the **perceived inferiority** of m_i by $P(s, c, \dots)$.

How the Model Operates

- Let v_1, v_i be values of the best move m_1 and ith-best move m_i.
- Given $s, c, \dots$, the model computes $x_i = g_{s,c}(v_1, v_i) =$ the **perceived inferiority** of m_i by $P(s, c, \dots)$.
- Besides g, the model picks a function $h(p_i)$ on probabilities.
- Could be $h(p) = p$ (bad), log (good enough?), $H(p_i)$, *logit*...

How the Model Operates

- Let v_1, v_i be values of the best move m_1 and ith-best move m_i.
- Given $s, c, \dots$, the model computes $x_i = g_{s,c}(v_1, v_i) =$ the **perceived inferiority** of m_i by $P(s, c, \dots)$.
- Besides g, the model picks a function $h(p_i)$ on probabilities.
- Could be $h(p) = p$ (bad), log (good enough?), $H(p_i)$, *logit*...
- The **Main Equation:**

$$\frac{h(p_i)}{h(p_1)} = 1 - x_i$$

How the Model Operates

- Let v_1, v_i be values of the best move m_1 and ith-best move m_i.
- Given $s, c, \dots$, the model computes $x_i = g_{s,c}(v_1, v_i) =$ the **perceived inferiority** of m_i by $P(s, c, \dots)$.
- Besides g, the model picks a function $h(p_i)$ on probabilities.
- Could be $h(p) = p$ (bad), log (good enough?), $H(p_i)$, *logit*...
- The **Main Equation:**

$$\frac{h(p_i)}{h(p_1)} = 1 - x_i = \exp\left(-\left(\frac{\delta(v_1, v_i)}{s}\right)^c\right),$$

- Here $\delta(v_1, v_i)$ scales $v_1 - v_i$ in regard to $|v_1|$.

Any equations in these values will entail

$$v_1 = v_2 \implies p_1 = p_2.$$

The Data: Old and New

- **Old:** Over 3 million moves of **Multi-PV** data: $>$ 250 GB.

The Data: Old and New

- **Old:** Over 3 million moves of **Multi-PV** data: $>$ 250 GB.
- Over 40 million moves of **Single-PV** data: $>$ 50 GB

The Data: Old and New

- **Old:** Over 3 million moves of **Multi-PV** data: $>$ 250 GB.
- Over 40 million moves of **Single-PV** data: $>$ 50 GB
- $=$ 150 million pages of text data at 2k/page.

The Data: Old and New

- **Old:** Over 3 million moves of **Multi-PV** data: > 250 GB.
- Over 40 million moves of **Single-PV** data: > 50 GB
- $=$ 150 million pages of text data at 2k/page.
- All taken on two quad-core home-style PC's plus a laptop using the GUI. This involved **retaining hashed move values** between game turns—which is the normal playing mode and only GUI option.

The Data: Old and New

- **Old:** Over 3 million moves of **Multi-PV** data: $>$ 250 GB.
- Over 40 million moves of **Single-PV** data: $>$ 50 GB
- $=$ 150 million pages of text data at 2k/page.
- All taken on two quad-core home-style PC's plus a laptop using the GUI. This involved **retaining hashed move values** between game turns—which is the normal playing mode and only GUI option.
- **New—using CCR:** Every published high-level game since 2014 in **Single-PV** mode.

The Data: Old and New

- **Old:** Over 3 million moves of **Multi-PV** data: $>$ 250 GB.
- Over 40 million moves of **Single-PV** data: $>$ 50 GB
- $=$ 150 million pages of text data at 2k/page.
- All taken on two quad-core home-style PC's plus a laptop using the GUI. This involved **retaining hashed move values** between game turns—which is the normal playing mode and only GUI option.
- **New—using CCR:** Every published high-level game since 2014 in **Single-PV** mode.
- **Master training sets** of 1.15 million moves by players of Elo ratings 1050, 1100, 1150, . . . (stepping by 50) . . . , 2700, 2750, 2800 in years 2010–2014, all in **Multi-PV mode**.

The Data: Old and New

- **Old:** Over 3 million moves of **Multi-PV** data: $>$ 250 GB.
- Over 40 million moves of **Single-PV** data: $>$ 50 GB
- $=$ 150 million pages of text data at 2k/page.
- All taken on two quad-core home-style PC's plus a laptop using the GUI. This involved **retaining hashed move values** between game turns—which is the normal playing mode and only GUI option.
- **New—using CCR:** Every published high-level game since 2014 in **Single-PV** mode.
- **Master training sets** of 1.15 million moves by players of Elo ratings 1050, 1100, 1150, . . . (stepping by 50) . . . , 2700, 2750, 2800 in years 2010–2014, all in **Multi-PV mode**.
- Taken with multiple Stockfish and Komodo versions using special batch scripts that **clear hash** between game turns.

An "ESP Test"

- In 8%–10% of positions, engine gives the top two moves the same value. Values are discrete up to 1 **centipawn**.

An "ESP Test"

- In 8%–10% of positions, engine gives the top two moves the same value. Values are discrete up to 1 **centipawn**.
- More often *some* pair of moves in the top 10 (say) will end up tied.

An "ESP Test"

- In 8%–10% of positions, engine gives the top two moves the same value. Values are discrete up to 1 **centipawn**.
- More often *some* pair of moves in the top 10 (say) will end up tied.
- Conditioned on one of the two moves having been played, let us invite humans to guess **which move is listed first by the program**.

An "ESP Test"

- In 8%–10% of positions, engine gives the top two moves the same value. Values are discrete up to 1 **centipawn**.
- More often *some* pair of moves in the top 10 (say) will end up tied.
- Conditioned on one of the two moves having been played, let us invite humans to guess **which move is listed first by the program**.
- The values are identical to the engine: it would not matter to the quality of the output which one the engine listed first. The values give no human reason to prefer one over the other.

An "ESP Test"

- In 8%–10% of positions, engine gives the top two moves the same value. Values are discrete up to 1 **centipawn**.
- More often *some* pair of moves in the top 10 (say) will end up tied.
- Conditioned on one of the two moves having been played, let us invite humans to guess **which move is listed first by the program**.
- The values are identical to the engine: it would not matter to the quality of the output which one the engine listed first. The values give no human reason to prefer one over the other.
- So this is a kind of ESP test.

An "ESP Test"

- In 8%–10% of positions, engine gives the top two moves the same value. Values are discrete up to 1 **centipawn**.
- More often *some* pair of moves in the top 10 (say) will end up tied.
- Conditioned on one of the two moves having been played, let us invite humans to guess **which move is listed first by the program**.
- The values are identical to the engine: it would not matter to the quality of the output which one the engine listed first. The values give no human reason to prefer one over the other.
- So this is a kind of ESP test. *How well do humans perform on it?*

An "ESP Test"

- In 8%–10% of positions, engine gives the top two moves the same value. Values are discrete up to 1 **centipawn**.
- More often *some* pair of moves in the top 10 (say) will end up tied.
- Conditioned on one of the two moves having been played, let us invite humans to guess **which move is listed first by the program**.
- The values are identical to the engine: it would not matter to the quality of the output which one the engine listed first. The values give no human reason to prefer one over the other.
- So this is a kind of ESP test. *How well do humans perform on it?*
- PEAR—Princeton Engineering Anomalies Research—notorious ESP project.

An "ESP Test"

- In 8%–10% of positions, engine gives the top two moves the same value. Values are discrete up to 1 **centipawn**.
- More often *some* pair of moves in the top 10 (say) will end up tied.
- Conditioned on one of the two moves having been played, let us invite humans to guess **which move is listed first by the program**.
- The values are identical to the engine: it would not matter to the quality of the output which one the engine listed first. The values give no human reason to prefer one over the other.
- So this is a kind of ESP test. *How well do humans perform on it?*
- PEAR—Princeton Engineering Anomalies Research—notorious ESP project.
- PEAR did 10,000s–100,000s of trials, trying to judge significance of deviations like 50.1% or even 50.01%.

An "ESP Test"

- In 8%–10% of positions, engine gives the top two moves the same value. Values are discrete up to 1 **centipawn**.
- More often *some* pair of moves in the top 10 (say) will end up tied.
- Conditioned on one of the two moves having been played, let us invite humans to guess **which move is listed first by the program**.
- The values are identical to the engine: it would not matter to the quality of the output which one the engine listed first. The values give no human reason to prefer one over the other.
- So this is a kind of ESP test. *How well do humans perform on it?*
- PEAR—Princeton Engineering Anomalies Research—notorious ESP project.
- PEAR did 10,000s–100,000s of trials, trying to judge significance of deviations like 50.1% or even 50.01%.
- How about *my* ESP test??

Sensitivity—Plotting Y against X

Conditioned on one of the top two moves being played, if their values (old: Rybka 3, depth 13; new: Stockfish and Komodo, depths 19+) differ by...:

1. **0.01**, the higher move is played 53–55% of the time.

Sensitivity—Plotting Y against X

Conditioned on one of the top two moves being played, if their values (old: Rybka 3, depth 13; new: Stockfish and Komodo, depths 19+) differ by...:

1. **0.01**, the higher move is played 53–55% of the time.
2. **0.02**, the higher move is played 58–59% of the time.

Sensitivity—Plotting Y against X

Conditioned on one of the top two moves being played, if their values (old: Rybka 3, depth 13; new: Stockfish and Komodo, depths 19+) differ by...:

1. **0.01**, the higher move is played 53–55% of the time.
2. **0.02**, the higher move is played 58–59% of the time.
3. **0.03**, the higher move is played 60–61% of the time.

Sensitivity—Plotting Y against X

Conditioned on one of the top two moves being played, if their values (old: Rybka 3, depth 13; new: Stockfish and Komodo, depths 19+) differ by...:

1. **0.01**, the higher move is played 53–55% of the time.
2. **0.02**, the higher move is played 58–59% of the time.
3. **0.03**, the higher move is played 60–61% of the time.
4. **0.00**, the higher move is played

Sensitivity—Plotting Y against X

Conditioned on one of the top two moves being played, if their values (old: Rybka 3, depth 13; new: Stockfish and Komodo, depths 19+) differ by...:

1. **0.01**, the higher move is played 53–55% of the time.
2. **0.02**, the higher move is played 58–59% of the time.
3. **0.03**, the higher move is played 60–61% of the time.
4. **0.00**, the higher move is played 57-59% of the time.

Sensitivity—Plotting Y against X

Conditioned on one of the top two moves being played, if their values (old: Rybka 3, depth 13; new: Stockfish and Komodo, depths 19+) differ by...:

1. **0.01**, the higher move is played 53–55% of the time.
2. **0.02**, the higher move is played 58–59% of the time.
3. **0.03**, the higher move is played 60–61% of the time.
4. **0.00**, the higher move is played 57-59% of the time.

- Last is not a typo—see post "**When is a Law Natural?**"

Sensitivity—Plotting Y against X

Conditioned on one of the top two moves being played, if their values (old: Rybka 3, depth 13; new: Stockfish and Komodo, depths 19+) differ by...:

1. **0.01**, the higher move is played 53–55% of the time.
2. **0.02**, the higher move is played 58–59% of the time.
3. **0.03**, the higher move is played 60–61% of the time.
4. **0.00**, the higher move is played 57-59% of the time.

- Last is not a typo—see post "**When is a Law Natural?**"
- Similar 58%-42% split seen for any pair of tied moves. What can explain it?

Sensitivity—Plotting Y against X

Conditioned on one of the top two moves being played, if their values (old: Rybka 3, depth 13; new: Stockfish and Komodo, depths 19+) differ by...:

1. **0.01**, the higher move is played 53–55% of the time.
2. **0.02**, the higher move is played 58–59% of the time.
3. **0.03**, the higher move is played 60–61% of the time.
4. **0.00**, the higher move is played 57-59% of the time.

- Last is not a typo—see post "**When is a Law Natural?**"
- Similar 58%-42% split seen for any pair of tied moves. What can explain it?
- Relation to slime molds and other "semi-Brownian" systems?

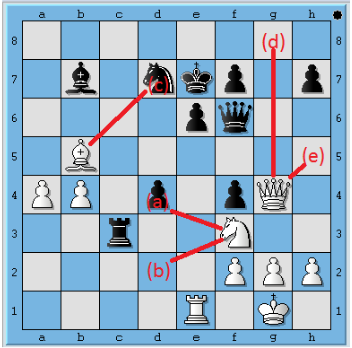
History and “Swing” over Increasing Depths

The ____ of drug-resistant strains of bacteria and viruses has ____ researchers’ hopes that permanent victories against many diseases have been achieved.

- (a) vigor . . corroborated
- (b) feebleness . . dashed
- (c) proliferation . . blighted
- (d) destruction . . disputed
- (e) disappearance . . frustrated

(source: itunes.apple.com)

=



Move	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Nd2	103	093	087	093	027	028	000	000	056	-007	039	028	037	020	014	017	000	006	000
Bxd7	048	034	-033	-033	-013	-042	-039	-050	-025	-010	001	000	-009	-027	-018	000	000	000	000
Qg8	114	114	-037	-037	-014	-014	-022	-068	-008	-056	-042	-004	-032	000	-014	-025	-045	-045	-050
. . .			. . .			. . .			. . .			. . .			. . .			. . .	
Nxd4	-056	-056	-113	-071	-071	-145	-020	-006	077	052	066	040	050	051	-181	-181	-181	-213	-213

Measuring "Swing" and Complexity and Difficulty

- Non-Parapsychological Explanation:

Measuring "Swing" and Complexity and Difficulty

- Non-Parapsychological Explanation: *Stable* Library Sorting.

Measuring "Swing" and Complexity and Difficulty

- Non-Parapsychological Explanation: *Stable* Library Sorting.
- Chess engines sort moves from last depth to schedule next round of search.

Measuring "Swing" and Complexity and Difficulty

- Non-Parapsychological Explanation: *Stable* Library Sorting.
- Chess engines sort moves from last depth to schedule next round of search.
- Stable $\rightarrow$ lower move jumps to 1st only with *strictly higher* value.

Measuring "Swing" and Complexity and Difficulty

- Non-Parapsychological Explanation: *Stable* Library Sorting.
- Chess engines sort moves from last depth to schedule next round of search.
- Stable $\rightarrow$ lower move jumps to 1st only with *strictly higher* value.
- Lead moves tend to have been higher at lower depths. Lower move "swings up."

Measuring "Swing" and Complexity and Difficulty

- Non-Parapsychological Explanation: *Stable* Library Sorting.
- Chess engines sort moves from last depth to schedule next round of search.
- Stable $\rightarrow$ lower move jumps to 1st only with *strictly higher* value.
- Lead moves tend to have been higher at lower depths. Lower move "swings up."
- Formulate numerical measure of swing "up" and "down" (a trap).

Measuring "Swing" and Complexity and Difficulty

- Non-Parapsychological Explanation: *Stable* Library Sorting.
- Chess engines sort moves from last depth to schedule next round of search.
- Stable $\rightarrow$ lower move jumps to 1st only with *strictly higher* value.
- Lead moves tend to have been higher at lower depths. Lower move "swings up."
- Formulate numerical measure of swing "up" and "down" (a trap).
- When best move swings up **4.0–5.0** versus **0.0–1.0**, players rated 2700+ find it only **30%** versus **70%**.

Measuring "Swing" and Complexity and Difficulty

- Non-Parapsychological Explanation: *Stable* Library Sorting.
- Chess engines sort moves from last depth to schedule next round of search.
- Stable $\rightarrow$ lower move jumps to 1st only with *strictly higher* value.
- Lead moves tend to have been higher at lower depths. Lower move "swings up."
- Formulate numerical measure of swing "up" and "down" (a trap).
- When best move swings up **4.0–5.0** versus **0.0–1.0**, players rated 2700+ find it only **30%** versus **70%**.
- **Huge differences** $\Longrightarrow$ corrections to the **main equation**.

Measuring “Swing” and Complexity and Difficulty

- Non-Parapsychological Explanation: *Stable* Library Sorting.
- Chess engines sort moves from last depth to schedule next round of search.
- Stable $\rightarrow$ lower move jumps to 1st only with *strictly higher* value.
- Lead moves tend to have been higher at lower depths. Lower move “swings up.”
- Formulate numerical measure of swing “up” and “down” (a trap).
- When best move swings up **4.0–5.0** versus **0.0–1.0**, players rated 2700+ find it only **30%** versus **70%**.
- **Huge differences** $\Longrightarrow$ corrections to the **main equation**.
- Will also separate *performance* and *prediction* in the model.

Second Curveball—pitched by same arm. . .

- **Single-PV** = normal playing (and cheating?) mode.

Second Curveball—pitched by same arm. . .

- **Single-PV** = normal playing (and cheating?) mode.
- **Multi-PV** values needed for main model equation.

Second Curveball—pitched by same arm. . .

- **Single-PV** = normal playing (and cheating?) mode.
- **Multi-PV** values needed for main model equation.
- Does difference matter for **MM%, EV%, ASD**?

Second Curveball—pitched by same arm. . .

- **Single-PV** = normal playing (and cheating?) mode.
- **Multi-PV** values needed for main model equation.
- Does difference matter for **MM%, EV%, ASD**?
- *Value* of first move seems unaffected. However (plotting Y vs. Z):

Second Curveball—pitched by same arm. . .

- **Single-PV** = normal playing (and cheating?) mode.
- **Multi-PV** values needed for main model equation.
- Does difference matter for **MM%, EV%, ASD**?
- *Value* of first move seems unaffected. However (plotting Y vs. Z):

Human players of all rating levels have 2–3% higher MM% and EV% to the Single-PV mode.

Second Curveball—pitched by same arm. . .

- **Single-PV** = normal playing (and cheating?) mode.
- **Multi-PV** values needed for main model equation.
- Does difference matter for **MM%, EV%, ASD**?
- *Value* of first move seems unaffected. However (plotting Y vs. Z):

Human players of all rating levels have 2–3% higher MM% and EV% to the Single-PV mode.

Thus my model is a biased predictor of MM% in Single-PV mode. Bias avoided by conducting test entirely in Multi-PV mode (arguably conservative). Why might this happen?

Second Curveball—pitched by same arm. . .

- **Single-PV** = normal playing (and cheating?) mode.
- **Multi-PV** values needed for main model equation.
- Does difference matter for **MM%, EV%, ASD**?
- *Value* of first move seems unaffected. However (plotting Y vs. Z):

> Human players of all rating levels have 2–3% higher MM% and EV% to the Single-PV mode.

Thus my model is a biased predictor of MM% in Single-PV mode. Bias avoided by conducting test entirely in Multi-PV mode (arguably conservative). Why might this happen?

> Single-PV mode maximally retards “late-blooming” moves from jumping ahead in the stable sort.

Third Curveball: A “Firewall at Zero

Surely Y = the frequency of large errors (“blunders”) ought to be continuous as a function of X = the value of the position.

Third Curveball: A “Firewall at Zero

Surely Y = the frequency of large errors (“blunders”) ought to be continuous as a function of X = the value of the position. But:

Third Curveball: A “Firewall at Zero

Surely Y = the frequency of large errors (“blunders”) ought to be continuous as a function of X = the value of the position. But:

Elo 2600–2850	Komodo 9.3				Stockfish 7 (modified)			
Value range	#pos	d10	d15	d20	#pos	d10	d15	d20
-0.30 to -0.21	4,710	9	13	18	4,193	13	10	14
-0.20 to -0.11	5,048	11	10	13	5,177	6	9	11
-0.20 to -0.01	4,677	11	13	16	5,552	8	9	16
0.00 exactly	9,168	24	25	28	9,643	43	40	38
+0.01 to +0.10	4,283	6	1	2	5,705	8	3	2
+0.11 to +0.20	5,198	7	5	3	5,495	10	5	3
+0.21 to +0.30	5,200	7	2	1	4,506	3	4	2

Reason evidently that 0.00 is a big *basin of attraction* in complex positions that may force one side to give perpetual check or force repetitions to avoid losing.

Third Curveball: A “Firewall at Zero

Surely Y = the frequency of large errors (“blunders”) ought to be continuous as a function of X = the value of the position. But:

Elo 2600–2850	Komodo 9.3				Stockfish 7 (modified)			
Value range	#pos	d10	d15	d20	#pos	d10	d15	d20
-0.30 to -0.21	4,710	9	13	18	4,193	13	10	14
-0.20 to -0.11	5,048	11	10	13	5,177	6	9	11
-0.20 to -0.01	4,677	11	13	16	5,552	8	9	16
0.00 exactly	9,168	24	25	28	9,643	43	40	38
+0.01 to +0.10	4,283	6	1	2	5,705	8	3	2
+0.11 to +0.20	5,198	7	5	3	5,495	10	5	3
+0.21 to +0.30	5,200	7	2	1	4,506	3	4	2

Reason evidently that 0.00 is a big *basin of attraction* in complex positions that may force one side to give perpetual check or force repetitions to avoid losing. Safety net provided $v_1 > 0$ but absent when $v_1 < 0$.

Third Curveball: A “Firewall at Zero

Surely Y = the frequency of large errors (“blunders”) ought to be continuous as a function of X = the value of the position. But:

Elo 2600–2850	Komodo 9.3				Stockfish 7 (modified)			
Value range	#pos	d10	d15	d20	#pos	d10	d15	d20
-0.30 to -0.21	4,710	9	13	18	4,193	13	10	14
-0.20 to -0.11	5,048	11	10	13	5,177	6	9	11
-0.20 to -0.01	4,677	11	13	16	5,552	8	9	16
0.00 exactly	9,168	24	25	28	9,643	43	40	38
+0.01 to +0.10	4,283	6	1	2	5,705	8	3	2
+0.11 to +0.20	5,198	7	5	3	5,495	10	5	3
+0.21 to +0.30	5,200	7	2	1	4,506	3	4	2

Reason evidently that 0.00 is a big *basin of attraction* in complex positions that may force one side to give perpetual check or force repetitions to avoid losing. Safety net provided $v_1 > 0$ but absent when $v_1 < 0$. Failure to charge adequately for large “notional errors.”

Fourth Curveball—Clearing Hash Does Matter

- Retaining hash apparently also retards "later-blooming" moves.

Fourth Curveball—Clearing Hash Does Matter

- Retaining hash apparently also retards "later-blooming" moves.
- Effect only 0.25–0.35%, not 2–3%, but significant now.

Fourth Curveball—Clearing Hash Does Matter

- Retaining hash apparently also retards "later-blooming" moves.
- Effect only 0.25–0.35%, not 2–3%, but significant now.
- Clearing is better for scientific reproducibility but further from actual playing conditions.

Fourth Curveball—Clearing Hash Does Matter

- Retaining hash apparently also retards "later-blooming" moves.
- Effect only 0.25–0.35%, not 2–3%, but significant now.
- Clearing is better for scientific reproducibility but further from actual playing conditions.

Thus my original "simple and self-evident" model needs substantial adjustment for all of these factors—to say nothing of factors I caught at the beginning. . .

Fourth Curveball—Clearing Hash Does Matter

- Retaining hash apparently also retards "later-blooming" moves.
- Effect only 0.25–0.35%, not 2–3%, but significant now.
- Clearing is better for scientific reproducibility but further from actual playing conditions.

Thus my original "simple and self-evident" model needs substantial adjustment for all of these factors—to say nothing of factors I caught at the beginning...

To conclude on a philosophic note:

Fourth Curveball—Clearing Hash Does Matter

- Retaining hash apparently also retards "later-blooming" moves.
- Effect only 0.25–0.35%, not 2–3%, but significant now.
- Clearing is better for scientific reproducibility but further from actual playing conditions.

Thus my original "simple and self-evident" model needs substantial adjustment for all of these factors—to say nothing of factors I caught at the beginning...

To conclude on a philosophic note: "Big Data" is critiqued for abandoning *theory*. Need not be so—my chess model is theory-driven and "severely underfitted."

Fourth Curveball—Clearing Hash Does Matter

- Retaining hash apparently also retards "later-blooming" moves.
- Effect only 0.25–0.35%, not 2–3%, but significant now.
- Clearing is better for scientific reproducibility but further from actual playing conditions.

Thus my original "simple and self-evident" model needs substantial adjustment for all of these factors—to say nothing of factors I caught at the beginning...

To conclude on a philosophic note: "Big Data" is critiqued for abandoning *theory*. Need not be so—my chess model is theory-driven and "severely underfitted." *But theory cannot abandon data*

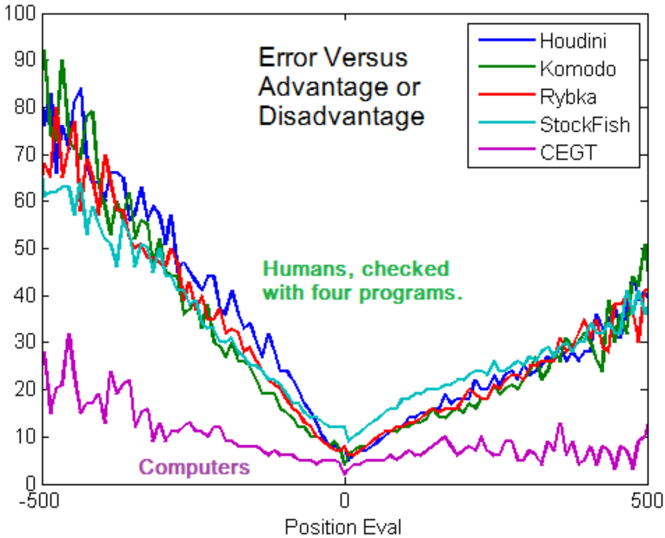
Fourth Curveball—Clearing Hash Does Matter

- Retaining hash apparently also retards "later-blooming" moves.
- Effect only 0.25–0.35%, not 2–3%, but significant now.
- Clearing is better for **scientific reproducibility** but further from actual playing conditions.

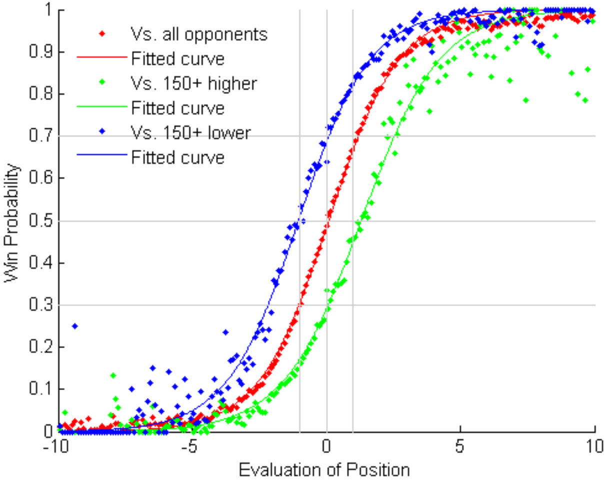
> Thus my original "simple and self-evident" model needs substantial adjustment for all of these factors—to say nothing of factors I caught at the beginning...

To conclude on a philosophic note: "Big Data" is critiqued for abandoning *theory*. Need not be so—my chess model is theory-driven and "severely underfitted." *But theory cannot abandon data*—nor a full understanding of the *history* and *hidden biases* it may embody.

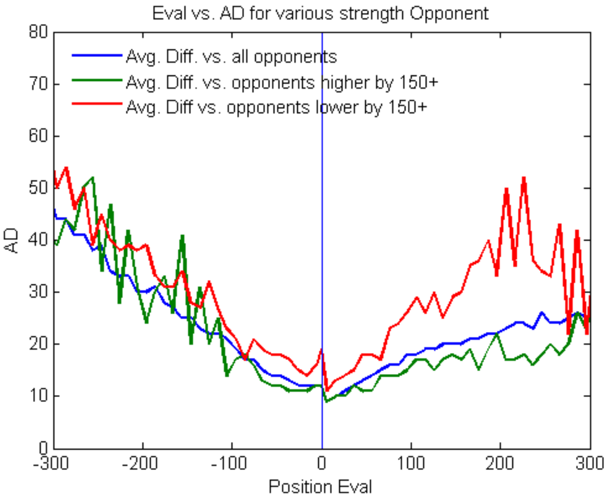
Human Versus Computer Phenomena



Human Versus Computer Phenomena



Eval-Error Curve With Unequal Players



Computer and Freestyle IPRs

Analyzed Ratings of Computer Engine Grand Tournament (on commodity PCs) and PAL/CSS Freestyle in 2007–08, plus the Thoresen Chess Engines Competition (16-core) Nov–Dec. 2013.

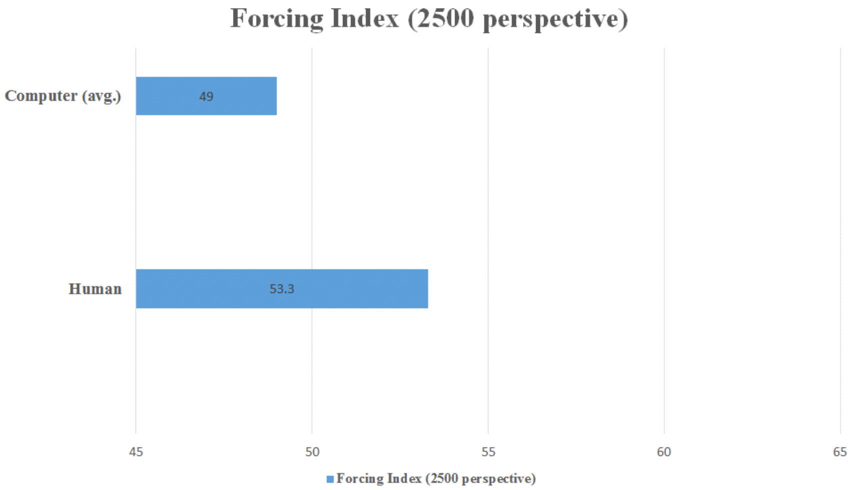
Event	Rating	2σ range	#gm	#moves
CEGT g1,50	3009	2962–3056	42	4,212
CEGT g25,26	2963	2921–3006	42	5,277
PAL/CSS 5ch	3102	3051–3153	45	3,352
PAL/CSS 6ch	3086	3038–3134	45	3,065
PAL/CSS 8ch	3128	3083–3174	39	3,057
TCEC 2013	3083	3062–3105	90	11,024

Computer and Freestyle IPRs—To Move 60

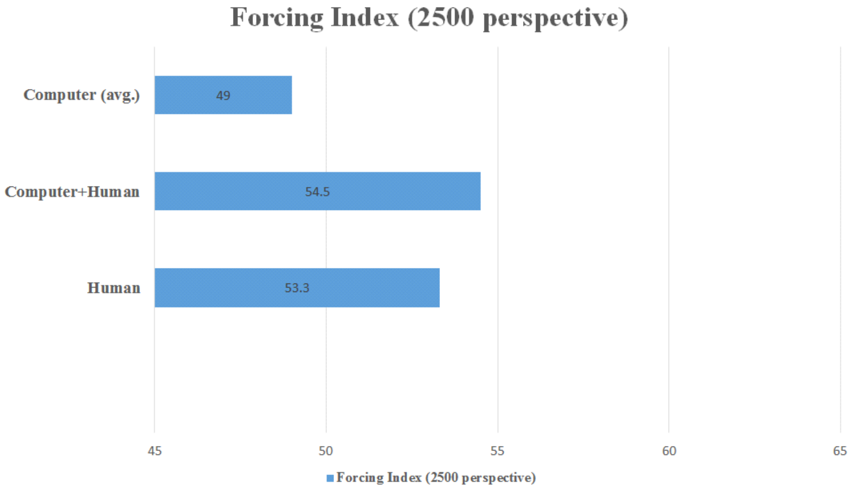
Computer games can go very long in dead drawn positions. TCEC uses a cutoff but CEGT did not. Human-led games tend to climax (well) before Move 60. This comparison halves the difference to CEGT, otherwise similar:

Sample set	Rating	2σ range	#gm	#moves
CEGT all	2985	2954–3016	84	9,489
PAL/CSS all	3106	3078–3133	129	9,474
TCEC 2013	3083	3062–3105	90	11,024
CEGT to60	3056	3023–3088	84	7,010
PAL/CSS to60	3112	3084–3141	129	8,744
TCEC to60	3096	3072–3120	90	8,184

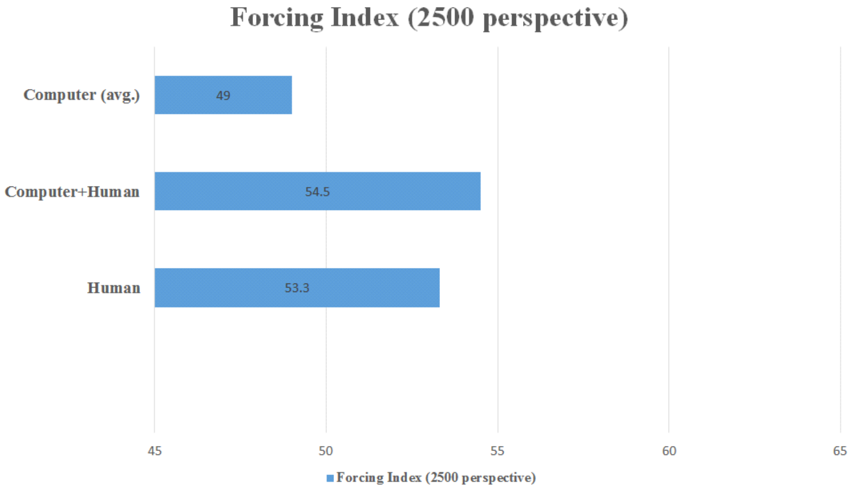
Degrees of Forcing Play



Add Human-Computer Tandems

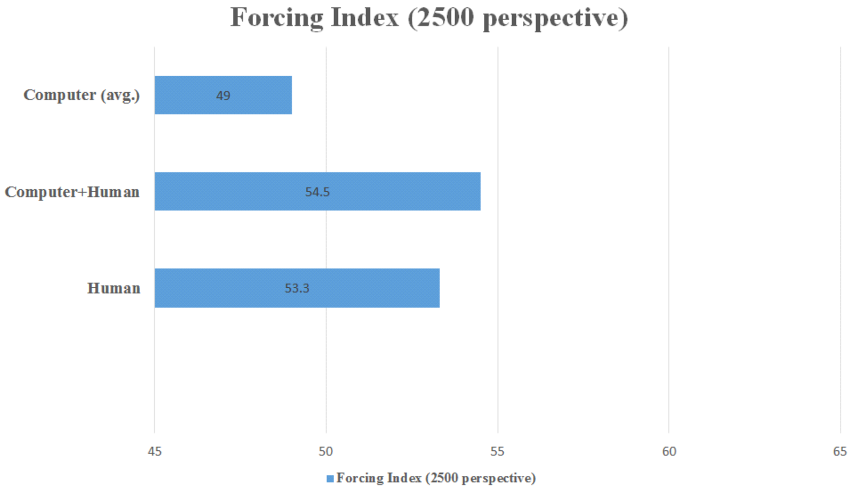


Add Human-Computer Tandems



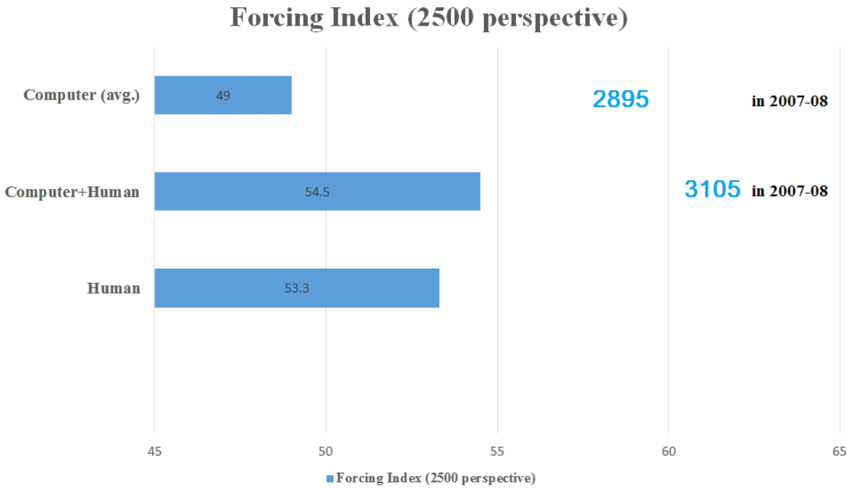
Evidently the humans called the shots.

Add Human-Computer Tandems



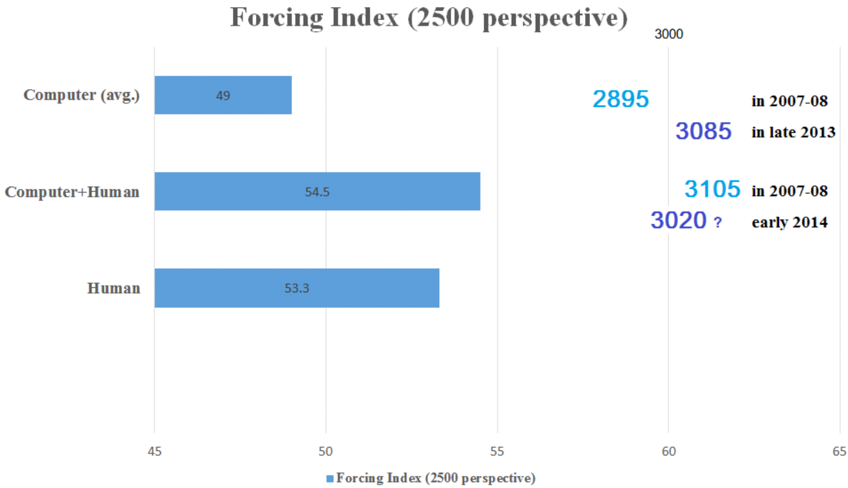
Evidently the humans called the shots. But how did they play?

2007–08 Freestyle Performance



Adding 210 Elo was significant. Forcing but good teamwork.

2014 Freestyle Tournament Performance



Tandems had marginally better W-L, but quality not clear...