Psychometric analysis of LLM responses

Table of contents

Temperature optimization 7
Response distribution analysis 10
Unoptimized KL divergence 10
Optimized KL divergence 12
CTT analysis 13
IRT analysis 20


```
import re
from glob import glob
from pathlib import Path

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import scipy
import seaborn as sns
import torch
from tqdm import tqdm

sns.set_theme()
plt.rcParams["font.family"] = "FreeSans"

ITEMS_DIR = Path("../data/output")
RESPONSES_DIR = Path("../llm-responses/output")
FIGURES_DIR = Path("../paper/figures")

MODELS = {
    "Llama-3.2-1B": "Llama",
    "Llama-3.2-3B": "Llama",
```

```
    "Llama-3.1-8B": "Llama",
    "Llama-3.1-70B": "Llama",
    "Llama-3.3-70B": "Llama",
    "OLMo-2-7B": "OLMo",
    "OLMo-2-13B": "OLMo",
    "Phi-3-3.8B": "Phi",
    "Phi-3-7B": "Phi",
    "Phi-3-14B": "Phi",
    "Phi-4-14B": "Phi",
    "Qwen2.5-0.5B": "Qwen",
    "Qwen2.5-1.5B": "Qwen",
    "Qwen2.5-3B": "Qwen",
    "Qwen2.5-7B": "Qwen",
    "Qwen2.5-14B": "Qwen",
    "Qwen2.5-32B": "Qwen",
    "Qwen2.5-72B": "Qwen",
    "OracleBaseline": "Baseline",
    "UniformBaseline": "Baseline",
    "HumanMode": "Baseline",
}
MODEL_FAMILY_COLORS = {
    "Llama": "#4c72b0",
    "OLMo": "#dd8452",
    "Phi": "#55a868",
    "Qwen": "#c44e52",
    "Baseline": "#8c8c8c",
}
MODEL_PALETTE = [MODEL_FAMILY_COLORS[family] for model, family in MODELS.items()]
```

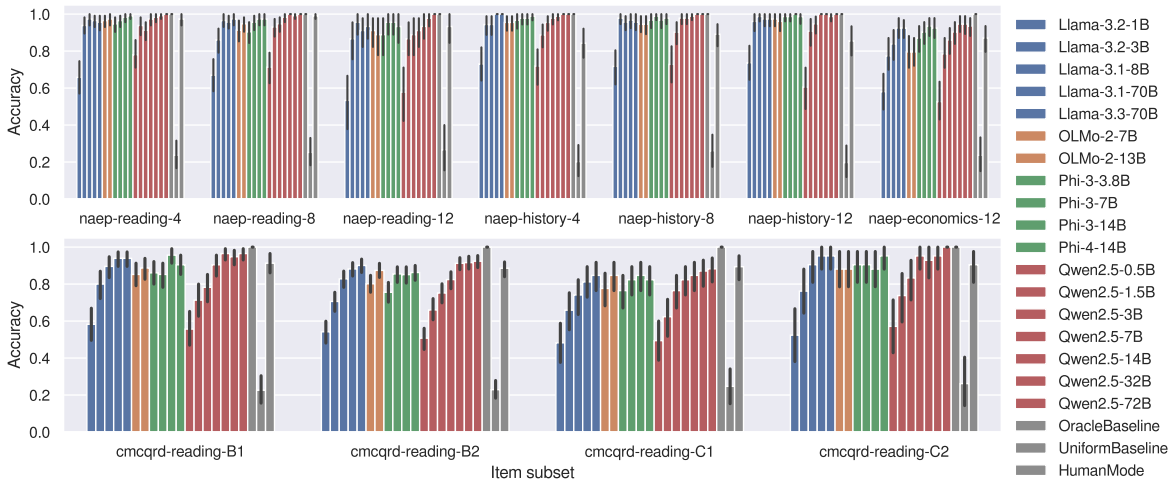
```
def load_data(dataset, subject):
    items_df = pd.read_json(ITEMS_DIR / f"{dataset}-{subject}-items.jsonl", lines=True)

    response_distributions_df = pd.read_json(
        ITEMS_DIR / f"{dataset}-{subject}-response-distributions.jsonl", lines=True
    )

    responses_dfs = []
    for filename in glob(
        f"{dataset}-{subject}-responses-*.jsonl", root_dir=RESPONSES_DIR
    ):
        (model,) = re.match(r".+-.+-responses-(.+)\.jsonl", filename).groups()
        df = pd.read_json(RESPONSES_DIR / filename, lines=True)
```

```
        df["model"] = model
        responses_dfs.append(df)
    responses_df = pd.concat(responses_dfs)
    responses_df["dataset"] = dataset
    responses_df["subject"] = subject

    responses_df = items_df.merge(response_distributions_df, on="item_id").merge(
        responses_df, on="item_id"
    )

    if dataset == "naep":
        responses_df.sort_values(["grade"], inplace=True)
        responses_df["subset"] = (
            responses_df[["dataset", "subject", "grade"]]
            .astype(str)
            .agg("-".join, axis=1)
        )
    elif dataset == "cmcqrd":
        responses_df.sort_values(["level"], inplace=True)
        responses_df["subset"] = (
            responses_df[["dataset", "subject", "level"]]
            .astype(str)
            .agg("-".join, axis=1)
        )

    return responses_df


naep_data = pd.concat(
    [
        load_data(dataset, subject)
        for dataset, subject in [
            ("naep", "reading"),
            ("naep", "history"),
            ("naep", "economics"),
        ]
    ]
)
cmcqrd_data = load_data("cmcqrd", "reading")

for data in [naep_data, cmcqrd_data]:
    # Ensure consistent order of subsets and models
```

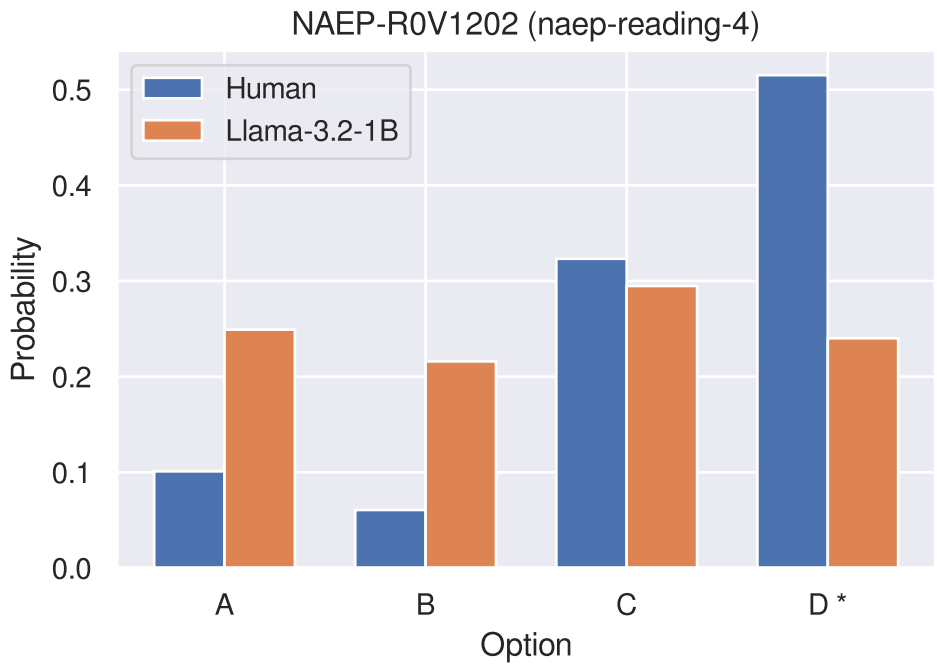
```
    subsets = data["subset"].unique()
    data["subset"] = data["subset"].astype("category").cat.set_categories(subsets)
    data["model"] = data["model"].astype("category").cat.set_categories(MODELS.keys())
    assert data["model"].notna().all(), "Missing model"
    data["model_family"] = data["model"].map(MODELS)
    data.sort_values(["subset", "model"], inplace=True)
    data.reset_index(drop=True, inplace=True)

    data["human_probs"] = data["response_distribution"].apply(
        lambda response_distribution: np.array(response_distribution)
        / sum(response_distribution)
    )
    data["model_probs"] = data["logits"].apply(
        lambda logits: scipy.special.softmax(logits, axis=1).mean(axis=0)
    )
    data["model_correct"] = data.apply(
        lambda row: np.argmax(row["model_probs"]) == row["correct_option_index"], axis=1
    )
```

```
# Accuracy
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(12, 5.5), sharey=True)

# Add HumanMode as a model
naep_accuracy_models = naep_data[["subset", "model", "model_correct"]].copy()
naep_accuracy_human = naep_data.groupby(["subset", "item_id"], observed=True).first().reset_
naep_accuracy_human["model"] = "HumanMode"
naep_accuracy_human["model_correct"] = naep_accuracy_human.apply(
    lambda row: np.argmax(row["response_distribution"]) == row["correct_option_index"], axis=
)
naep_accuracy_all = pd.concat([naep_accuracy_models, naep_accuracy_human])

cmcqrd_accuracy_models = cmcqrd_data[["subset", "model", "model_correct"]].copy()
cmcqrd_accuracy_human = cmcqrd_data.groupby(["subset", "item_id"], observed=True).first().res
cmcqrd_accuracy_human["model"] = "HumanMode"
cmcqrd_accuracy_human["model_correct"] = cmcqrd_accuracy_human.apply(
    lambda row: np.argmax(row["response_distribution"]) == row["correct_option_index"], axis=
)
cmcqrd_accuracy_all = pd.concat([cmcqrd_accuracy_models, cmcqrd_accuracy_human])

# Hatch for human mode
sns.barplot(
    data=naep_accuracy_all,
```

```
    x="subset",
    y="model_correct",
    hue="model",
    palette=MODEL_PALETTE,
    err_kws={"linewidth": 1.5},
    ax=ax1,
)
ax1.set_xlabel("")
ax1.set_ylabel("Accuracy")

sns.barplot(
    data=cmcqrd_accuracy_all,
    x="subset",
    y="model_correct",
    hue="model",
    palette=MODEL_PALETTE,
    legend=False,
    ax=ax2,
)
ax2.set_xlabel("Item subset")
ax2.set_ylabel("Accuracy")

ax1.legend(loc="upper left", bbox_to_anchor=(1, 1), frameon=False)

fig.savefig(FIGURES_DIR / "accuracy.pdf", bbox_inches="tight")
```



```python
def plot_probs(
    response,
    model_probs_col="model_probs",
    human_probs_col="human_probs",
    reference_prob_col=None,
    reference_prob_label="Reference",
):
    human_probs = response[human_probs_col]
    model_probs = response[model_probs_col]
    fig, ax = plt.subplots()
    bar_width = 0.35
    index = np.arange(len(human_probs))
    ax.bar(index, human_probs, bar_width, label="Human")
    ax.bar(index + bar_width, model_probs, bar_width, label=response["model"])
    if reference_prob_col:
        reference_prob = response[reference_prob_col]
        ax.plot(
            [-0.5, len(human_probs) - 0.5],
            [reference_prob, reference_prob],
            color="black",
            linestyle="--",
            label=reference_prob_label,
        )
    ax.set_xlabel("Option")
    ax.set_ylabel("Probability")
    ax.set_title(f"{response['item_id']} ({response['subset']})")
    ax.set_xticks(index + bar_width / 2)
    labels = ["A", "B", "C", "D"]
    labels[response["correct_option_index"]] += " *"
    ax.set_xticklabels(labels)
    ax.legend()


plot_probs(naep_data.iloc[0])
```



Temperature optimization

We optimize the temperature separately for each item subset (subject and NAEP grade or CMCQRD level) to minimize the KL divergence between the LLM response distribution and the human response distribution.

> **NOTE:** We optimize on the same data that we evaluate on, in order to get an optimistic estimate / upper bound of psychometric plausibility in the following analyses.

```
def optimize_temperature(logits, human_probs, init=1.0):
    logits = torch.tensor(list(logits))
    human_probs = torch.tensor(list(human_probs))
    temperature = torch.tensor(init, requires_grad=True)

    optimizer = torch.optim.Adam([temperature], lr=0.01)
    best_loss = float("inf")
    steps_without_improvement = 0

    while True:
        optimizer.zero_grad()
```

```
        tempered_logits = logits / temperature
        model_logprobs = torch.nn.functional.log_softmax(tempered_logits, dim=-1)
        model_logprobs = model_logprobs.mean(dim=-2)

        loss = torch.nn.functional.kl_div(
            model_logprobs, human_probs.clone(), reduction="batchmean"
        )
        loss.backward()
        optimizer.step()

        if loss.item() >= best_loss:
            steps_without_improvement += 1
            if steps_without_improvement > 100:
                break
        else:
            steps_without_improvement = 0
            best_loss = loss.item()

    return temperature.item()


def grouped_optimize_temperature(df):
    optimized_temperatures = pd.Series(index=df.index, dtype=float)
    for _, group in tqdm(
        df.groupby(["subset", "model"], observed=True), desc="Optimizing temperatures"
    ):
        optimized_temperatures[group.index] = optimize_temperature(
            group["logits"], group["human_probs"]
        )
    return optimized_temperatures


for data in [naep_data, cmcqrd_data]:
    data["optimized_temperature"] = grouped_optimize_temperature(data)

    data["optimized_model_probs"] = data.apply(
        lambda row: scipy.special.softmax(
            np.array([*row["logits"]]) / row["optimized_temperature"], axis=1
        ).mean(axis=0),
        axis=1,
    )
```

```
Optimizing temperatures:   0%|          | 0/140 [00:00<?, ?it/s]/tmp/ipykernel_3846470/177978
  human_probs = torch.tensor(list(human_probs))
Optimizing temperatures:   1%|          | 1/140 [00:04<09:28,  4.09s/it]Optimizing temperatu
Optimizing temperatures:   0%|          | 0/80 [00:00<?, ?it/s]Optimizing temperatures:   1%
```

```python
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(12, 5.5), sharey=True)

sns.barplot(
    data=naep_data,
    x="subset",
    y="optimized_temperature",
    hue="model",
    errorbar=None,
    palette=MODEL_PALETTE,
    ax=ax1,
)
ax1.set_xlabel("")
ax1.set_ylabel("Optimized temperature")

sns.barplot(
    data=cmcqrd_data,
    x="subset",
    y="optimized_temperature",
    hue="model",
    errorbar=None,
    palette=MODEL_PALETTE,
    legend=False,
    ax=ax2,
)
ax2.set_xlabel("Item subset")
ax2.set_ylabel("Optimized temperature")

ax1.legend(loc="upper left", bbox_to_anchor=(1, 1), frameon=False)

fig.savefig(FIGURES_DIR / "temperature.pdf", bbox_inches="tight")
```

Response distribution analysis

In this analysis, we compare the token probability distribution produced by the LLMs to the human response distribution.

```
for data in [naep_data, cmcqrd_data]:
    data["kldiv"] = data.apply(
        lambda row: scipy.stats.entropy(row["human_probs"], row["model_probs"]), axis=1
    )
    data["optimized_kldiv"] = data.apply(
        lambda row: scipy.stats.entropy(
            row["human_probs"], row["optimized_model_probs"]
        ),
        axis=1,
    )
```

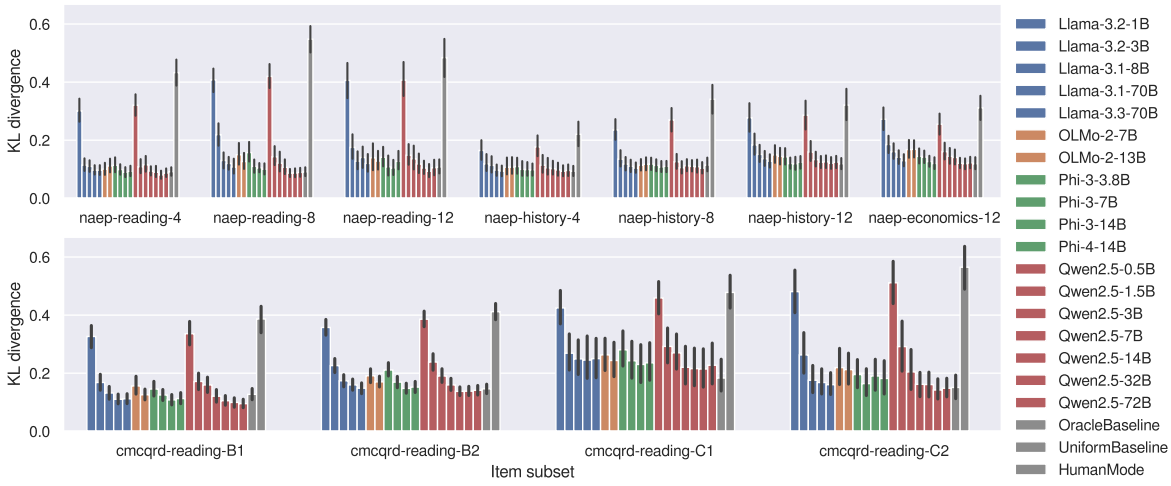
Unoptimized KL divergence

```
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(12, 5.5), sharey=True)

sns.barplot(
    data=naep_data,
    x="subset",
    y="kldiv",
    hue="model",
```

```
    palette=MODEL_PALETTE,
    err_kws={"linewidth": 1.5},
    ax=ax1,
)
ax1.set_xlabel("")
ax1.set_ylabel("KL divergence")

sns.barplot(
    data=cmcqrd_data,
    x="subset",
    y="kldiv",
    hue="model",
    palette=MODEL_PALETTE,
    legend=False,
    ax=ax2,
)
ax2.set_xlabel("Item subset")
ax2.set_ylabel("KL divergence")

ax1.legend(loc="upper left", bbox_to_anchor=(1, 1), frameon=False)

fig.savefig(FIGURES_DIR / "kldiv-unoptimized.pdf", bbox_inches="tight")
```

Optimized KL divergence

```
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(12, 5.5), sharey=True)

sns.barplot(
    data=naep_data,
    x="subset",
    y="optimized_kldiv",
    hue="model",
    palette=MODEL_PALETTE,
    err_kws={"linewidth": 1.5},
    ax=ax1,
)
ax1.set_xlabel("")
ax1.set_ylabel("KL divergence")

sns.barplot(
    data=cmcqrd_data,
    x="subset",
    y="optimized_kldiv",
    hue="model",
    palette=MODEL_PALETTE,
    legend=False,
    ax=ax2,
)
ax2.set_xlabel("Item subset")
ax2.set_ylabel("KL divergence")

ax1.legend(loc="upper left", bbox_to_anchor=(1, 1), frameon=False)

fig.savefig(FIGURES_DIR / "kldiv-optimized.pdf", bbox_inches="tight")
```



CTT analysis

In this analysis, we compare the item facility values (i.e., the proportion of students who answered each item correctly) to the probability of the correct answer option produced by the LLMs. Psychometrically plausible LLM responses should be more confidently correct with easier items than with more difficult items.

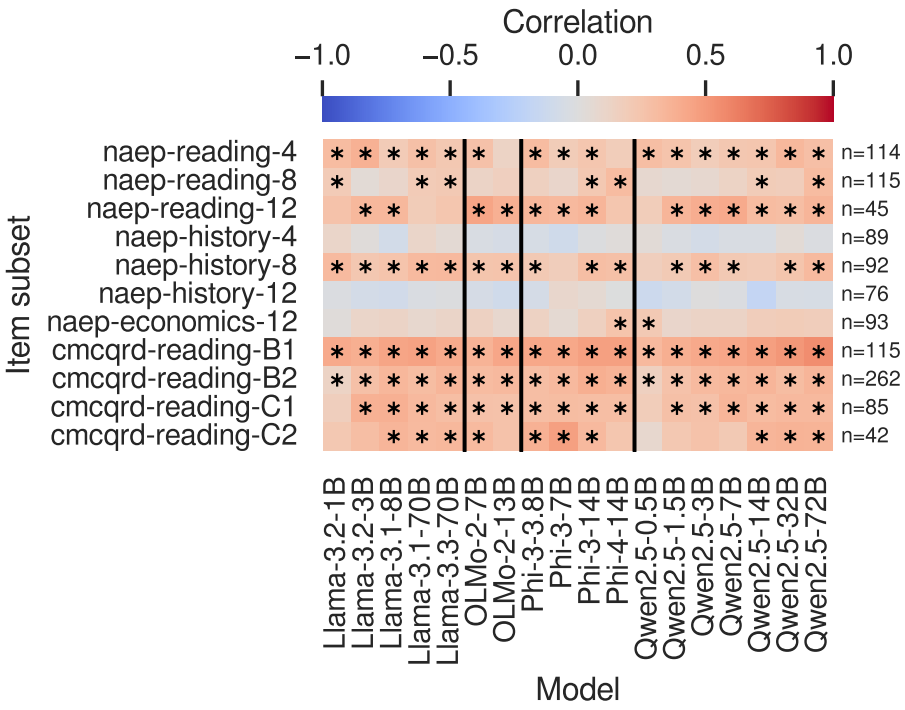
```
# Baselines are not useful for comparison (same probability for all items)
naep_data = naep_data[~naep_data["model"].str.endswith("Baseline")]
cmcqrd_data = cmcqrd_data[~cmcqrd_data["model"].str.endswith("Baseline")]

for data in [naep_data, cmcqrd_data]:
    data["model"] = data["model"].cat.remove_unused_categories()

    data["human_correct_prob"] = data.apply(
        lambda row: row["human_probs"][row["correct_option_index"]], axis=1
    )
    data["model_correct_prob"] = data.apply(
        lambda row: row["model_probs"][row["correct_option_index"]], axis=1
    )
    data["optimized_model_correct_prob"] = data.apply(
        lambda row: row["optimized_model_probs"][row["correct_option_index"]], axis=1
    )

all_data = pd.concat([naep_data, cmcqrd_data])
all_data["subset"] = (
    all_data["subset"]
```

```
    .astype("category")
    .cat.set_categories(
        naep_data["subset"].cat.categories.append(cmcqrd_data["subset"].cat.categories)
    )
)
```

```
# Correlation between LLM and human response probabilities
correct_probs_human_corr = (
    all_data.groupby(["subset", "model"], observed=True)
    .apply(
        lambda group: scipy.stats.pearsonr(
            group["human_correct_prob"],
            group["optimized_model_correct_prob"],
            alternative="two-sided",
        )
    )
    .apply(pd.Series, index=["correlation", "p-value"])
).unstack()

# Heatmap of correlations
fig, ax = plt.subplots(figsize=(7.0, 4.0))
sns.heatmap(
    correct_probs_human_corr["correlation"],
    annot=(correct_probs_human_corr["p-value"] < 0.05).replace({True: "*", False: ""}),
    annot_kws={"color": "black"},
    fmt="",
    cmap="coolwarm",
    center=0,
    vmin=-1,
    vmax=1,
    cbar_kws={"label": "Correlation", "location": "top", "fraction": 0.072},
    square=True,
    ax=ax,
)
ax.set_xlabel("Model")
ax.set_ylabel("Item subset")

# Lines between model families
for i, (model, next_model) in enumerate(
    zip(
        correct_probs_human_corr["correlation"].columns[:-1],
        correct_probs_human_corr["correlation"].columns[1:],
```

```
    )
):
    if MODELS.get(model) != MODELS.get(next_model):
        ax.axvline(i + 1, color="black")

# Sample size per subset, and model
sample_size = (
    all_data.groupby(["subset", "model"], observed=True).size().unstack().iloc[:, 0]
)
for i, index in enumerate(correct_probs_human_corr.index):
    ax.annotate(
        f"n={sample_size[index]:.0f}",
        xy=(len(correct_probs_human_corr["correlation"].columns) + 0.3, i + 0.5),
        ha="left",
        va="center",
        annotation_clip=False,
        size=8,
    )

fig.tight_layout()
fig.savefig(FIGURES_DIR / "ctt-correlation.pdf", bbox_inches="tight")
```

```
/tmp/ipykernel_3846470/2245631322.py:4: DeprecationWarning: DataFrameGroupBy.apply operated o
  .apply(
```



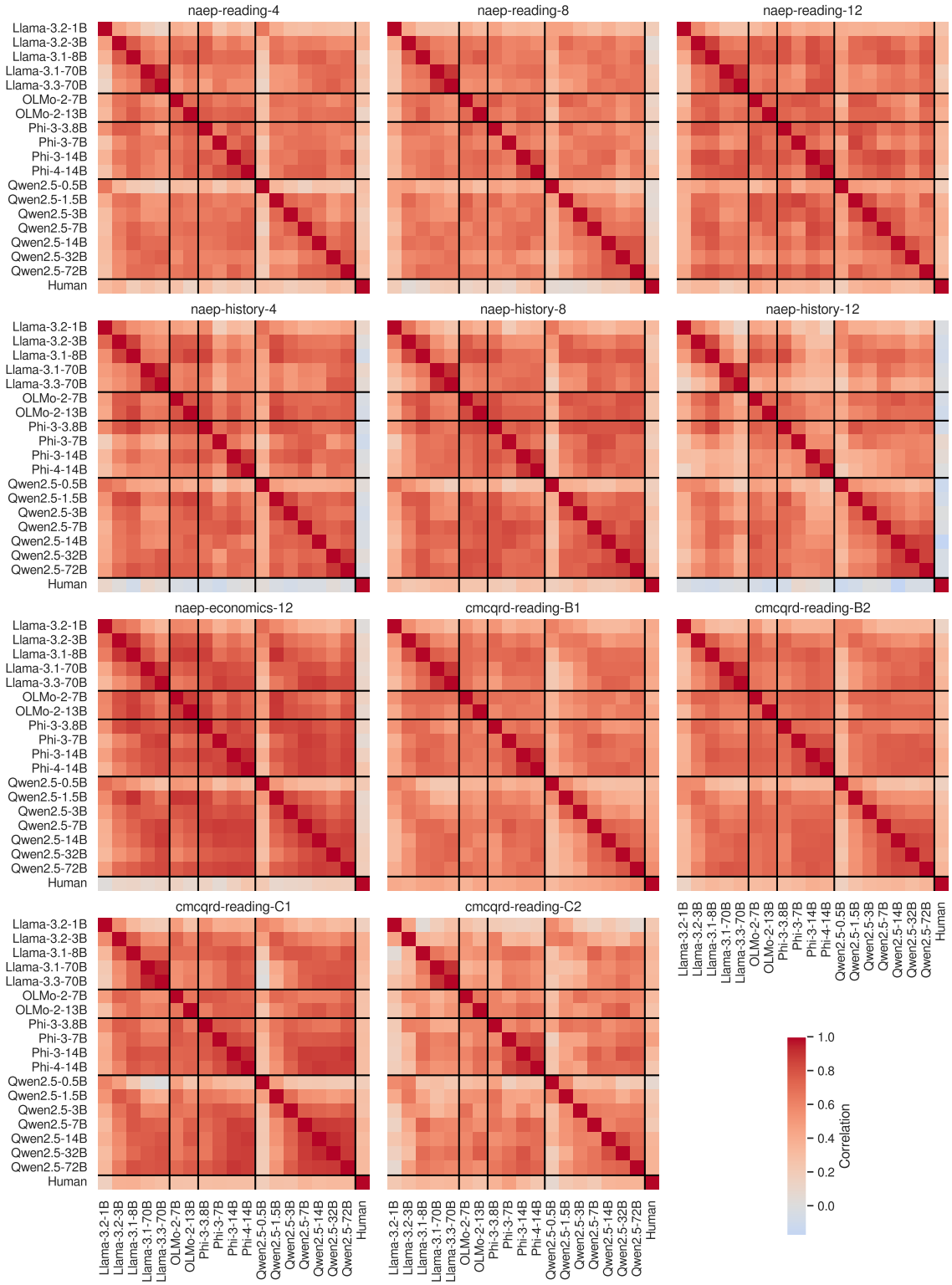

```
# Subset with strongest correlation
strongest_subset = correct_probs_human_corr["correlation"].max(axis="columns").idxmax()
correct_probs_human_corr.loc[strongest_subset, "correlation"]
```

```
model
Llama-3.2-1B     0.408693
Llama-3.2-3B     0.388440
Llama-3.1-8B     0.439477
Llama-3.1-70B    0.457404
Llama-3.3-70B    0.412323
OLMo-2-7B        0.366660
OLMo-2-13B       0.432958
Phi-3-3.8B       0.422561
Phi-3-7B         0.424779
Phi-3-14B        0.465331
Phi-4-14B        0.465496
Qwen2.5-0.5B     0.323238
Qwen2.5-1.5B     0.386761
Qwen2.5-3B       0.419301
Qwen2.5-7B       0.432884
Qwen2.5-14B      0.472846
```

```
Qwen2.5-32B      0.508665
Qwen2.5-72B      0.557089
Name: cmcqrd-reading-B1, dtype: float64
```

```python
# Full correlation matrices for all LLMs and human
correct_probs = all_data.pivot(
    index=["item_id", "subset"], columns="model", values="optimized_model_correct_prob"
)
correct_probs["Human"] = all_data.groupby(["item_id", "subset"])[
    "human_correct_prob"
].first()
correct_probs_corr = correct_probs.groupby("subset").corr()
g = sns.FacetGrid(
    correct_probs_corr.reset_index("subset"), col="subset", col_wrap=3, height=4
)


def heatmap(data, **kwargs):
    data = data.drop(columns="subset")
    sns.heatmap(data, **kwargs)


g.map_dataframe(heatmap, cmap="coolwarm", center=0, cbar=False, square=True)
g.set_titles("{col_name}")

# Color bar
cbar_ax = g.figure.add_axes([0.82, 0.06, 0.02, 0.15])
g.figure.colorbar(g.facet_axis(0, 5).collections[0], cax=cbar_ax, label="Correlation")

# Lines between model families
for ax in g.axes:
    ax.set_xlabel("")
    ax.set_ylabel("")
    for i, (model, next_model) in enumerate(
        zip(correct_probs_corr.columns[:-1], correct_probs_corr.columns[1:])
    ):
        if MODELS.get(model) != MODELS.get(next_model):
            ax.axvline(i + 1, color="black")
            ax.axhline(i + 1, color="black")

g.figure.subplots_adjust(hspace=0.1)
g.savefig(FIGURES_DIR / "ctt-correlation-full.pdf", bbox_inches="tight")
```

```
/tmp/ipykernel_3846470/1301630354.py:5: FutureWarning: The default of observed=False is depre
  correct_probs["Human"] = all_data.groupby(["item_id", "subset"])[
/tmp/ipykernel_3846470/1301630354.py:8: FutureWarning: The default of observed=False is depre
  correct_probs_corr = correct_probs.groupby("subset").corr()
```


naep-reading-4
naep-reading-8
naep-reading-12
naep-history-4
naep-history-8
naep-history-12
naep-economics-12
cmcqrd-reading-B1
cmcqrd-reading-B2
cmcqrd-reading-C1
cmcqrd-reading-C2
Llama-3.2-1B
Llama-3.2-3B
Llama-3.1-8B
Llama-3.1-70B
Llama-3.3-70B
OLMo-2-7B
OLMo-2-13B
Phi-3-3.8B
Phi-3-7B
Phi-3-14B
Phi-4-14B
Qwen2.5-0.5B
Qwen2.5-1.5B
Qwen2.5-3B
Qwen2.5-7B
Qwen2.5-14B
Qwen2.5-32B
Qwen2.5-72B
Human
Correlation
1.0
0.8
0.6
0.4
0.2
0.0

IRT analysis

In this analysis, we study how well the LLM response probabilities fit a three-parameter logistic IRT model fitted to human responses. Specifically, we view each LLM as a single test taker with $\theta = 0$ (i.e., average ability) and check whether this test taker's response probabilities correlate with the expected probabilities from the IRT model.

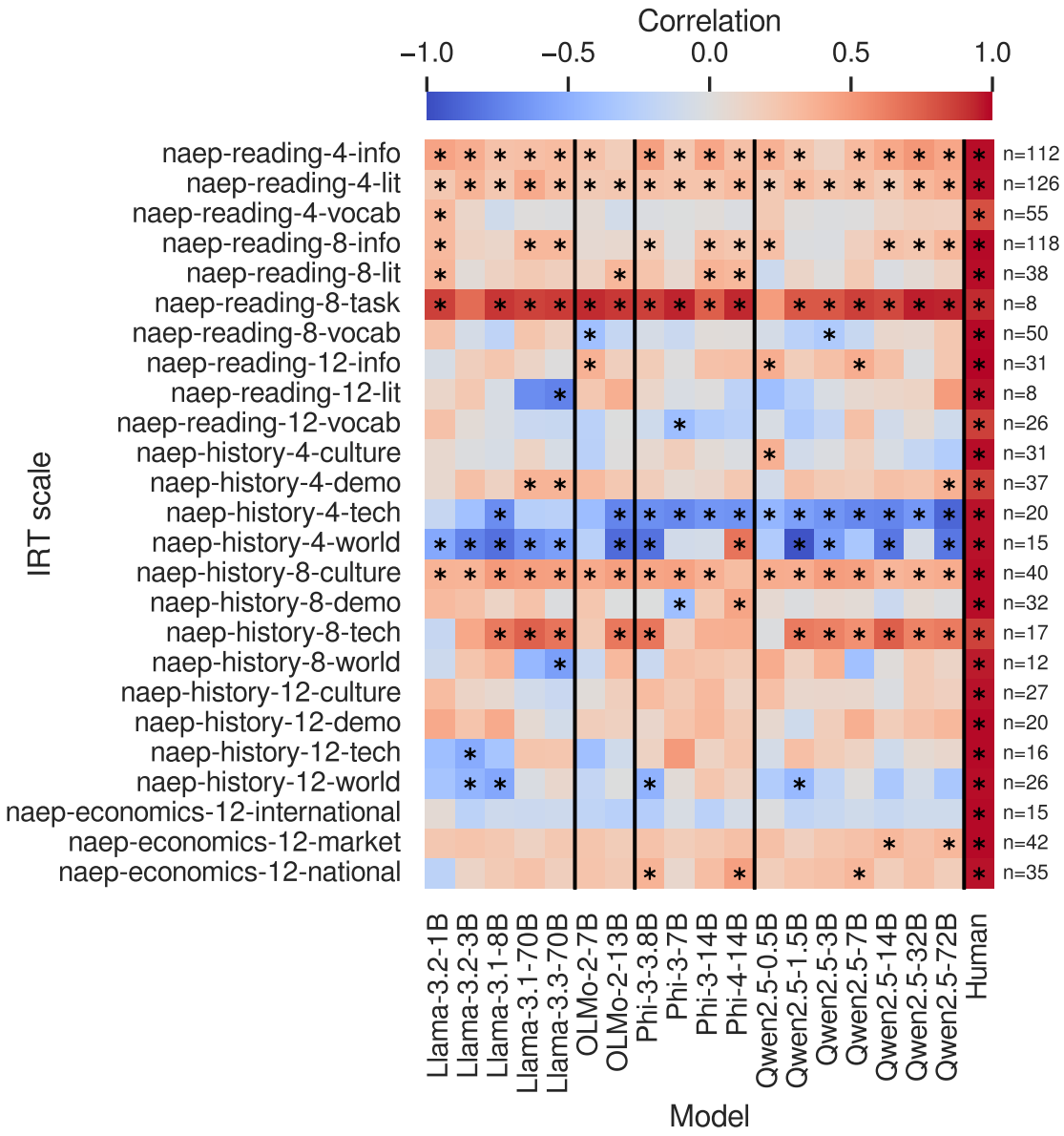
> **NOTE:** This analysis is only applicable to NAEP data, as CMCQRD data only contains rescaled difficulty parameters, meaning that we cannot reconstruct the response probability for the average test taker.

```
naep_parameters_df = pd.read_json(ITEMS_DIR / "naep-parameters.jsonl", lines=True)
naep_parameters_df = naep_parameters_df[naep_parameters_df["irt_model"] == "3pl"].drop(
    columns=["irt_model", "d_1", "d_2", "d_3", "d_4"]
)
naep_irt_data = naep_data.merge(naep_parameters_df, on=["item_id", "subject", "grade"])


# Expected probability of correct response at theta = 0
def p_at_theta0(a, b, c):
    return c + (1 - c) / (1 + np.exp(-1.7 * a * (0 - b)))


naep_irt_data["p_at_theta0"] = naep_irt_data.apply(
    lambda row: p_at_theta0(row["a"], row["b"], row["c"]), axis=1
)
```

```
# Correlation between LLM response probabilities and IRT model probabilities
correct_probs_p_corr = (
    naep_irt_data.groupby(["subset", "scale", "model"], observed=True)
    .apply(
        lambda group: scipy.stats.pearsonr(
            group["p_at_theta0"],
            group["optimized_model_correct_prob"],
            alternative="two-sided",
        ),
        include_groups=False,
    )
    .apply(pd.Series, index=["correlation", "p-value"])
).unstack()

# Correlation between human response probabilities and IRT model probabilities
correct_probs_p_human_corr = (
```

```
    naep_irt_data
    # Deduplicate response distributions (keep only one model)
    .groupby(["subset", "scale", "item_id", "year"], observed=True)
    .first()
    # Correlate
    .groupby(["subset", "scale"], observed=True)
    .apply(
        lambda group: scipy.stats.pearsonr(
            group["p_at_theta0"],
            group["human_correct_prob"],
            alternative="two-sided",
        ),
        include_groups=False,
    )
    .apply(pd.Series, index=["correlation", "p-value"])
)
correct_probs_p_corr["correlation", "Human"] = correct_probs_p_human_corr["correlation"]
correct_probs_p_corr["p-value", "Human"] = correct_probs_p_human_corr["p-value"]

# Heatmap of correlations
fig, ax = plt.subplots(figsize=(7.5, 6.5))
g = sns.heatmap(
    correct_probs_p_corr["correlation"],
    annot=(correct_probs_p_corr["p-value"] < 0.05).replace({True: "*", False: ""}),
    annot_kws={"color": "black"},
    fmt="",
    cmap="coolwarm",
    center=0,
    vmin=-1,
    vmax=1,
    cbar_kws={"label": "Correlation", "location": "top", "fraction": 0.0355, "pad": 0.025},
    square=True,
    ax=ax,
)
ax.set_xlabel("Model")
ax.set_ylabel("IRT scale")
ax.yaxis.labelpad = -20
ax.set_yticklabels(
    [
        f"{subset}-{scale}" if scale else subset
        for subset, scale in correct_probs_p_corr.index
    ]
```

```
)

# Lines between model families
for i, (model, next_model) in enumerate(
    zip(
        correct_probs_p_corr["correlation"].columns[:-1],
        correct_probs_p_corr["correlation"].columns[1:],
    )
):
    if MODELS.get(model) != MODELS.get(next_model):
        ax.axvline(i + 1, color="black")

# Sample size per subset, scale, and model
sample_size = (
    naep_irt_data.groupby(["subset", "scale", "model"], observed=True)
    .size()
    .unstack()
    .iloc[:, 0]
)
for i, index in enumerate(correct_probs_p_corr.index):
    ax.annotate(
        f"n={sample_size[index]:.0f}",
        xy=(len(correct_probs_p_corr["correlation"].columns) + 0.3, i + 0.5),
        ha="left",
        va="center",
        annotation_clip=False,
        size=8,
    )

fig.tight_layout()
fig.savefig(FIGURES_DIR / "irt-correlation.pdf", bbox_inches="tight")
```




```
# Regression plots of LLM response probabilities vs. IRT model probabilities
naep_irt_data["subject-scale"] = naep_irt_data["subject"] + "-" + naep_irt_data["scale"]
g = sns.lmplot(
    naep_irt_data,
    x="p_at_theta0",
    y="optimized_model_correct_prob",
    hue="model",
    col="subject-scale",
```

```
    row="grade",
)
g.set(xlim=(0, 1), ylim=(0, 1))
g.set_titles("NAEP {col_name}, grade {row_name}")
g.set_axis_labels(
    "Expected probability at $\\theta = 0$", "Observed response probability"
)

g.savefig(FIGURES_DIR / "irt-regression.pdf", bbox_inches="tight")
```