The round_model_performances api is straightforward. Round details and your scores arrive in a nice flat list, put them in a DataFrame and run historical score analysis to your heart's content.

It is now being replaced with round_model_performances_v2, which has all the new score types but requires a bit of unpacking. Here are a couple functions that may come in handy.

If you just want to flatten the result from round_model_performances_v2:

def unpack_rmp(packed_rmp: list) -> list: """Unpack the submissionScores from round_model_performances_v2.

```
Args:
    packed_rmp: the response from a call to round_model_performances_v2

Returns:
    list of dicts: list of round entries with all scores pulled up into the round dict
"""
flat_rmp = []
for row in packed_rmp:
    flat_row = {}

    for akey in row:
        flat_row[akey] = row[akey]

    del flat_row["submissionScores"]

    for ascore in row["submissionScores"] or []:
        name = ascore["displayName"]

        flat_row[name] = ascore["value"]
        flat_row[f"{name}Percentile"] = ascore["percentile"]

        flat_row["date"] = ascore["date"]
        flat_row["day"] = ascore["day"]
        flat_row["payoutPending"] = ascore["payoutPending"]
        flat_row["payoutSettled"] = ascore["payoutSettled"]

    flat_rmp.append(flat_row)

return flat_rmp
```

Using it as below gives you a nice flat scores DataFrame the same way the retiring round_model_performances does:

rmp = napi.round_model_performances_v2("e48dabf1-d699-42b2-9074-63aa06d797d9") flat = unpack_rmp(rmp) pd.DataFrame(flat).set_index("roundNumber").sort_index().dropna(subset="v2_corr20")

If you are like me, and want to minimize changes through your codebase, you could add two more functions:

def get_model_id(api, model_name: str) -> str: """Look up the model_id for a given model_name

```
Args:
    api: an instance of numerapi.NumerAPI or numerapi.SignalsAPI
    model_name: human readable name of the model

Returns:
    Model UUID
"""
if api.tournament_id == 8:
    endpoint = "v3UserProfile"
elif api.tournament_id == 11:
    endpoint = "v2SignalsProfile"

query = f"""
  query($model_name: String!) {{
    {endpoint}(modelName: $model_name) {{
      id
    }}
  }}
"""
arguments = {"model_name": model_name}
response = api.raw_query(query, arguments)
id = response["data"][endpoint]["id"]

return id
```

def round_model_performances_v2_flat(api, model_name: str): """A wrapper for round_model_performances_v2 to unpack scores into flat format.

```
Args:
    api: an instance of numerapi.NumerAPI or numerapi.SignalsAPI
```

```
    model_name: human readable name of the model

Returns:
    list of dicts: list of round entries with all scores pulled up into the round dict
"""
model_id = get_model_id(api, model_name)
rmp = api.round_model_performances_v2(model_id)
return unpack_rmp(rmp)
```

And use it as an (almost) drop-in replacement. Where you used to have:

napi.round_model_performances("v42_example_preds")

you can now use

round_model_performances_v2_flat(napi, "v42_example_preds")

I say almost a drop-in replacement because of some fields changed names, e.g. corr20V2

in current API became v2_corr20

in the new API. Still, a much smaller refactoring than switching to model_ids or unpacking the data everywhere I need it.