Alles klar 👍 – ich verstehe, du möchtest mehrere **PHP-Skripte**, die direkt aus deiner Datenbank JSON-Ausgaben erzeugen, so wie du es schon für die semantische Dichtekarte gemacht hast. Damit kannst du verschiedene im Kapitel 6 beschriebene FRZK-Operatoren (σ, M, R, E usw.) oder h(x,y,z)-Funktionen aus den Bewertungsdaten berechnen und im JSON-Format bereitstellen.

Ich skizziere dir drei zusätzliche PHP-Skripte:

## 1. ****JSON für semantische Dichtekarte (h(x,y))****

Dieses Skript berechnet Mittelwerte aller val\_\*-Spalten je Rasterzelle (x,y) und gibt sie als JSON aus. (Erweiterung deines bisherigen Skripts, nur strukturierter.)

<?php

// density\_map.php

header('Content-Type: application/json; charset=utf-8');

$pdo = new PDO("mysql:host=localhost;dbname=icas;charset=utf8mb4","root","");

// Parameter

$xbins = 12;

$ybins = 6;

$meta = [

"table" => "mtr\_rueckkopplung\_teilnehmer",

"x\_field" => "teilnehmer\_id",

"y\_field" => "gruppe\_id",

"val\_columns" => [

"mitarbeit","absprachen","selbststaendigkeit","konzentration",

"fleiss","lernfortschritt","beherrscht\_thema","transferdenken",

"basiswissen","vorbereitet","themenauswahl","materialien",

"methodenvielfalt","individualisierung","aufforderung","zielgruppen",

"emotions"

],

"aggregation" => "avg",

"xbins" => $xbins,

"ybins" => $ybins

];

// min/max ermitteln

$range = $pdo->query("SELECT MIN({$meta['x\_field']}) as xmin, MAX({$meta['x\_field']}) as xmax,

MIN({$meta['y\_field']}) as ymin, MAX({$meta['y\_field']}) as ymax

FROM {$meta['table']}")->fetch(PDO::FETCH\_ASSOC);

$dx = ($range['xmax'] - $range['xmin']) / $xbins;

$dy = ($range['ymax'] - $range['ymin']) / $ybins;

$result = [

"meta" => $meta,

"x\_labels" => [],

"y\_labels" => [],

"h" => [],

"counts" => []

];

// Achsen-Beschriftung

for ($i=0; $i<$xbins; $i++) {

$result["x\_labels"][] = ($range['xmin']+$i\*$dx) . ".." . ($range['xmin']+($i+1)\*$dx);

}

for ($j=0; $j<$ybins; $j++) {

$result["y\_labels"][] = ($range['ymin']+$j\*$dy) . ".." . ($range['ymin']+($j+1)\*$dy);

}

// Raster füllen

for ($i=0; $i<$xbins; $i++) {

$xlo = $range['xmin']+$i\*$dx;

$xhi = $xlo+$dx;

$rowH = []; $rowC = [];

for ($j=0; $j<$ybins; $j++) {

$ylo = $range['ymin']+$j\*$dy;

$yhi = $ylo+$dy;

$sql = "SELECT AVG((" . implode("+",$meta['val\_columns']) . ")/".count($meta['val\_columns']).") as hval, COUNT(\*) as c

FROM {$meta['table']}

WHERE {$meta['x\_field']} >= $xlo AND {$meta['x\_field']} < $xhi

AND {$meta['y\_field']} >= $ylo AND {$meta['y\_field']} < $yhi";

$cell = $pdo->query($sql)->fetch(PDO::FETCH\_ASSOC);

$rowH[] = $cell['c']>0 ? floatval($cell['hval']) : null;

$rowC[] = intval($cell['c']);

}

$result["h"][] = $rowH;

$result["counts"][] = $rowC;

}

echo json\_encode($result, JSON\_PRETTY\_PRINT|JSON\_UNESCAPED\_UNICODE);

## 2. ****JSON für Orientierungsverlust O(x,y)****

Hier wird aus density\_map.php die Gradientengröße berechnet und normalisiert → O=1−∣∇h∣/max⁡(∣∇h∣)O = 1 - |\nabla h| / \max(|\nabla h|)O=1−∣∇h∣/max(∣∇h∣).

<?php

// orientation\_loss.php

header('Content-Type: application/json; charset=utf-8');

$data = json\_decode(„{

"meta": {

"table": "mtr\_rueckkopplung\_teilnehmer",

"x\_field": "teilnehmer\_id",

"y\_field": "gruppe\_id",

"val\_columns": [

"mitarbeit",

"absprachen",

"selbststaendigkeit",

"konzentration",

"fleiss",

"lernfortschritt",

"beherrscht\_thema",

"transferdenken",

"basiswissen",

"vorbereitet",

"themenauswahl",

"materialien",

"methodenvielfalt",

"individualisierung",

"aufforderung",

"zielgruppen",

"emotions"

],

"aggregation": "avg",

"xbins": 12,

"ybins": 6

},

"x\_labels": [

"2..3.8333333333333",

"3.8333333333333..5.6666666666667",

"5.6666666666667..7.5",

"7.5..9.3333333333333",

"9.3333333333333..11.166666666667",

"11.166666666667..13",

"13..14.833333333333",

"14.833333333333..16.666666666667",

"16.666666666667..18.5",

"18.5..20.333333333333",

"20.333333333333..22.166666666667",

"22.166666666667..24"

],

"y\_labels": [

"1..2.3333333333333",

"2.3333333333333..3.6666666666667",

"3.6666666666667..5",

"5..6.3333333333333",

"6.3333333333333..7.6666666666667",

"7.6666666666667..9"

],

"h": [

[

3.5294117647058822,

1.411764705882353,

null,

1.5294117647058822,

null,

null

],

[

null,

null,

2.6470588235294117,

null,

null,

null

],

[

null,

null,

null,

2.5588235294117645,

null,

null

],

[

null,

null,

2.735294117647059,

null,

1.7647058823529411,

null

],

[

null,

null,

null,

2.2941176470588234,

2.852941176470588,

3.6470588235294117

],

[

0.9411764705882353,

null,

null,

1.1372549019607845,

null,

null

],

[

null,

null,

null,

2.2352941176470584,

null,

null

],

[

3.147058823529412,

null,

3.235294117647059,

null,

2.6470588235294117,

3.235294117647059

],

[

null,

null,

null,

null,

null,

1.6274509803921566

],

[

1.7058823529411764,

null,

null,

null,

null,

2.5294117647058822

],

[

1.5294117647058822,

null,

null,

2,

null,

null

],

[

2.1176470588235294,

null,

null,

null,

null,

null

]

],

"counts": [

[

1,

1,

0,

1,

0,

0

],

[

0,

0,

1,

0,

0,

0

],

[

0,

0,

0,

2,

0,

0

],

[

0,

0,

2,

0,

2,

0

],

[

0,

0,

0,

2,

2,

1

],

[

1,

0,

0,

3,

0,

0

],

[

0,

0,

0,

3,

0,

0

],

[

2,

0,

1,

0,

2,

1

],

[

0,

0,

0,

0,

0,

3

],

[

2,

0,

0,

0,

0,

1

],

[

2,

0,

0,

1,

0,

0

],

[

1,

0,

0,

0,

0,

0

]

]

}

„), true);

$h = $data['h'];

$ny = count($data['y\_labels']);

$nx = count($data['x\_labels']);

$grad = []; $maxg = 0;

for ($iy=0;$iy<$ny;$iy++){

$row = [];

for ($ix=0;$ix<$nx;$ix++){

$c = $h[$ix][$iy] ?? null;

if ($c===null){ $row[] = null; continue; }

$left = ($ix>0 && $h[$ix-1][$iy]!==null)?$h[$ix-1][$iy]:$c;

$right = ($ix<$nx-1 && $h[$ix+1][$iy]!==null)?$h[$ix+1][$iy]:$c;

$down = ($iy>0 && $h[$ix][$iy-1]!==null)?$h[$ix][$iy-1]:$c;

$up = ($iy<$ny-1 && $h[$ix][$iy+1]!==null)?$h[$ix][$iy+1]:$c;

$gx = ($right-$left)/2.0;

$gy = ($up-$down)/2.0;

$g = sqrt($gx\*$gx+$gy\*$gy);

$row[] = $g;

if ($g>$maxg) $maxg=$g;

}

$grad[] = $row;

}

$O = [];

for ($iy=0;$iy<$ny;$iy++){

$row = [];

for ($ix=0;$ix<$nx;$ix++){

if ($grad[$iy][$ix]===null){ $row[] = null; continue; }

$row[] = 1.0 - $grad[$iy][$ix]/$maxg;

}

$O[] = $row;

}

echo json\_encode(["meta"=>$data["meta"],"O"=>$O,"x\_labels"=>$data["x\_labels"],"y\_labels"=>$data["y\_labels"]], JSON\_PRETTY\_PRINT|JSON\_UNESCAPED\_UNICODE);

## 3. ****JSON für Operatoren-Zeitreihe I(t)****

Wenn du pro Unterrichtseinheit/Teilnehmer Zeitpunkte (t) hast, kannst du Operatoren (σ,S,D,M,R,E) aus den val\_\*-Werten berechnen. Beispiel: Mittelung pro Zeitfenster.

<?php

// operators\_time.php

header('Content-Type: application/json; charset=utf-8');

$pdo = new PDO("mysql:host=localhost;dbname=icas;charset=utf8mb4","root","");

// Annahme: Tabelle hat Spalte `zeitpunkt`

$sql = "SELECT zeitpunkt,

AVG(val\_mitarbeit) as sigma,

AVG(val\_materialien) as S,

AVG(val\_absprachen) as D,

AVG(val\_selbststaendigkeit) as M,

AVG(val\_transferdenken) as R,

AVG(val\_lernfortschritt) as E

FROM mtr\_rueckkopplung\_teilnehmer

GROUP BY zeitpunkt ORDER BY zeitpunkt";

$data = $pdo->query($sql)->fetchAll(PDO::FETCH\_ASSOC);

echo json\_encode(["I(t)"=>$data], JSON\_PRETTY\_PRINT|JSON\_UNESCAPED\_UNICODE);

✅ Mit diesen Skripten bekommst du JSON-Ausgaben für:

* **h(x,y)**: Dichtekarte,
* **O(x,y)**: Orientierungsverlust,
* **I(t)**: Operatorenverlauf.

Damit kannst du die in **Kapitel 6** beschriebenen Konzepte (σ-Zonen, M-Zonen, Dichtefunktion h, Zeitentwicklung σ(t)) direkt aus deinen DB-Daten abbilden.

Perfekt 👍 – ich baue dir drei eigenständige **Python-Skripte** (Heatmap, Gradient/O-Karte, Zeitreihe). Alle greifen direkt auf die JSON-Ausgabe deiner PHP-Skripte (density\_map.php, orientation\_loss.php, operators\_time.php).

Du kannst sie in einem **Jupyter Notebook** kombinieren oder als **.py-Dateien** einzeln laufen lassen.

# 1️⃣ Heatmap für h(x,y)h(x,y)h(x,y) ****semantische Dichtekarte****

# heatmap\_h.py

import requests

import numpy as np

import matplotlib.pyplot as plt

# --- JSON von deinem PHP-Endpunkt laden ---

url = "http://localhost/density\_map.php"

data = requests.get(url).json()

x\_labels = data["x\_labels"]

y\_labels = data["y\_labels"]

h = np.array(data["h"], dtype=float).T # Transponieren für korrekte Orientierung

# --- Plot ---

plt.figure(figsize=(10, 6))

plt.imshow(h, cmap="viridis", origin="lower", aspect="auto")

plt.colorbar(label="Semantische Dichte h(x,y)")

plt.xticks(ticks=np.arange(len(x\_labels)), labels=x\_labels, rotation=45)

plt.yticks(ticks=np.arange(len(y\_labels)), labels=y\_labels)

plt.title("Semantische Dichtekarte h(x,y)")

plt.xlabel("x-Bereich (kognitive Dimension)")

plt.ylabel("y-Bereich (soziale Dimension)")

plt.tight\_layout()

plt.show()

# 2️⃣ Gradientenkarten für O(x,y)O(x,y)O(x,y) ****Orientierungsverlust****

# orientation\_loss.py

import requests

import numpy as np

import matplotlib.pyplot as plt

url = "http://localhost/orientation\_loss.php"

data = requests.get(url).json()

x\_labels = data["x\_labels"]

y\_labels = data["y\_labels"]

O = np.array(data["O"], dtype=float).T

plt.figure(figsize=(10, 6))

plt.imshow(O, cmap="magma", origin="lower", aspect="auto", vmin=0, vmax=1)

plt.colorbar(label="Orientierungsverlust O(x,y)")

plt.xticks(ticks=np.arange(len(x\_labels)), labels=x\_labels, rotation=45)

plt.yticks(ticks=np.arange(len(y\_labels)), labels=y\_labels)

plt.title("Gradientenkarte – Orientierungsverlust O(x,y)")

plt.xlabel("x-Bereich (kognitive Dimension)")

plt.ylabel("y-Bereich (soziale Dimension)")

plt.tight\_layout()

plt.show()

# 3D

import requests

import numpy as np

import matplotlib.pyplot as plt

from mpl\_toolkits.mplot3d import Axes3D

# URL des PHP-Endpunkts

url = "http://localhost/easyDidak/library/php/Orientierungsverlust O(x,y)\_old.php"

# Hole die JSON-Daten von der PHP-Datei

data = requests.get(url).json()

# X- und Y-Achsen-Beschriftungen

x\_labels = data["x\_labels"]

y\_labels = data["y\_labels"]

O = np.array(data["O"], dtype=float) # O bleibt unverändert

# Erstelle ein Gitter für die X- und Y-Achse

x = np.arange(len(x\_labels))

y = np.arange(len(y\_labels))

# Erstelle ein 3D-Plot

fig = plt.figure(figsize=(12, 8))

ax = fig.add\_subplot(111, projection='3d')

# Erstelle das Meshgrid für X und Y (für die 3D-Fläche)

X, Y = np.meshgrid(x, y)

# Plot der 3D-Oberfläche

surf = ax.plot\_surface(X, Y, O, cmap="magma", edgecolor="none")

# Farbskala (Colorbar)

cbar = fig.colorbar(surf)

cbar.set\_label("Orientierungsverlust O(x,y)")

# Achsenbeschriftungen

ax.set\_xlabel('x-Bereich (kognitive Dimension)')

ax.set\_ylabel('y-Bereich (soziale Dimension)')

ax.set\_zlabel('Orientierungsverlust O(x,y)')

ax.set\_title("3D-Oberfläche der Orientierungsverluste")

# Setze die X- und Y-Ticks auf die Labels aus den JSON-Daten

ax.set\_xticks(x)

ax.set\_xticklabels(x\_labels, rotation=45, ha="right")

ax.set\_yticks(y)

ax.set\_yticklabels(y\_labels)

# Layout anpassen

plt.tight\_layout()

# Zeige das Plot

plt.show()

# 3️⃣ Zeitreihe für I(t)I(t)I(t) ****Operatoren-Zeitreihe I(t)****

# operators\_time.py

import requests

import matplotlib.pyplot as plt

import pandas as pd

url = "http://localhost/operators\_time.php"

data = requests.get(url).json()["I(t)"]

df = pd.DataFrame(data)

df["zeitpunkt"] = pd.to\_datetime(df["zeitpunkt"])

# --- Plot ---

plt.figure(figsize=(10,6))

for col in df.columns:

if col != "zeitpunkt":

plt.plot(df["zeitpunkt"], df[col], marker="o", label=col)

plt.legend()

plt.title("Zeitreihe der Operatoren I(t)")

plt.xlabel("Zeitpunkt")

plt.ylabel("Bewertung / Mittelwert")

plt.xticks(rotation=45)

plt.tight\_layout()

plt.show()

# 📊 Hinweise

* **Input**:
  + density\_map.php → liefert h(x,y)h(x,y)h(x,y)
  + orientation\_loss.php → liefert O(x,y)O(x,y)O(x,y)
  + operators\_time.php → liefert Zeitreihe I(t)I(t)I(t)
* **Anpassung**:  
  Falls deine Endpunkte nicht auf localhost laufen, ersetze die url = "http://localhost/...php"-Zeile durch die richtige Adresse.
* **Plotly-Alternative**:  
  Wenn du interaktive Plots willst, kannst du matplotlib durch plotly.express.imshow() (für Heatmaps) und plotly.express.line() (für Zeitreihen) ersetzen.

erkläre mir, wie ich plotly.express.imshow() (für Heatmaps) und plotly.express.line() (für Zeitreihen) in den skripten benutze

###### ChatGPT:

Sehr gute Entscheidung 👍 – mit **Plotly** werden deine Grafiken interaktiv (Zoom, Hover, Export). Ich zeige dir, wie du die bestehenden Matplotlib-Plots durch **Plotly Express** ersetzt.

# 📌 1. Heatmap mit plotly.express.imshow()

# heatmap\_h\_plotly.py

import requests

import numpy as np

import plotly.express as px

url = "http://localhost/density\_map.php"

data = requests.get(url).json()

x\_labels = data["x\_labels"]

y\_labels = data["y\_labels"]

h = np.array(data["h"], dtype=float).T # Transponieren, damit x/y stimmt

fig = px.imshow(

h,

labels=dict(x="x-Bereich (kognitive Dimension)", y="y-Bereich (soziale Dimension)", color="Semantische Dichte h(x,y)"),

x=x\_labels,

y=y\_labels,

color\_continuous\_scale="Viridis",

aspect="auto"

)

fig.update\_layout(title="Interaktive Heatmap h(x,y)")

fig.show()

👉 Vorteile:

* Hover zeigt dir den Wert der Zelle.
* Mit der Toolbar kannst du zoomen, schwenken, als PNG exportieren.

# 📌 2. Gradientenkarten (O(x,y)) mit imshow

# orientation\_loss\_plotly.py

import requests

import numpy as np

import plotly.express as px

url = "http://localhost/orientation\_loss.php"

data = requests.get(url).json()

x\_labels = data["x\_labels"]

y\_labels = data["y\_labels"]

O = np.array(data["O"], dtype=float).T

fig = px.imshow(

O,

labels=dict(x="x-Bereich (kognitive Dimension)", y="y-Bereich (soziale Dimension)", color="Orientierungsverlust O(x,y)"),

x=x\_labels,

y=y\_labels,

color\_continuous\_scale="Magma",

aspect="auto",

zmin=0, zmax=1 # Skala 0–1

)

fig.update\_layout(title="Interaktive Gradientenkarte O(x,y)")

fig.show()

# 📌 3. Zeitreihe mit plotly.express.line()

# operators\_time\_plotly.py

import requests

import pandas as pd

import plotly.express as px

url = "http://localhost/operators\_time.php"

data = requests.get(url).json()["I(t)"]

df = pd.DataFrame(data)

df["zeitpunkt"] = pd.to\_datetime(df["zeitpunkt"])

# Mehrere Spalten gleichzeitig plotten

fig = px.line(

df,

x="zeitpunkt",

y=[col for col in df.columns if col != "zeitpunkt"],

labels={"value": "Bewertung / Mittelwert", "variable": "Operator"},

markers=True

)

fig.update\_layout(title="Interaktive Zeitreihe der Operatoren I(t)")

fig.show()

👉 Vorteile:

* Hover zeigt genaue Werte pro Operator und Zeitpunkt.
* Zoom auf Zeitabschnitte möglich.
* Export-Funktion integriert.

# 📊 Zusammenfassung

* **Matplotlib** = statische Abbildung (gut für Druck).
* **Plotly** = interaktive Abbildung (gut für Exploration, Präsentation).
* Du kannst beide parallel nutzen:
  + **Matplotlib** → für deine Dissertation (statische Bilder einfügen).
  + **Plotly** → für deine Analyse und Verständnis.