

High-Quality Map of Rivers and Observation Wells in Kushtia

✓ Step 1: Install and import required libraries

```
!pip install geopandas matplotlib contextily shapely
!pip install matplotlib-scalebar
```

```
import geopandas as gpd
import matplotlib.pyplot as plt
from shapely.geometry import Point
from matplotlib_scalebar.scalebar import ScaleBar
import zipfile, os
```

```
Requirement already satisfied: geopandas in /usr/local/lib/python3.12/dist-packa
Requirement already satisfied: matplotlib in /usr/local/lib/python3.12/dist-pack
Collecting contextily
```

```
  Downloading contextily-1.6.2-py3-none-any.whl.metadata (2.9 kB)
```

```
Requirement already satisfied: shapely in /usr/local/lib/python3.12/dist-package
Requirement already satisfied: numpy>=1.24 in /usr/local/lib/python3.12/dist-pac
Requirement already satisfied: pyogrio>=0.7.2 in /usr/local/lib/python3.12/dist-
Requirement already satisfied: packaging in /usr/local/lib/python3.12/dist-packa
Requirement already satisfied: pandas>=2.0.0 in /usr/local/lib/python3.12/dist-p
Requirement already satisfied: pyproj>=3.5.0 in /usr/local/lib/python3.12/dist-p
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.12/dis
Requirement already satisfied: cycycler>=0.10 in /usr/local/lib/python3.12/dist-pa
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.12/di
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.12/di
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.12/dist-packa
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.12/dis
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.12
Requirement already satisfied: geopy in /usr/local/lib/python3.12/dist-packages
Collecting mercantile (from contextily)
```

```
  Downloading mercantile-1.2.1-py3-none-any.whl.metadata (4.8 kB)
```

```
Collecting rasterio (from contextily)
```

```
  Downloading rasterio-1.4.3-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86
```

```
Requirement already satisfied: requests in /usr/local/lib/python3.12/dist-packag
Requirement already satisfied: joblib in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: xyzservices in /usr/local/lib/python3.12/dist-pac
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-pa
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-
Requirement already satisfied: certifi in /usr/local/lib/python3.12/dist-package
Requirement already satisfied: sip>=1.5 in /usr/local/lib/python3.12/dist-packag
Requirement already satisfied: geographiclib<3,>=1.52 in /usr/local/lib/python3.
Requirement already satisfied: click>=3.0 in /usr/local/lib/python3.12/dist-pack
Collecting affine (from rasterio->contextily)
```

```
  Downloading affine-2.4.0-py3-none-any.whl.metadata (4.0 kB)
```

```

downloading affine-2.4.0-py3-none-any.whl.metadata (4.0 kB)
Requirement already satisfied: attrs in /usr/local/lib/python3.12/dist-packages
Collecting cligj>=0.5 (from rasterio->contextily)
  Downloading cligj-0.7.2-py3-none-any.whl.metadata (5.0 kB)
Collecting click-plugins (from rasterio->contextily)
  Downloading click_plugins-1.1.1.2-py2.py3-none-any.whl.metadata (6.5 kB)
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-pa
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/d
Downloading contextily-1.6.2-py3-none-any.whl (17 kB)
Downloading mercantile-1.2.1-py3-none-any.whl (14 kB)
Downloading rasterio-1.4.3-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_6
22.3/22.3 MB 111.0 MB/s eta 0:00:00
Downloading cligj-0.7.2-py3-none-any.whl (7.1 kB)
Downloading affine-2.4.0-py3-none-any.whl (15 kB)
Downloading click_plugins-1.1.1.2-py2.py3-none-any.whl (11 kB)
Installing collected packages: mercantile, cligj, click-plugins, affine, rasteri
Successfully installed affine-2.4.0 click-plugins-1.1.1.2 cligj-0.7.2 contextily
Collecting matplotlib-scalebar
  Downloading matplotlib_scalebar-0.9.0-py3-none-any.whl.metadata (18 kB)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.12/dist-pack
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.12/dis
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.12/dist-pa
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.12/di
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.12/di
Requirement already satisfied: numpy>=1.23 in /usr/local/lib/python3.12/dist-pac

```

✓ Step 2: Mount Google Drive

```

from google.colab import drive
drive.mount('/content/drive')

```

Mounted at /content/drive

✓ Step 3: Unzip and load data

```

data_path = "/content/drive/My Drive/Data For Analysis"

# --- District shapefile unzip ---
with zipfile.ZipFile(os.path.join(data_path, "gadm41_BGD_2.json.zip"), 'r') as
    zip_ref.extractall("/content/districts")

districts = gpd.read_file("/content/districts/gadm41_BGD_2.json")

# --- Rivers shapefile unzip ---
with zipfile.ZipFile(os.path.join(data_path, "bgd_hyd_rivers_lged.zip"), 'r') a
    zip_ref.extractall("/content/rivers")

```

```

rivers = gpd.read_file("/content/rivers/bgd_hyd_rivers_lged.shp")

# --- Wells CSV load ---
import pandas as pd
wells_df = pd.read_csv(os.path.join(data_path, "kushtia_wells.csv"))
wells_gdf = gpd.GeoDataFrame(wells_df, geometry=gpd.points_from_xy(wells_df.Lon

```

✓ Step 4: Filter only Kushtia district and rivers

```

# Kushtia district polygon
kushtia = districts[districts['NAME_2'] == 'Kushtia']

# Clip rivers to Kushtia boundary
rivers_kushtia = gpd.clip(rivers, kushtia)

# Clip wells inside Kushtia
wells_kushtia = gpd.clip(wells_gdf, kushtia)

```

✓ Step 5: Draw the map (Auto Save in Drive, TIFF, SVG, PDF, EPS)

```

import geopandas as gpd
import matplotlib.pyplot as plt
from shapely.geometry import Point
from matplotlib_scalebar.scalebar import ScaleBar
import zipfile, os
import pandas as pd
from google.colab import drive

# ----- Mount Google Drive -----
drive.mount('/content/drive')

# Change this path to "Data For Analysis" folder
save_path = "/content/drive/My Drive/Data For Analysis"
os.makedirs(save_path, exist_ok=True)

# ----- CRS: Project to UTM (EPSG:32646) -----
kushtia_utm = kushtia.to_crs(epsg=32646)
rivers_kushtia_utm = rivers_kushtia.to_crs(epsg=32646)
wells_kushtia_utm = wells_kushtia.to_crs(epsg=32646)

# Font setup
plt.rcParams["font.family"] = "Times New Roman"

```

```

fig, ax = plt.subplots(figsize=(8, 8))

# ----- Plot Layers -----
kushtia_utm.boundary.plot(ax=ax, color="black", linewidth=1.2, label="District
rivers_kushtia_utm.plot(ax=ax, color="#1f78b4", linewidth=1.0, label="Major Riv
wells_kushtia_utm.plot(ax=ax, color="#e31a1c", markersize=35, marker="o", label

# ----- Legend -----
legend = ax.legend(
    loc="upper right",
    bbox_to_anchor=(0.98, 0.85),
    frameon=True,
    fontsize=9,
    title="Map Elements",
    title_fontsize=10
)
legend.get_frame().set_edgecolor("black")
legend.get_frame().set_alpha(0.8)

# ----- Scale bar -----
scalebar = ScaleBar(
    dx=1,
    units="m",
    dimension="si-length",
    location="lower left",
    scale_loc="bottom",
    box_alpha=0.9,
    frameon=True
)
ax.add_artist(scalebar)

# ----- North arrow -----
x, y, arrow_length = 0.95, 0.98, 0.08
ax.annotate(
    'N', xy=(x, y), xytext=(x, y - arrow_length),
    arrowprops=dict(facecolor='black', width=4, headwidth=12),
    ha='center', va='center', fontsize=12, xycoords=ax.transAxes
)

# Clean axis
ax.set_axis_off()

# ----- Save in multiple formats -----
raster_path = os.path.join(save_path, "kushtia_map_publication.tif")
vector_svg = os.path.join(save_path, "kushtia_map_publication.svg")
vector_pdf = os.path.join(save_path, "kushtia_map_publication.pdf")
vector_eps = os.path.join(save_path, "kushtia_map_publication.eps")

plt.savefig(raster_path, dpi=600, bbox_inches="tight") # High-resolution rast
plt.savefig(vector_svg, dpi=600, bbox_inches="tight") # Vector (SVG)

```

```
plt.savefig(vector_pdf, dpi=600, bbox_inches="tight") # Vector (PDF)
plt.savefig(vector_eps, dpi=600, bbox_inches="tight") # Vector (EPS)

plt.show()

print("✅ Maps saved in:", save_path)
```

[illegible]

[illegible]

[illegible]

LIADNTMC:metalep1ib font manager:findfont: Font family: 'Times New Roman' not found

[illegible]

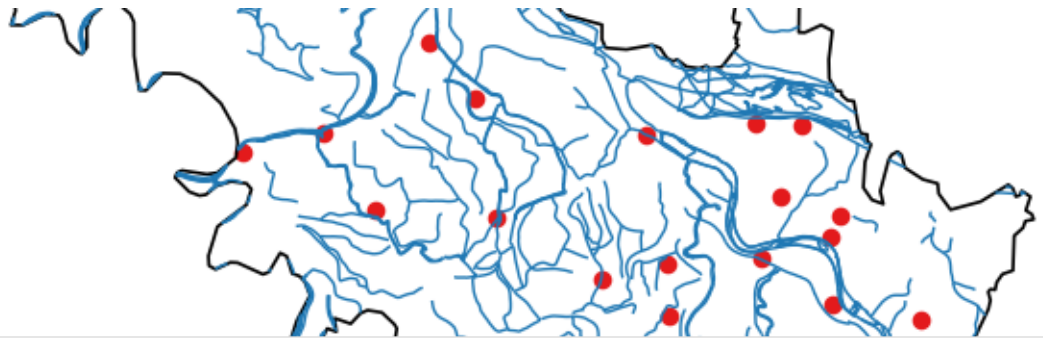
[illegible]

Learning	Goal 1	Goal 2	Goal 3	Goal 4	Goal 5	Goal 6	Goal 7	Goal 8	Goal 9	Goal 10	Goal 11	Goal 12	Goal 13	Goal 14	Goal 15	Goal 16	Goal 17	Goal 18	Goal 19	Goal 20	Goal 21	Goal 22	Goal 23	Goal 24	Goal 25	Goal 26	Goal 27	Goal 28	Goal 29	Goal 30	Goal 31	Goal 32	Goal 33	Goal 34	Goal 35	Goal 36	Goal 37	Goal 38	Goal 39	Goal 40	Goal 41	Goal 42	Goal 43	Goal 44	Goal 45	Goal 46	Goal 47	Goal 48	Goal 49	Goal 50	Goal 51	Goal 52	Goal 53	Goal 54	Goal 55	Goal 56	Goal 57	Goal 58	Goal 59	Goal 60	Goal 61	Goal 62	Goal 63	Goal 64	Goal 65	Goal 66	Goal 67	Goal 68	Goal 69	Goal 70	Goal 71	Goal 72	Goal 73	Goal 74	Goal 75	Goal 76	Goal 77	Goal 78	Goal 79	Goal 80	Goal 81	Goal 82	Goal 83	Goal 84	Goal 85	Goal 86	Goal 87	Goal 88	Goal 89	Goal 90	Goal 91	Goal 92	Goal 93	Goal 94	Goal 95	Goal 96	Goal 97	Goal 98	Goal 99	Goal 100
Learning	Goal 1	Goal 2	Goal 3	Goal 4	Goal 5	Goal 6	Goal 7	Goal 8	Goal 9	Goal 10	Goal 11	Goal 12	Goal 13	Goal 14	Goal 15	Goal 16	Goal 17	Goal 18	Goal 19	Goal 20	Goal 21	Goal 22	Goal 23	Goal 24	Goal 25	Goal 26	Goal 27	Goal 28	Goal 29	Goal 30	Goal 31	Goal 32	Goal 33	Goal 34	Goal 35	Goal 36	Goal 37	Goal 38	Goal 39	Goal 40	Goal 41	Goal 42	Goal 43	Goal 44	Goal 45	Goal 46	Goal 47	Goal 48	Goal 49	Goal 50	Goal 51	Goal 52	Goal 53	Goal 54	Goal 55	Goal 56	Goal 57	Goal 58	Goal 59	Goal 60	Goal 61	Goal 62	Goal 63	Goal 64	Goal 65	Goal 66	Goal 67	Goal 68	Goal 69	Goal 70	Goal 71	Goal 72	Goal 73	Goal 74	Goal 75	Goal 76	Goal 77	Goal 78	Goal 79	Goal 80	Goal 81	Goal 82	Goal 83	Goal 84	Goal 85	Goal 86	Goal 87	Goal 88	Goal 89	Goal 90	Goal 91	Goal 92	Goal 93	Goal 94	Goal 95	Goal 96	Goal 97	Goal 98	Goal 99	Goal 100

WARNING:matplotlib.font_manager.findfont: Font family 'Times New Roman' not found

Map Elements

- District Boundary
- Major Rivers
- Observation Wells



Start coding or [generate](#) with AI.