À propos

Sommaire

## Quarto

|  |  |
| --- | --- |
|  | **Ce site est fait avec *Quarto***  <https://quarto.org/docs/websites>. |

## Exemples de code python (jupyter)

|  |  |  |
| --- | --- | --- |
|  | **L’exemple du site Quarto**  For a demonstration of a line plot on a polar axis, see [Figure 1](#fig-polar).  import numpy as np import matplotlib.pyplot as plt  r = np.arange(0, 2, 0.01) theta = 2 \* np.pi \* r fig, ax = plt.subplots(  subplot\_kw = {'projection': 'polar'}  ) ax.plot(theta, r) ax.set\_rticks([0.5, 1, 1.5, 2]) ax.grid(True) plt.show()   |  | | --- | | Figure 1: A line plot on a polar axis | |

## Exemple de code OJS (observable)

|  |  |
| --- | --- |
|  | **Exemple de l’utilisation de la bibliothèque bertin.js**  *Manhamady OUEDRAOGO (Burkina Faso) & Nicolas LAMBERT (France)* <https://ee-cist.github.io/CAR2_cartodyn/TP2/docs/index.html>  //| panel: sidebar viewof year = Inputs.range(  [1990, 2019],   {value: 2019, step: 1, label: "Année"} ) viewof k = Inputs.range(  [20, 100],   {value: 50, step: 1, label: "Rayon max"} ) meta = FileAttachment("data/worldbank\_meta.csv").csv() viewof indicator = Inputs.select(  new Map(meta.map((d) => [d.indicator, d.shortcode])),  { label: "Indicateur" } ) projections = ["Patterson", "NaturalEarth1", "Bertin1953", "InterruptedSinusoidal", "Armadillo", "Baker", "Gingery", "Berghaus", "Loximuthal", "Healpix", "InterruptedMollweideHemispheres", "Miller", "Aitoff", "ConicEqualArea", "Eckert3", "Hill"] viewof proj = Inputs.select(projections, {label: "Projection", width: 350}) viewof color = Inputs.color({label: "couleur", value: "#4682b4"}) viewof simpl = Inputs.range( [0.01, 0.5], {value: 0.1, step: 0.01, label: "Simplification"} ) viewof x = Inputs.range( [-180, 180], {value: 0, step: 1, label: "Rotation (x)"} ) viewof y = Inputs.range( [-90, 90], {value: 0, step: 1, label: "Rotation (y)"} ) Carte bertin.draw({ params: {projection: proj + `.rotate([${x}, ${y}])`, clip: true },  layers:[  { type : "header", text: title},  {type: "bubble", geojson: data, values: indicator,   fill: color, fixmax: varmax, k,   tooltip: ["$name",d => d.properties[indicator]]},  {geojson: world2, fill: "#CCC"},  {type: "graticule"},   {type: "outline"} ]}) Données Inputs.table(statsyear, { columns: [  "country",  "capital\_city",  "region",  indicator  ]}) Top 10 viewof topnb = Inputs.range([5, 30], {label: "Nombre de pays représentés", step: 1}) top = statsyear.sort((a, b) => d3.descending(+a[indicator], +b[indicator]))  .slice(0, topnb) Plot.plot({  marginLeft: 60,  grid: true,  x: {  //type: "log",  label: "Années →"  },  y: {  label: "↑ Population",  //type: "log",  },  marks: [  Plot.barY(top, {  x: "iso3c",  y: indicator,  sort: { x: "y", reverse: true },  fill: color  }),  Plot.ruleY([0])  ] })  world = FileAttachment("data/world.json").json() stats = FileAttachment("data/worldbank\_data.csv").csv() geo = require("geotoolbox@latest") world2 = geo.simplify(world, {k: simpl}) bertin = require("bertin@latest") statsyear = stats.filter(d => d.date == year) data = bertin.merge(world2, "id", statsyear, "iso3c") varmax = d3.max(stats.filter(d => d.date == 2019), d => +d[indicator]) title = meta.map((d) => [d.indicator, d.shortcode]).find((d) => d[1] == indicator)[0] + " in " + year |