econ-viz 软件包：微观经济学图形

Pin Yue Sung[*†] Ling Tak Douglas Chung[*]

2026-09-26 · v1.12.0 · ·

目录

[*]台湾政治大学国际经营与贸易学系，台北，台湾。
[†]通讯作者。电子邮箱：contact@econ-viz.org

第 1 节　简介

econ-viz 是微观经济学绘图软件包，涵盖效用模型、最优消费组合求解与可视化。Python API 与命令行工具均可生成无差异曲线、预算线、消费者均衡、需求曲线及 Edgeworth 箱形图。

图形默认显示第一象限，坐标轴以箭头表示并隐藏数字刻度，标签支持 TeX 数学语法。输出格式包括 PNG、PDF、SVG、TikZ 与 GIF。

1.1　功能范围

econ-viz 的功能分为效用模型、均衡求解、效用水平、图层与导出格式五个部分。模型与求解结果可分开使用；同一模型也可交由画布、多面板图、需求图或分析 API 处理。

1.2　阅读指引

本手册按主题分成四个部分（参见表 1）：

表 1
章节主题

主题	子主题	说明	章节
绘图与导出	画布与图层	坐标轴、曲线与均衡点	第 4 节
	多面板图与需求图	并排比较与需求曲线	第 5 节
	主题与配置文件	配色、样式与配置文件	第 9 节
	输出格式	导出图像与 TikZ	第 10 节
模型与分析	内置模型	常见效用函数	第 6 节
	自定义与多商品模型	自定义函数与多商品	第 7 节
	比较静态与 Slutsky 矩阵	需求导数与齐次性	第 8 节
动画与交互	GIF 动画	参数变动的 GIF	第 12 节
	Jupyter 交互组件	笔记本滑块调参数	第 13 节
其他输入方式	命令行工具	使用命令行出图	第 11 节
	LaTeX 效用函数解析	从算式创建模型	第 14 节

初次使用时，依次阅读第 3 节与第 4 节。查询特定 API 时，可直接前往对应章节或命令索引。

[1] Python 官方网站提供各操作系统的安装程序：`https://www.python.org/downloads/`。

[2] uv 是 Astral 开发的 Python 软件包与项目管理工具，速度快且可一并管理 Python 版本；安装方式与完整说明见官方文档：`https://docs.astral.sh/uv/`。

第 2 节　安装

2.1　系统需求

econ-viz 需要 Python 3.10 以上版本[1]。本章以 uv[2] 管理软件包与项目，并附上对应的 pip 命令。

安装软件包时会一并安装 NumPy、SciPy、matplotlib 与 SymPy，分别用于数组运算、数值求解、绘图与符号运算[3]。

2.2　安装 uv

本手册的命令以 uv 为准。现有项目仍可使用 pip、pipx 或 Poetry；软件包 API 不受管理工具影响。

依操作系统运行下列安装命令。

macOS、Linux

```
curl -LsSf https://astral.sh/uv/install.sh | sh
```

Windows

```
powershell -ExecutionPolicy ByPass -c "irm https://astral.sh/uv/
install.ps1 | iex"
```

2.3　安装 econ-viz

创建项目并添加 econ-viz：

```
uv init my-diagrams
cd my-diagrams
uv add econ-viz
```

使用 `uv run` 在项目环境中运行程序：

```
uv run python main.py
```

将第 3 节的基本示例存为项目目录下的 `main.py`，再运行上述命令。`uv add` 记录项目依赖，`uv run` 使用该项目的 Python 环境。安装与运行必须使用同一个环境。

若要固定本手册使用的版本，可在添加依赖时指定版本号：

```
uv add "econ-viz==1.12.0"
```

若使用现有的 Python 虚拟环境，可通过 pip 安装：

```
python -m pip install -U econ-viz
```

软件包名称是 `econ-viz`，Python 导入名称是 `econ_viz`：

```
from econ_viz import Canvas, solve
```

`import` 语句不得使用连字符。若发生 `ModuleNotFoundError`，检查运行程序的解释器是否与安装软件包时使用的环境相同。

2.4　可选依赖

动画与 Jupyter 交互组件需要可选依赖。按用途安装其中一组：

[3] 一般 Python 绘图不需要另外安装 LaTeX；只有要编译导出的 TikZ 源代码时，才需要 LaTeX 环境。

`animation` 供 `Animator` 写入 GIF（详见第 12 节）；安装 Pillow。
`interactive` 供 `WidgetViewer` 在 Jupyter 显示控件（详见第 13 节）；安装 ipywidgets 与 IPython。
`all` 全部可选依赖。

```
uv add "econ-viz[animation]"    # GIF 导出（Pillow）
uv add "econ-viz[interactive]"  # 笔记本交互组件
uv add "econ-viz[all]"          # 全部可选依赖
```

安装可选依赖不会自动运行动画或打开笔记本，仍需按各章示例调用对应的 API。

2.5　安装命令行工具

仅使用命令行接口时，可将 econ-viz 安装为独立工具（详见第 11 节）：

```
uv tool install econ-viz
```

此方式将命令行工具安装在独立环境。需要在 Python 程序中导入软件包时，仍须在该项目运行 `uv add econ-viz`。在项目内以 `uv run econ-viz` 调用工具，可让命令行与 Python 程序使用同一版本。

2.6　开发环境设置

```
git clone https://github.com/EconViz/econ-viz.git
cd econ-viz
uv sync --all-extras
```

其中 `uv sync --all-extras` 会安装开发依赖与所有可选依赖。完成后运行测试：

```
uv run pytest
```

2.7　验证安装

```
uv run econ-viz --version   # econ-viz 1.12.0
uv run econ-viz help
```

注释中的版本号仅为示例，实际输出取决于安装版本。也可用以下命令确认 Python 能导入绘图与求解接口：

```
uv run python -c "from econ_viz import Canvas, solve; print('OK')"
```

在服务器或其他没有图形界面的环境中，请以 `save()` 或命令行的 `--output` 输出文件。`show()` 需要可用的交互式绘图后端，窗口打不开不代表安装失败。

第 3 节　快速开始

3.1　基本示例

本章以 Cobb-Douglas 消费者问题说明完整流程：

$$\max_{x,y} u(x,y) = x^{\frac{1}{2}} y^{\frac{1}{2}}$$

$$\text{s.t. } 2x + 3y = 30$$

程序先求解最优消费组合，再绘制无差异曲线、预算集与均衡点（参见图 1）。

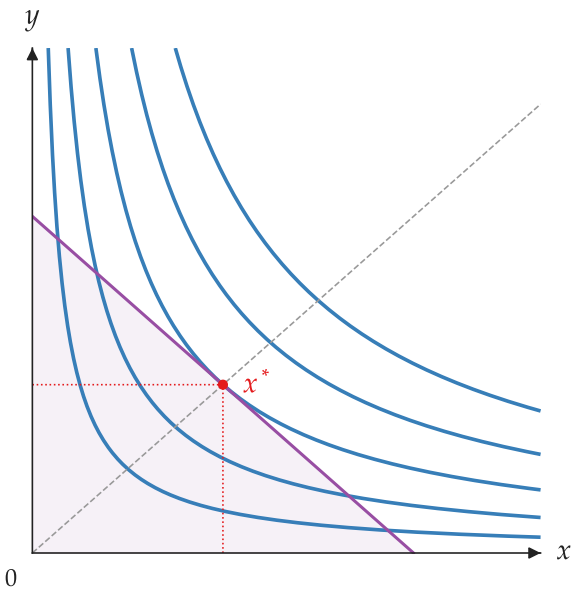

图 1
消费者均衡。

例 1

```python
from econ_viz import Canvas, levels, solve
from econ_viz.models import CobbDouglas

model = CobbDouglas(alpha=0.5, beta=0.5)
eq    = solve(model, px=2.0, py=3.0, income=30.0)
lvls  = levels.around(eq.utility, n=5)

cvs = Canvas(
    x_max=20, y_max=15,
    x_label="x", y_label="y",
    title=r"Cobb-Douglas $x^{0.5} y^{0.5}$"
)
cvs.add_utility(model, levels=lvls)
cvs.add_budget(2.0, 3.0, 30.0, fill=True)
cvs.add_equilibrium(eq, show_ray=True)
cvs.save("cobb_douglas.png")
```

运行主程序 `main.py` 后，会在当前工作目录创建 `cobb_douglas.png`。本例的最优消费量为 $x^* = 7.5$、$y^* = 5$，支出为 $2 \times 7.5 + 3 \times 5 = 30$，恰好等于收入。预算线的两个轴截距则为 $\frac{I}{p_x} = 15$ 与 $\frac{I}{p_y} = 10$。

3.2　逐步说明

绘图流程依次为：创建模型、求解均衡、选取效用水平、创建画布、添加图层与导出。

`econ_viz.models`

```python
from econ_viz.models import CobbDouglas
model = CobbDouglas(alpha=0.5, beta=0.5)
```

创建效用模型。内置模型详见第 6 节；自定义函数使用 `CustomUtility`，详见第 7 节。

`solve`　`solve(⟨模型⟩, px=⟨浮点数⟩, py=⟨浮点数⟩, income=⟨浮点数⟩)`

根据价格与收入求解，返回不可变的 `Equilibrium` 对象。

`x`　最优消费量 x^*。

float

`y`　最优消费量 y^*。

float

`utility`　模型在最优点的效用值。

float

`bundle_type`　解的类型：`"interior"` 为内部解、`"boundary"` 为数值解位于数量下界（例如 Stone-Geary 的最低消费限制）、`"kink"` 为完全互补的折点解、`"corner"` 为完全替代的角解。

str

```
from econ_viz import solve
eq = solve(model, px=2.0, py=3.0, income=30.0)
print(eq.x, eq.y, round(eq.utility, 3))

# 7.5 5.0 6.124
```

`levels.around`　`levels.around(⟨效用值⟩, n=⟨整数⟩)`

以指定效用值为中心，生成 n 个效用水平。图 1 使用 `n=5`，因此绘制五条无差异曲线。

`levels.around()` 会保留指定的效用值，均衡点会落在其中一条无差异曲线上。若直接传入 `levels=5`，画布会依采样到的效用范围选取水平，不保证包含均衡效用。要比较多张图上的同一组偏好，可先计算一次 `lvls`，再让各张图使用相同水平。

`Canvas`　`Canvas(x_max=⟨浮点数⟩, y_max=⟨浮点数⟩, ...)`

创建画布并设置坐标范围，其余参数详见第 4 节。

坐标范围只控制显示区域，不参与 `solve()` 的计算。均衡点未出现在图中时，检查 `eq.x` 与 `eq.y` 是否超出范围，再调整坐标上限。

`Canvas.add_*`　在同一张画布上添加无差异曲线、预算线或均衡点。这些方法可以链式调用（详见第 4.4 节）。`add_equilibrium()` 使用现有的求解结果，不会重新求解；若之后修改价格、收入或模型，需重新调用 `solve()`，并创建对应的新图。

`Canvas.save`　`save()` 根据扩展名导出文件（详见第 10 节）；`show()` 打开 matplotlib 交互窗口。输出

`Canvas.show`　路径以运行程序时的工作目录为准，若需指定子目录，请事先创建目录。

检查输入与求解失败

价格与收入必须为正；输入不符合条件时，`solve()` 抛出 `InvalidParameterError`[4]。

求解未收敛时，`solve()` 抛出 `OptimizationError`，不会返回无效的均衡点。

```
from econ_viz import InvalidParameterError, OptimizationError, solve

try:
    eq = solve(model, px=2.0, py=3.0, income=30.0)
except (InvalidParameterError, OptimizationError) as error:
    print(error)
else:
    print(round(eq.x, 3), round(eq.y, 3), eq.bundle_type)
```

数值解可能有微小误差，例如结果接近但不恰好等于 `7.5`。验证时应使用适当容差；显示结果时再用 `round()` 格式化，避免过早四舍五入影响后续计算。

第 4 节　Canvas 画布

`Canvas` 以 matplotlib 的 `Figure` 与 `Axes` 为基础，提供无差异曲线、预算线与均衡点等图层。

[4] 一般平滑模型使用序列最小二乘法（sequential least squares programming, SLSQP）求解。

4.1　构造函数

Canvas

```
from econ_viz import ArrowStyle, Canvas, Stroke, themes

cvs = Canvas(
    x_max=20,
    y_max=15,
    x_label="x",
    y_label="y",
    title=r"Cobb-Douglas $x^{0.5} y^{0.5}$",
    dpi=300,
    x_label_pos="right",    # "top"、"right" 或 "bottom"
    y_label_pos="top",      # "left"、"top" 或 "right"
    font="DejaVu Sans",
    math_font="stix",
    axis_stroke=Stroke(width=1.0, arrow=ArrowStyle.TRIANGLE),
    theme=themes.default,
)
```

更新：*v1.7.0*

创建画布，并设置坐标范围、轴标签、字体、线条样式与主题。未明确传入的视觉设置取自当前的 `Config` 与 `Theme`。

`x_max`

float · 默认 `10`

横轴上限。

`y_max`

float · 默认 `10`

纵轴上限。

`x_label`

str · 默认 `"X"`

横轴末端的标签。

`y_label`

str · 默认 `"Y"`

纵轴末端的标签。

`title`

str | *None* · 默认 `None`

图形标题。

`dpi`

int · 默认 `300`

位图导出分辨率，限制在 1–1200 之间。

`x_label_pos`

str · 默认 `"right"`

横轴标签相对箭头的位置，也接受 `LabelPosition`（参见表 2）。

表 2　`x_label_pos` 值

值	位置
`"top"`	箭头上方
`"right"`	箭头右侧
`"bottom"`	箭头下方

`y_label_pos`

str · 默认 `"top"`

纵轴标签相对箭头的位置（参见表 3）。

表 3　`y_label_pos` 值

值	位置
`"left"`	箭头左侧
`"top"`	箭头上方
`"right"`	箭头右侧

`font`
str | list · 默认 `None`
所有文字使用的字体，或候选字体列表。此项设置仅作用于指定画布，不会改变 matplotlib 的全局设置。

`math_font`
str · 默认 `None`
matplotlib 的数学字体：`dejavusans`、`dejavuserif`、`cm`、`stix` 或 `stixsans` 等。

`axis_stroke`
Stroke · 默认 `theme`
同时设置两轴的粗细、线条样式、颜色与箭头（详见第 4.3 节）。

`x_axis_stroke`
Stroke · 默认 `None`
单独覆盖横轴。

`y_axis_stroke`
Stroke · 默认 `None`
单独覆盖纵轴。

`theme`
Theme · 默认 `themes.default`
配色与样式主题（详见第 9 节）。

4.2　绘图方法

`add_*` 方法在当前的画布添加图层，并返回同一个 `Canvas`，因此可链式调用（详见第 4.4 节）。

`add_utility`

```
cvs.add_utility(
    func,
    levels=3,           # 曲线条数或效用值列表
    color=None,         # 默认 theme.ic_color
    linewidth=None,     # 默认 theme.ic_linewidth
    show_rays=False,
    show_kinks=False,
    kink_radius=1.0,
    show_bliss=True,    # 标出饱和点（Satiation）
    show_ic_labels=False,
    stroke=None,
    ray_stroke=None,
    ic_label=None,
    highlight_level=None,
    secondary_stroke=None,
    label_style="numeric",
)
```

绘制无差异曲线。

`levels`
int | list · 默认 `3`
曲线条数或效用值列表；例如 `levels.around(eq.utility, n=5)` 会在均衡效用附近选取五个水平。

`highlight_level`
float | None · 默认 `None`
焦点效用值。绘制时选择最接近的曲线，以主要样式显示。

`secondary_stroke`
Stroke | None · 默认 `None`
其他曲线的样式；省略时使用 `theme.secondary_ic_stroke`。

`show_ic_labels`
bool · 默认 `False`
显示曲线标签。标签沿曲线切线旋转，并保持在可见绘图区内（参见图 3）。

`label_style`
str · 默认 `"numeric"`
`"numeric"` 显示效用值；`"ordinal"` 按效用递增顺序显示 $u_1, u_2, \ldots$。

```
lvls = levels.around(eq.utility, n=5)
cvs.add_utility(
    model,
    levels=lvls,
```

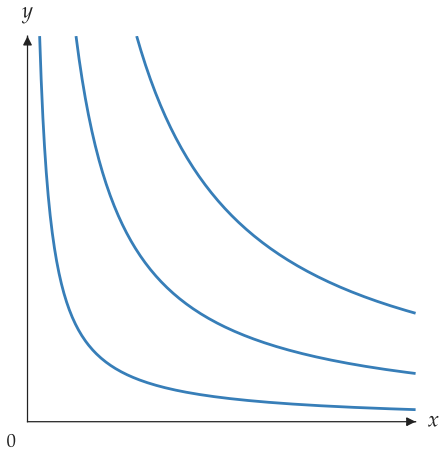

图 2
三条无差异曲线。

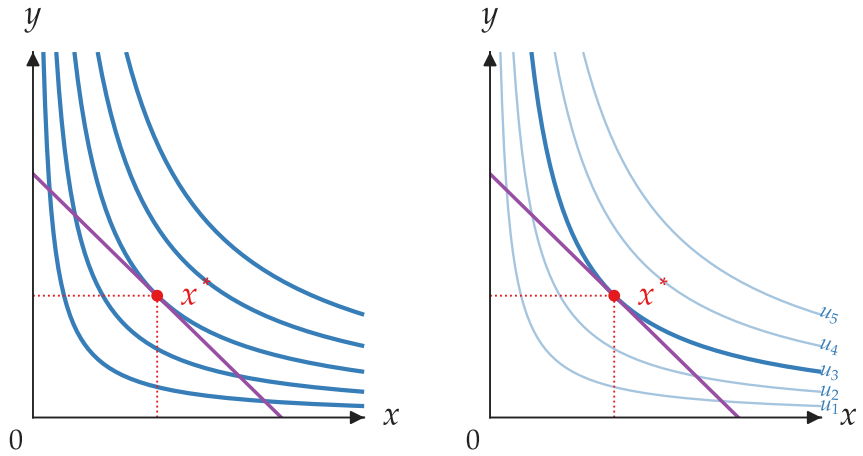

图 3
无差异曲线主次层级。

```
    highlight_level=eq.utility,
    show_ic_labels=True,
    label_style="ordinal",
)
```

`add_budget`

```
cvs.add_budget(
    px, py, income,
    color=None,
    linewidth=None,
    linestyle="-",
    label=None,          # 图例标签（LaTeX）
    fill=False,          # 可行集合阴影
    fill_alpha=None,     # 默认 theme.budget_fill_alpha
    stroke=None,
)
```

绘制预算线 $p_x x + p_y y = I$。

`fill`
bool | Fill · 默认 `False`
`True` 使用主题的阴影设置；传入 `Fill` 可覆盖阴影颜色与透明度。

`stroke`
Stroke | None · 默认 `None`
统一设置预算线的颜色、线宽、线型与透明度，优先于 `color`、`linewidth`、`linestyle` 等简写参数（详见第 4.3 节）。

`label`
str | None · 默认 `None`
图例文字；调用 `show_legend()` 后才会显示图例。

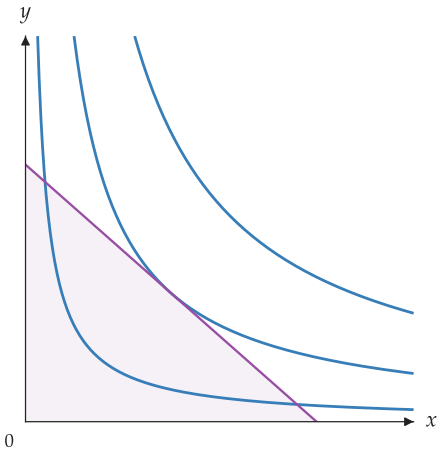

图 4
预算线与阴影。

`add_equilibrium`

```
cvs.add_equilibrium(
    eq,                 # solve() 的返回值
    color=None,
    markersize=None,
    label="x^*",
    drop_dashes=True,   # 到两轴的虚线
    show_ray=False,     # 扩张路径
    drop_stroke=None,
    ray_stroke=None,
    marker=None,
)
```

标记 `solve()` 返回的最优消费组合。

`drop_dashes`

bool · 默认 `True`

显示至两轴的投影线。

`show_ray`

bool · 默认 `False`

显示通过原点与均衡点的射线。

`marker`

Marker | *None* · 默认 `None`

均衡点的标记样式（详见第 4.3 节）。

`add_ray`

```
cvs.add_ray(
    slope,              # dy/dx
    color=None,
    linewidth=None,
    stroke=None,
)
```

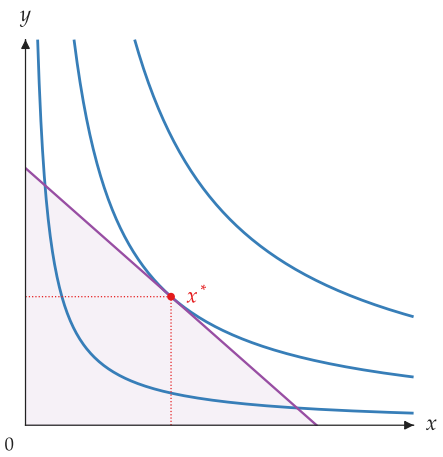

图 5
均衡点与虚线。

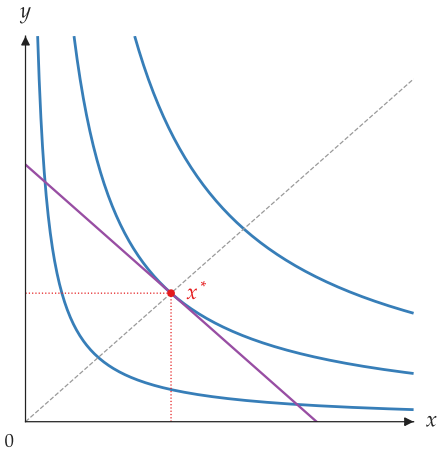

图 6
扩张路径射线。

绘制通过原点、斜率为 `slope`（$\mathrm{d}y/\mathrm{d}x$）的射线，用于标记扩张路径或固定比例。

`add_point`

```
cvs.add_point(
    x, y,
    label=None,
    color=None,
    markersize=None,
    offset=None,        # 标签位移（pt）
    marker=None,
)
```

在指定坐标标记点。

`label`
str | Label | None · 默认 `None`

标记文字，可传入字符串或 `Label`。

`offset`
tuple | None · 默认 `None`

文字相对标记点的位移，单位为 pt。

`marker`
Marker | None · 默认 `None`

标记样式。

`show`
`save`

```
cvs.show()                  # 交互窗口
cvs.save("figure.png")      # .png / .pdf / .svg / .tex
```

`show()` 打开交互窗口；`save()` 根据扩展名导出文件（详见第 10 节），并释放对应的 matplotlib 图形，因此应最后调用。

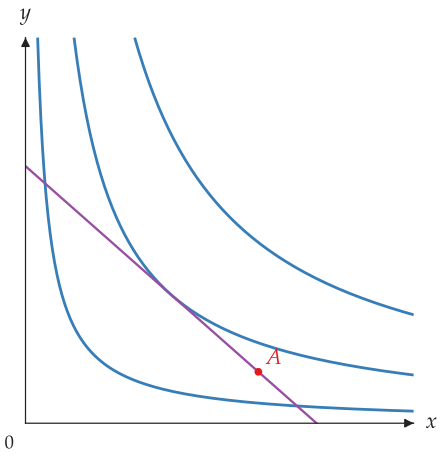

图 7
标记点 A。

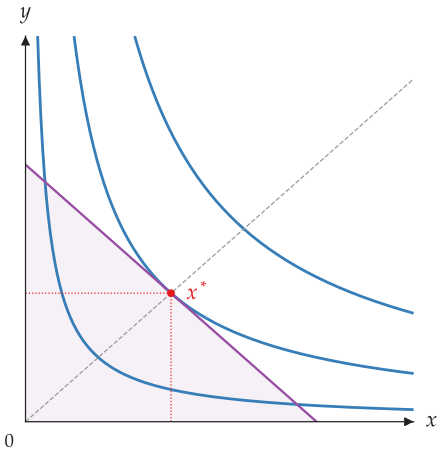

图 8
完整画布。

4.3　样式

LineStyle 定义线型，ArrowStyle 定义箭头样式，Stroke 统一设置线宽、线型、颜色与箭头，可应用于坐标轴、预算线、路径与投影线。

线条样式

LineStyle　LineStyle.SOLID、DASHED、DOTTED、DASHDOT

新增：*v1.7.0*　提供实线、虚线、点线与点画线（参见图 9）。坐标轴接受枚举值或对应字符串；其他线条通过 Stroke(style=...) 设置。

```
import matplotlib.pyplot as plt

from econ_viz import Canvas, LineStyle

styles = [
    LineStyle.SOLID,
    LineStyle.DASHED,
    LineStyle.DOTTED,
    LineStyle.DASHDOT,
]

fig, axes = plt.subplots(1, 4, figsize=(8.5, 2.2))
for ax, style in zip(axes, styles):
    Canvas(
        title=style.value,
        x_line_style=style,
        y_line_style=style,
        fig=fig,
        ax=ax,
    )

# 轴标签会放在箭头外侧；四张图并排时要多留一点水平间距。
fig.subplots_adjust(wspace=0.62)
fig.savefig("line_styles.png", dpi=160, transparent=True)
```

箭头样式

ArrowStyle　ArrowStyle.SIMPLE、TRIANGLE、FANCY、WEDGE

新增：*v1.7.0*　提供四种箭头样式（参见图 10）。坐标轴通过 x_arrow_style、y_arrow_style 设置；其他线条通过 Stroke(arrow=...) 设置。

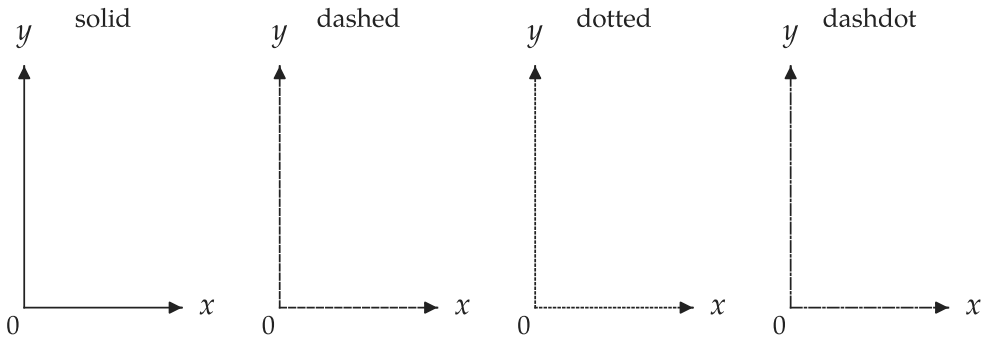

图 9
四种线条样式。

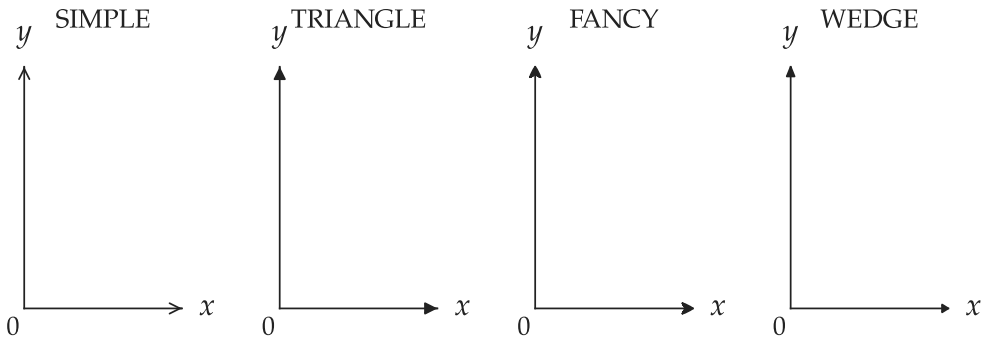

图 10
四种箭头样式。

Stroke

Stroke　`Stroke(width=⟨浮点数⟩, style=⟨线条样式⟩, color=⟨字符串⟩, arrow=⟨箭头样式⟩, opacity=⟨浮点数⟩)`

新增：*v1.7.0*
更新：*v1.9.0*

设置线宽、线型、颜色、箭头与透明度。未指定的字段沿用当前主题。`opacity` 的有效范围为 0 至 1。

Axis

Axis　`Axis(label=None, label_position=None, stroke=None)`

新增：*v1.8.0*

集中设置单一坐标轴的标签、标签位置与线条。`Canvas`、`Figure`、`DemandDiagram` 与 `EdgeworthBox` 均接受 `x_axis` 与 `y_axis`。若同时传入简写参数，`Axis` 中已设置的字段优先。

```
from econ_viz import Axis, Canvas, Label, Stroke

cvs = Canvas(
    x_axis=Axis(
        label=Label(text="x_1", fontsize=16),
        label_position="bottom",
        stroke=Stroke(width=1.2),
    ),
    y_axis=Axis(label="x_2"),
)
```

Marker 与 Label

Marker　`Marker(color=None, size=None, shape=None, opacity=None)`

新增：*v1.8.0*
更新：*v1.9.0*

设置标记点的颜色、大小、形状与透明度。`shape` 接受 matplotlib 的标记字符串，例如 `"o"`、`"s"`、`"^"`、`"D"` 与 `"*"`。

Label　`Label(text=None, position=None, offset=None, color=None, fontsize=None, visible=None, opacity=None)`

新增：*v1.8.0*　更新：*v1.9.0*　设置文字、位置、位移、颜色、字号、可见性与透明度。位置可为上、下、左、右或四个角落；`Label(visible=False)` 隐藏对应文字。

```
from econ_viz import Label, Marker

cvs.add_equilibrium(
    eq,
    marker=Marker(shape="s", size=8),
    label=Label(position="bottom-left", offset=8),
)
cvs.add_point(
    12, 2,
    marker=Marker(color="black", shape="D"),
    label=Label(text="A", position="left", fontsize=14),
)
```

Fill

Fill　`Fill(color=None, opacity=None)`

新增：*v1.8.0*　更新：*v1.9.0*　设置阴影区域的颜色与透明度。`alpha` 是 `opacity` 的兼容简写；同一次调用不得传入不同的 `alpha` 与 `opacity`。

```
from econ_viz import Fill

cvs.add_budget(
    2, 3, 30,
    fill=Fill(color="lightgrey", opacity=0.4),
)
```

4.4　链式调用

`add_*` 方法支持链式调用，最后以 `save()` 导出：

```
Canvas(x_max=20, y_max=15) \
    .add_utility(model, levels=lvls) \
    .add_budget(2.0, 3.0, 30.0, fill=True) \
    .add_equilibrium(eq, show_ray=True) \
    .save("figure.png")
```

第 5 节　多面板图与需求图

5.1　多面板图

Figure　`Figure(⟨版面⟩, x_max=⟨浮点数⟩, y_max=⟨浮点数⟩, ..., shared_x=False, shared_y=False)`

新增：*v1.2.0*　创建多面板图，坐标范围与样式参数与 `Canvas` 相同。使用 `fig[idx]` 获取面板，再调用 `add_utility()`、`add_budget()` 等绘图方法。

例 2

```python
from econ_viz import Figure, Layout, Legend, levels, solve
from econ_viz.models import CobbDouglas

fig = Figure(
    Layout.SIDE_BY_SIDE,
    x_max=20,
    y_max=15,
    x_label="x",
    y_label="y",
    title="Before / After Price Change",
    shared_y=True,
)

cases = [
    (CobbDouglas(alpha=0.5, beta=0.5), 2.0, 3.0, 30.0, r"Before: $p_x=2$"),
    (CobbDouglas(alpha=0.3, beta=0.7), 4.0, 3.0, 30.0, r"After: $p_x=4$"),
]

for idx, (model, px, py, income, title) in enumerate(cases):
    eq = solve(model, px=px, py=py, income=income)
    panel = fig[idx]
    panel.ax.set_title(title)
    panel.add_utility(model, levels=levels.around(eq.utility, n=5), label="IC")
    panel.add_budget(px, py, income, fill=True, label="BC")
    panel.add_equilibrium(eq, show_ray=True)

fig[0].show_legend(legend=Legend(position="upper right"))
fig.save("figure_side_by_side.png")
```

`Layout` 内置版面有 `SINGLE`、`STACKED`、`SIDE_BY_SIDE`、`TOP_TWO_BOTTOM_ONE`、`TOP_ONE_BOTTOM_TWO`、`GRID_2X2` 和 `GRID_3X3`。

新增：*v1.2.0*

5.2　路径

`PricePath`

`IncomePath`

```python
from econ_viz import IncomePath, LinearBudget, PricePath
from econ_viz.models import CobbDouglas

model = CobbDouglas(alpha=0.5, beta=0.5)
budget = LinearBudget(px=2.0, py=2.0, income=40.0)

price_path = PricePath(
    model,
    budget=budget,
    price="px",
    price_range=(0.8, 6.0),
    n=40,
)
```

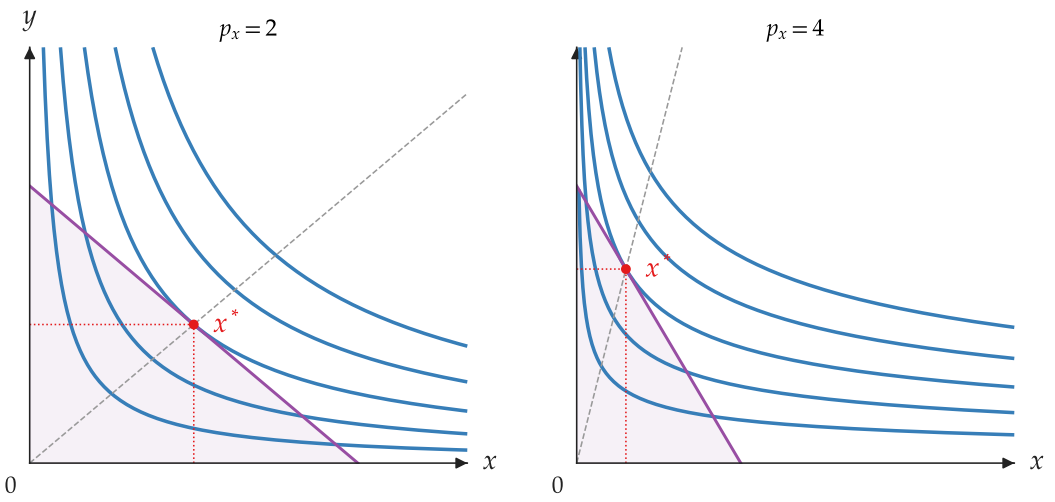

图 11
价格变动前后比较。

```
income_path = IncomePath(
    model,
    budget=budget,
    income_range=(20.0, 80.0),
    n=30,
)
```

新增：*v1.2.0*　`PricePath` 与 `IncomePath` 分别计算价格、收入变动时的最优消费组合。使用 `Canvas.add_path()` 绘制价格消费曲线（PCC）或收入消费曲线（ICC）；`PricePath` 也可用于需求图。

`Canvas.add_path`

```
from econ_viz import Canvas, levels

eq = price_path.equilibria[len(price_path.equilibria) // 2]
lvls = levels.around(eq.utility, n=5)

Canvas(x_max=25, y_max=20) \
    .add_utility(model, levels=lvls) \
    .add_path(price_path, label="PCC") \
    .save("price_path.png")
```

新增：*v1.2.0*　将 `PricePath` 或 `IncomePath` 添加画布。图 12 中，Cobb-Douglas 模型在 p_x 变动时，y 的需求量不变，因此 PCC 为水平线。

5.3　需求图

`DemandDiagram`

```
DemandDiagram(⟨价格路径⟩, title=None, ...)
.add_marshallian_panel(price_markers=None, show_pcc=False,
show_demand_guides=True)
```

新增：*v1.2.0*　根据 `PricePath` 绘制两个面板：上方为预算线与最优消费组合，下方为对应的 Marshall 需求曲线（参见图 13）。目前只接受 `PricePath`，并分别处理平滑、折点与角解。`show_pcc=True` 在上方面板显示价格消费曲线。

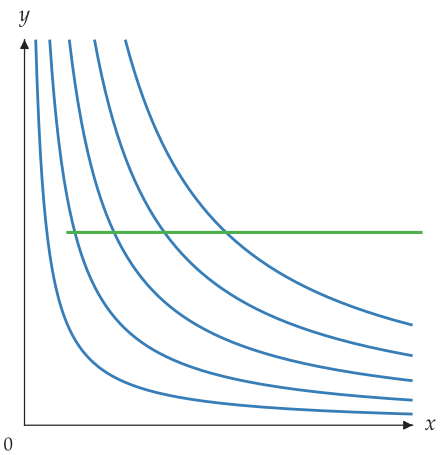

图 12
价格消费曲线。

例 3

```
from econ_viz import DemandDiagram, LinearBudget, PricePath
from econ_viz.models import CobbDouglas

model = CobbDouglas(alpha=0.5, beta=0.5)
budget = LinearBudget(px=2.0, py=2.0, income=40.0)
path = PricePath(
    model,
    budget=budget,
    price="px",
    price_range=(0.8, 6.0),
    n=40,
)

fig = DemandDiagram(path, title="Demand: Cobb-Douglas")
fig.add_marshallian_panel(
    price_markers=[1.5, 4.0],
    show_pcc=False,
    show_demand_guides=True,
)
fig.save("demand_cobb_douglas.png")
```

5.4　价格效应分解

`decompose_price_effect`　`decompose_price_effect(⟨模型⟩, px=(⟨旧值⟩, ⟨新值⟩), py=⟨浮点数⟩, income=⟨浮点数⟩, method=⟨方法⟩)`

新增：*v1.5.0*

将价格效应分解为替代效应与收入效应。Hicks 补偿 (Hicks, 1939) 维持原效用水平；Slutsky 补偿 (Slutsky, 1915) 则使原消费组合在新价格下恰好可负担。返回值包含下列字段：

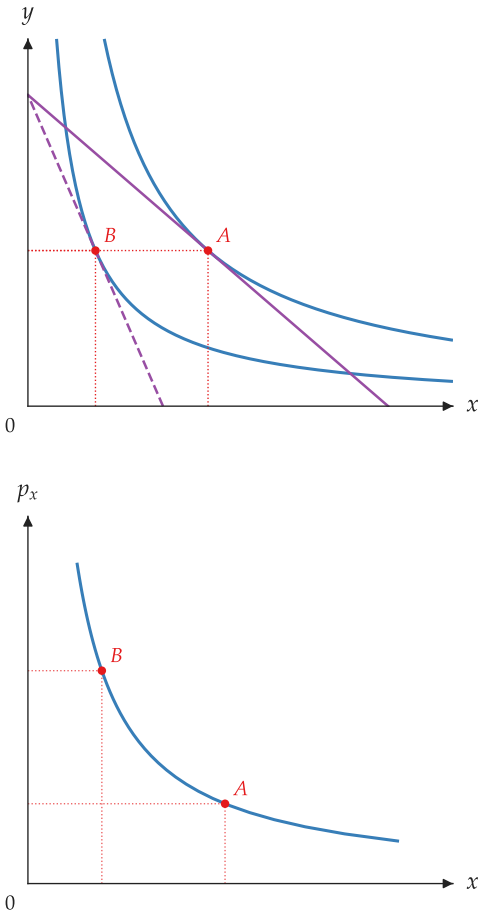

图 13
需求曲线最优点。

`A, B, C` 原始、补偿后与最终的消费组合。

`substitution_effect` 替代效应。

`income_effect` 收入效应。

`total_effect` 总效应。

`compensated_income` 补偿后的收入。

`Canvas.add_decomposition` 在画布上标示 A、B、C 三个消费组合、预算线、无差异曲线与效应箭头。Hicks 分解绘制通过 A 与 C 的曲线；Slutsky 分解另绘制通过 B 的曲线。

新增：*v1.5.0*

更新：*v1.9.0*

`show_curves` 绘制分解用的无差异曲线；已有自定义无差异曲线时设为 `False`，避免重复。

bool · 默认 `True`

`curve_stroke` 无差异曲线的样式。

Stroke | *None* · 默认 `None`

`curve_label` 显示并设置 U_0、U_1 与 U_B 标签。

Label | *None* · 默认 `None`

`substitution, income` 替代效应与收入效应的样式（参见下方 `Effect`）。

Effect | *None* · 默认 `None`

`point_marker` 消费组合的标记。

Marker | *None* · 默认 `None`

`point_label` 消费组合的文字。

Label | *None* · 默认 `None`

`legend` 图例位置与样式（参见下方 `Legend`）。

Legend | *None* · 默认 `None`

`Effect` `Effect(color=None, y=None, label=None, label_position="right", label_offset=4, opacity=None)`

新增：*v1.8.0*

更新：*v1.9.0*

设置替代效应或收入效应的颜色、横轴下方箭头高度、标签及透明度。线条本身仍可由 `substitution_stroke` 或 `income_stroke` 覆盖。

`Legend` `Legend(position="auto", fontsize=None, frame=None, columns=None, visible=None, opacity=None)`

新增：*v1.9.0*

`"auto"` 选择与图形重叠最少的内侧角落；四个角落均与图形重叠时，图例移到右侧。也可指定 `"upper left"` 等内侧角落，或 `"top"`、`"bottom"`、`"left"`、`"right"` 等外侧位置。`Legend(visible=False)` 隐藏图例。

例 4

```
from econ_viz import Canvas, DecompositionMethod, Effect, Label, Legend
from econ_viz.models import CobbDouglas
from econ_viz.optimizer import decompose_price_effect

model = CobbDouglas(alpha=0.5, beta=0.5)
result = decompose_price_effect(
    model,
    px=(2.0, 4.0),
    py=3.0,
    income=60.0,
    method=DecompositionMethod.HICKS,
)

(
    Canvas(x_max=25, y_max=25, title="Hicks decomposition")
    .add_decomposition(
        result,
        show_arrows=True,
        label_effects=True,
        show_x_projections=True,
        substitution=Effect(label="SE"),
        income=Effect(label="IE", opacity=0.8),
        curve_label=Label(position="right"),
        legend=Legend(position="bottom"),
    )
    .save("hicks.png")
)
```

5.5　Edgeworth 箱形图

EdgeworthBox　EdgeworthBox(⟨效用函数 *A*⟩, ⟨效用函数 *B*⟩, total_x=⟨浮点数⟩, total_y=⟨浮点数⟩, ...)

新增：*v1.3.0*　创建两人交换经济的 Edgeworth 箱形图 (Edgeworth, 1881)。*A* 的原点在左下，*B* 的原点在右上。使用 `add_*` 方法添加禀赋、契约曲线、核、价格线与 Walras 均衡。

平滑偏好的契约曲线使用 `method="mrs"`；有折点、角解或自定义分段效用函数时，使用 `method="pareto"`。

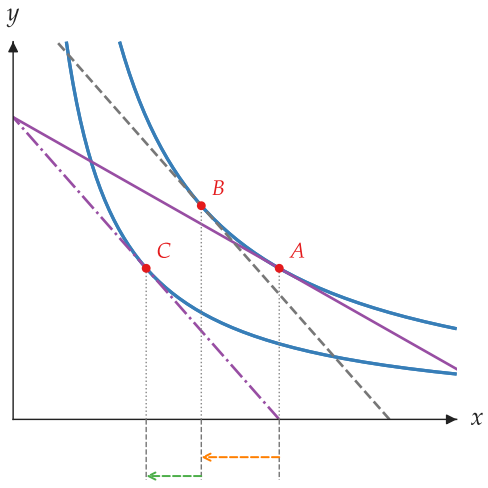

图 14　Hicks 分解。

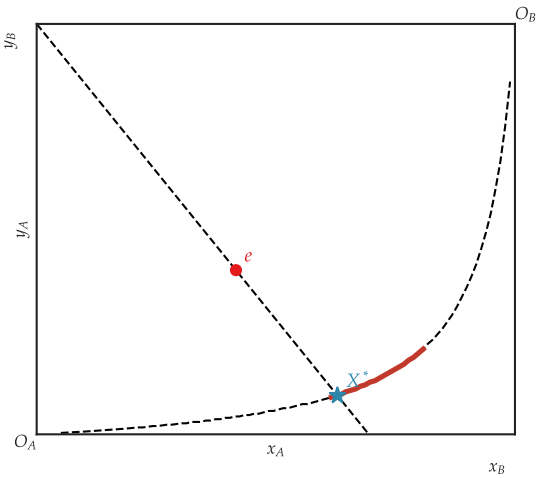

图 15
契约曲线与均衡。

例 5

```python
from econ_viz import EdgeworthBox
from econ_viz.models import CobbDouglas

box = EdgeworthBox(
    CobbDouglas(alpha=0.8, beta=0.2),
    CobbDouglas(alpha=0.2, beta=0.8),
    total_x=12.0,
    total_y=10.0,
    title="Asymmetric Cobb-Douglas",
)

(
    box.add_endowment(5.0, 4.0)
    .add_contract_curve(n=100, method="mrs")
    .add_core()
    .add_price_line(px=1.2, py=1.0)
    .add_walrasian_equilibrium(px=1.2, py=1.0)
    .show_legend(loc="center left", bbox_to_anchor=(1.02, 0.5))
    .save("edgeworth.png")
)
```

第 6 节　核心模型

内置模型位于 `econ_viz.models`，可传入 `solve()` 与 `Canvas.add_utility()`。本章按偏好特性分组，列出各模型的效用函数、参数、特有行为与图形。

6.1　平滑偏好

Cobb-Douglas、CES 与 Translog 的无差异曲线为平滑曲线；最优解取决于模型参数、价格与收入。

Cobb-Douglas

`CobbDouglas`　`CobbDouglas(alpha=0.5, beta=0.5)`

Cobb-Douglas 效用函数 (Cobb & Douglas, 1928)：

$$u(x, y) = x^{\alpha} y^{\beta}$$

正指数决定两种商品的相对权重，无差异曲线以坐标轴为渐近线。

`alpha`　商品 x 的权重。

默认 `0.5`

beta　商品 y 的权重。

默认 0.5

CES

CES　CES(alpha=0.5, beta=0.5, rho=0.5)

CES (Arrow et al., 1961) 通过 rho 调整商品间的替代弹性：

$$u(x,y) = (\alpha x^{\rho} + \beta y^{\rho})^{\frac{1}{\rho}}.$$

替代弹性为 $\sigma = 1/(1-\rho)$。极限形式分别为 Cobb-Douglas（$\rho \to 0$）、完全互补（$\rho \to -\infty$）与完全替代（$\rho \to 1$）。

alpha　商品 x 的权重。

默认 0.5

beta　商品 y 的权重。

默认 0.5

rho　替代参数，且 $\rho \neq 1$。

默认 0.5

Translog

Translog　Translog(alpha_0=0.0, alpha_x=0.5, alpha_y=0.5, beta_xx=0.0, beta_yy=0.0, beta_xy=0.0)

新增：*v1.1.0*　Translog (Christensen et al., 1975) 以对数的一次项、二次项与交叉项表示效用：

$$\ln u(x,y) = \alpha_0 + \alpha_x \ln x + \alpha_y \ln y + \frac{1}{2}\beta_{xx}(\ln x)^2 + \frac{1}{2}\beta_{yy}(\ln y)^2 + \beta_{xy} \ln x \ln y.$$

二次项与交叉项控制曲率。所有 β 系数为零时，退化为 Cobb-Douglas 型的对数线性形式。

alpha_0　对数效用截距。

默认 0.0

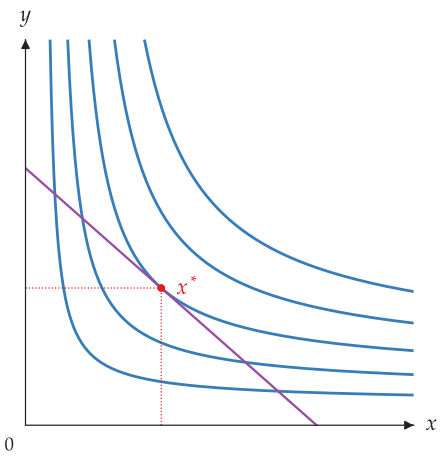

图 16
CES 曲线。

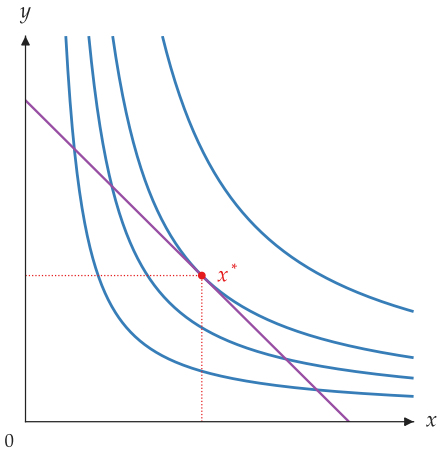

图 17
Translog 示例。

`alpha_x` $\ln x$ 的一次项权重。

默认 `0.5`

`alpha_y` $\ln y$ 的一次项权重。

默认 `0.5`

`beta_xx` x 方向的曲率。

默认 `0.0`

`beta_yy` y 方向的曲率。

默认 `0.0`

`beta_xy` x 与 y 的交叉效应。

默认 `0.0`

6.2　折点与角解

无差异曲线有折点或为直线时，最优解可能位于折点或坐标轴，不能仅以相切条件判定。

完全互补

`Leontief`　`Leontief(a=1.0, b=1.0)`

完全互补效用函数：

$$u(x,y) = \min\{ax, by\}$$

无差异曲线呈 L 型。权重与价格为正时，最优点满足固定比例 $ax = by$。

`a` 商品 x 的权重。

默认 `1.0`

`b` 商品 y 的权重。

默认 `1.0`

完全替代

`PerfectSubstitutes`　`PerfectSubstitutes(a=1.0, b=1.0)`

完全替代效用函数：

$$u(x,y) = ax + by.$$

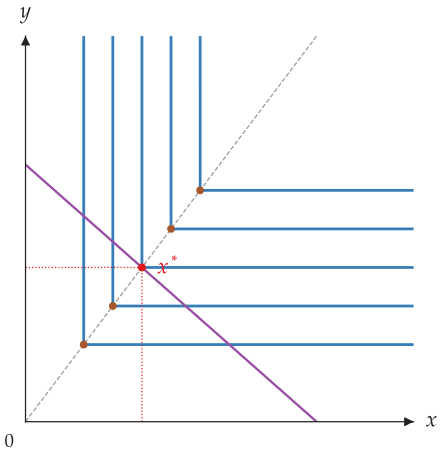

图 18
完全互补射线。

无差异曲线为直线。每元边际效用不同时，最优解为仅购买较高者的角解；两者相等时，预算线上的消费组合皆为最优解。

a　商品 x 的边际效用。

默认 `1.0`

b　商品 y 的边际效用。

默认 `1.0`

最大值效用

maximum　`CustomUtility(func=lambda x, y: np.maximum(a * x, b * y))`

以两个加权数量的最大值定义效用：

$$u(x,y) = \max\{ax, by\}$$

此偏好不具凸性。在正权重与正价格下，最优解位于效用更高的预算线轴截距。此模型以 `CustomUtility` 创建（详见第 7 节）。

a　商品 x 的权重。

默认 `1.0`

b　商品 y 的权重。

默认 `1.0`

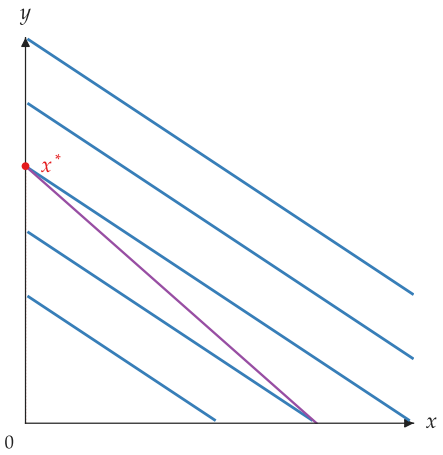

图 19
完全替代角解。

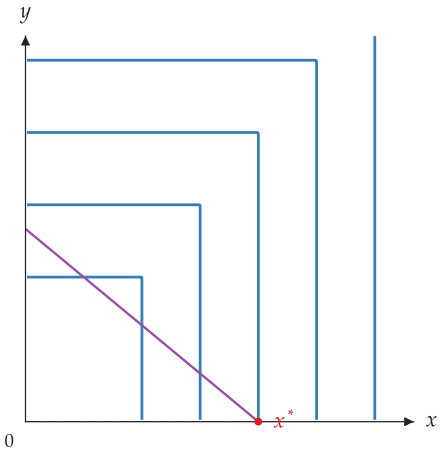

图 20
最大值效用。

6.3　收入与参考点

以下模型分别描述准线性偏好、最低消费需求与具有饱和点的饱和偏好。

准线性

`QuasiLinear`　`QuasiLinear(v_func=numpy.log, linear_in="y")`

准线性效用中，一种商品以线性项表示：

$$u(x,y) = f(x) + y.$$

在内部解范围内，非线性商品的需求不受收入影响。`linear_in` 指定线性项对应的商品。

`v_func`　递增且凹的函数 f。

默认 `numpy.log`

`linear_in`　以线性方式进入的商品。

默认 `"y"`

Haagsma

`Haagsma`　`Haagsma(alpha_x=1.0, alpha_y=2.0, gamma_x=2.0, gamma_y=27.0)`

新增：*v1.9.0*

Haagsma (2012) 效用函数为

$$u(x,y) = \alpha_x \ln(x - \gamma_x) - \alpha_y \ln\left(\gamma_y - y\right).$$

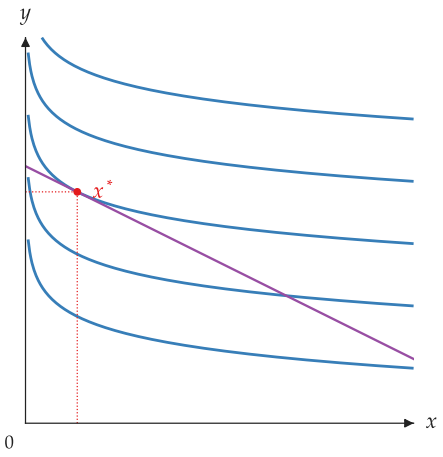

图 21
准线性偏好。

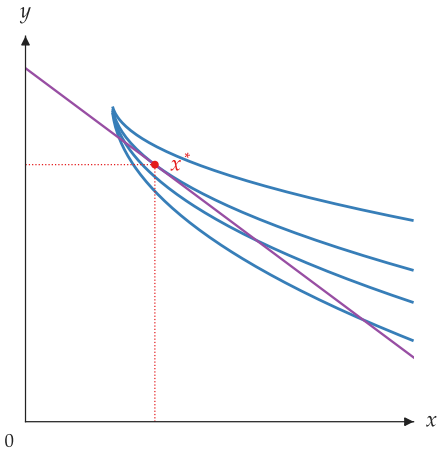

图 22
$\gamma_y = 10$ 的 Haagsma。

定义域为 $x > \gamma_x$ 且 $0 \le y < \gamma_y$，并要求 $0 < \alpha_x < \alpha_y$。商品 x 的 Marshall 需求随收入下降；当 $\gamma_y p_y < I < \gamma_y p_y + \gamma_x p_x$ 时，商品 x 为季芬财。

收入达到 $\gamma_y p_y + \gamma_x p_x$ 时，效用在预算线上没有有限最大值，模型会抛出 `Invalid ParameterError`。

`alpha_x` 商品 x 的权重，必须小于 `alpha_y`。
默认 `1.0`

`alpha_y` 商品 y 的权重。
默认 `2.0`

`gamma_x` 商品 x 的数量下界。
默认 `2.0`

`gamma_y` 商品 y 的数量上界。
默认 `27.0`

`demand(px, py, income)` 返回闭式内部解。

`is_giffen(px, py, income)` 检查指定预算下商品 x 是否为季芬财。

Stone-Geary

`StoneGeary` `StoneGeary(alpha=0.5, beta=0.5, bar_x=1.0, bar_y=1.0)`

Stone-Geary (Geary, 1950; Stone, 1954) 在 Cobb-Douglas 中添加最低消费量：

$$u(x,y) = (x - \bar{x})^{\alpha}(y - \bar{y})^{\beta}.$$

满足基本需求量 $\bar{x}$ 与 $\bar{y}$ 后，剩余收入依权重分配。

`alpha` 超额 x 的权重。
默认 `0.5`

`beta` 超额 y 的权重。
默认 `0.5`

`bar_x` 基本需求量 $\bar{x}$。
默认 `1.0`

`bar_y` 基本需求量 $\bar{y}$。
默认 `1.0`

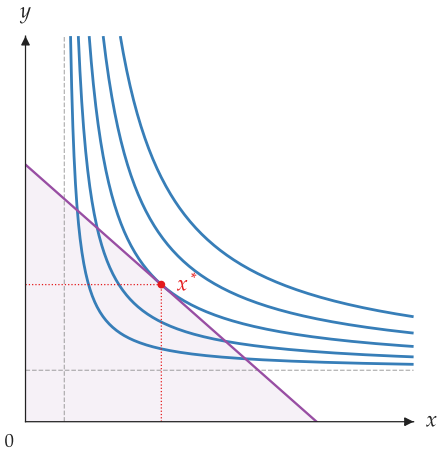

图 23
Stone-Geary。

饱和偏好

`Satiation`　`Satiation(bliss_x=5.0, bliss_y=5.0, a=1.0, b=1.0)`

饱和偏好以 (x^*, y^*) 为效用最高的饱和点：

$$u(x,y) = -a(x-x^*)^2 - b(y-y^*)^2.$$

当 $a, b > 0$，偏离饱和点会使效用下降，无差异曲线为封闭椭圆，偏好不具单调性。

`bliss_x`　饱和点坐标 x^*。

默认 `5.0`

`bliss_y`　饱和点坐标 y^*。

默认 `5.0`

`a`　x 轴方向的曲率。

默认 `1.0`

`b`　y 轴方向的曲率。

默认 `1.0`

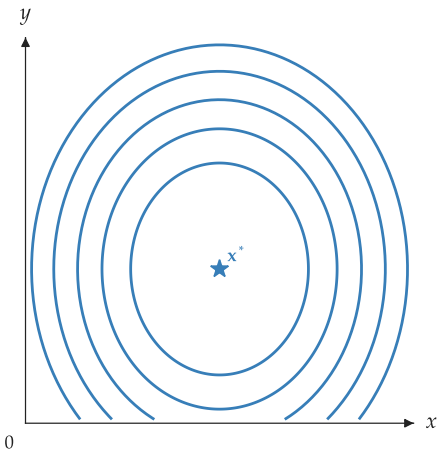

图 24
饱和偏好。

第 7 节　进阶模型

`CustomUtility` 定义二元效用函数。`MultiGoodCD` 表示多商品 Cobb-Douglas；固定其他商品的数量后，可将模型投影为二维图。

自定义效用函数

`CustomUtility`　`CustomUtility(func=⟨可调用对象⟩, name=⟨字符串⟩)`

接受向量化的 Python 函数，返回的模型可用于 `solve()` 与 `Canvas`。函数须接受两个 NumPy 数组，并返回相同形状的效用数组。以下以对数效用为例：

$$u(x, y) = \ln x + \ln y.$$

`func`　以 x 与 y 为变量的向量化效用函数。

`name`　模型的显示名称。

```
import numpy as np
from econ_viz import Canvas, levels, solve
from econ_viz.models import CustomUtility

model = CustomUtility(
    func=lambda x, y: np.log(x) + np.log(y),
    name="log+log",
)
eq = solve(model, px=2.0, py=3.0, income=30.0)

(
    Canvas(x_max=20, y_max=15, title="Custom Utility")
    .add_utility(model, levels=levels.around(eq.utility, n=5))
    .add_budget(2.0, 3.0, 30.0)
    .add_equilibrium(eq)
    .save("custom.png")
)
```

多商品 Cobb-Douglas

`MultiGoodCD`　`MultiGoodCD(⟨权重⟩)`

`MultiGoodCD.freeze`　`.freeze(⟨商品⟩=⟨数量⟩, ...)`

`MultiGoodCD` 表示 N 种商品的 Cobb-Douglas：

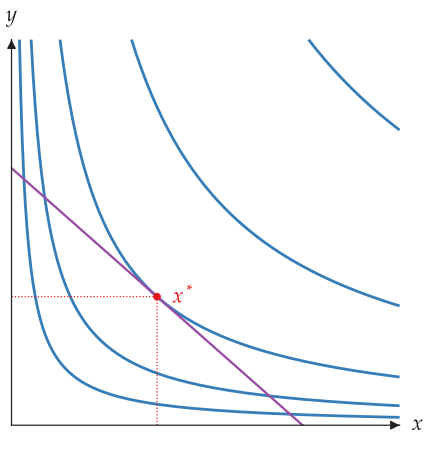

图 25
对数自定义模型。

$$u(x_1, \dots, x_N) = \prod_{i=1}^{N} x_i^{\alpha_i}.$$

使用 `freeze()` 固定 x、y 以外的商品数量，返回可用于二维 `Canvas` 的 `CustomUtility`。

`shares` 商品名称与指数 α_i 的对应。

`freeze(...)` x、y 以外商品的固定数量。

```
from econ_viz import Canvas, levels, solve
from econ_viz.models import MultiGoodCD

model = MultiGoodCD({"x": 0.3, "y": 0.3, "z": 0.4})
two_good_model = model.freeze(z=10.0)
eq = solve(two_good_model, px=2.0, py=3.0, income=30.0)

(
    Canvas(x_max=20, y_max=15, title="Multi-Good Cobb-Douglas")
    .add_utility(
        two_good_model,
        levels=levels.around(eq.utility, n=5),
    )
    .add_budget(2.0, 3.0, 30.0, fill=True)
    .add_equilibrium(eq)
    .save("multigood.png")
)
```

第 8 节　分析

分析 API 提供 Marshall 需求的比较静态、Slutsky 矩阵，以及效用函数的齐次性与位似性检查。

8.1　比较静态

`comparative_statics` `comparative_statics(⟨模型⟩, px=⟨浮点数⟩, py=⟨浮点数⟩, income=⟨浮点数⟩)`

新增：*v1.1.0*

以有限差分估计两商品 Marshall 需求对 p_x、p_y 与收入的六个偏导数，返回 `Comparative Statics`。

- 在 `solve(...)` 的结果附近使用中央有限差分，默认的相对步长是 `1e-3`。

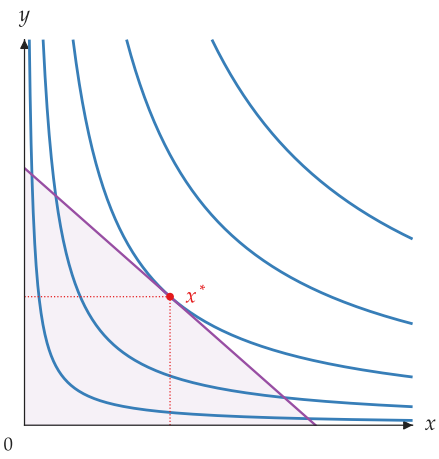

图 26

三商品固定 $z = 10$。

- 检测到自身价格导数为正或收入导数为负时会发出警告，分别对应季芬财与劣等财的需求特性。

`dx_dpx` $\partial x^*/\partial p_x$。

float

`dx_dpy` $\partial x^*/\partial p_y$。

float

`dx_dI` $\partial x^*/\partial I$。

float

`dy_dpx` $\partial y^*/\partial p_x$。

float

`dy_dpy` $\partial y^*/\partial p_y$。

float

`dy_dI` $\partial y^*/\partial I$。

float

```
from econ_viz.models import CobbDouglas
from econ_viz.optimizer import comparative_statics

model = CobbDouglas(alpha=0.4, beta=0.6)
cs = comparative_statics(model, px=2.0, py=3.0, income=60.0)

print(round(cs.dx_dpx, 1), round(cs.dx_dpy, 1), round(cs.dx_dI, 1))
print(round(cs.dy_dpx, 1), round(cs.dy_dpy, 1), round(cs.dy_dI, 1))

# -6.0 0.0 0.2
# 0.0 -4.0 0.2
```

8.2　Slutsky 矩阵

`slutsky_matrix` `slutsky_matrix(⟨模型⟩, px=⟨浮点数⟩, py=⟨浮点数⟩, income=⟨浮点数⟩)`

更新：*v1.2.3*

以 Marshall 需求导数与收入效应计算两商品的 Slutsky 替代矩阵，并检查对称性、负半定性与齐次性；检查未通过时发出警告。使用 `as_array()` 可转为 NumPy 数组。

```
from econ_viz import slutsky_matrix
from econ_viz.models import CobbDouglas

S = slutsky_matrix(
    CobbDouglas(alpha=0.4, beta=0.6),
    px=2.0, py=3.0, income=60.0,
)

print(round(S.s_xx, 1), round(S.s_xy, 1))
print(round(S.s_yx, 1), round(S.s_yy, 1))
print(S.as_array().round(1))

# -3.6 2.4
# 2.4 -1.6
# [[-3.6  2.4]
#  [ 2.4 -1.6]]
```

8.3　齐次性

`HomogeneityAnalyzer` `HomogeneityAnalyzer(⟨模型⟩)`

新增：*v1.1.0*

检查效用函数的齐次性、位似性与需求的零次齐次性。

`degree()` 估计齐次次数，详见第 8.4 节。

`euler_check(x, y)` 计算指定消费组合的 Euler 定理残差。

`is_homothetic()` 边际替代率（MRS）在等比例缩放下是否不变。

`demand_degree_zero(px, py, income)` Marshall 需求是否为零次齐次。

```python
from econ_viz.analysis import HomogeneityAnalyzer
from econ_viz.models import CobbDouglas

analyzer = HomogeneityAnalyzer(CobbDouglas(alpha=0.4, beta=0.6))
result = analyzer.degree()

print(round(result.degree, 6))
print(result.returns_to_scale)
print(round(analyzer.euler_check(3.0, 4.0), 6))
print(analyzer.is_homothetic())
print(analyzer.demand_degree_zero(px=2.0, py=3.0, income=60.0))

# 1.0
# ReturnsToScale.CONSTANT
# 0.0
# True
# True
```

8.4　规模报酬

`degree()` 返回 `HomogeneityResult`，包含下列字段：

`degree` *float* 估计的齐次次数。

`is_homogeneous` *bool* 是否为齐次函数。

`returns_to_scale` *ReturnsToScale* 规模报酬的分类，取值如下。

Cobb-Douglas 的齐次次数为 $\alpha + \beta$：

$$u(\lambda x, \lambda y) = \lambda^{\alpha+\beta} u(x, y).$$

`INCREASING` $\alpha + \beta > 1$，规模报酬递增。

`CONSTANT` $\alpha + \beta = 1$，规模报酬不变。

`DECREASING` $\alpha + \beta < 1$，规模报酬递减。

`NOT_HOMOGENEOUS` 没有一致的齐次次数。

```python
def shifted_utility(x, y):
    return x**0.4 * y**0.6 + 1.0


models = [
    CobbDouglas(alpha=0.7, beta=0.6),
    CobbDouglas(alpha=0.4, beta=0.6),
    CobbDouglas(alpha=0.2, beta=0.5),
    shifted_utility,
]

for model in models:
    result = HomogeneityAnalyzer(model).degree()
    degree = None if result.degree is None else round(result.degree, 1)
    print(degree, result.returns_to_scale.name)

# 1.3 INCREASING
```

```
# 1.0 CONSTANT
# 0.7 DECREASING
# None NOT_HOMOGENEOUS
```

第 9 节　主题与设置

Theme 定义图形的默认外观；Config 从 TOML 文件加载主题覆盖与字体设置。Canvas、Figure、DemandDiagram、价格效应分解及 EdgeworthBox 共用这些样式对象。

9.1　内置主题

themes

```
from econ_viz import Canvas, themes

cvs = Canvas(theme=themes.paper)
```

更新：*v1.11.0*

Python API 提供下列主题。主题对象不可变；需要调整字段时，可创建新的 Theme，或通过 Config 覆盖指定字段。

表 4
内置主题

名称	特性
default	色盲友好的默认配色
colorblind	与 default 相同的配色别名
nord	采用 Nord 色盘的冷色系主题
paper	透明背景与出版用细线条
monochrome	以线型与标记区分图层的黑白主题
presentation	放大标签、线条与标记，供投影使用
dark	深色背景与相应的前景色

Python API 与配置文件的 base 可使用全部七个名称[5]。

默认主题的颜色取自一组色盲友好的配色 (thriveth, 2014)。

themes.COLORBLIND_CYCLE_HEX 与 themes.COLORBLIND_CYCLE_RGB 提供这组配色的十六进制与 RGB 值。经济学教学大量依赖图形，因此不应只靠颜色传达含义，以免视觉障碍的学生无法辨识 (Kugler & Andrews, 1996)。

9.2　自定义主题

Theme

```
from econ_viz import Theme

my_theme = Theme(
    name="custom",
    background_color="white",
    label_scale=1.0,
    axis_color="#333333",
    label_color="#333333",
    ic_color="#2563eb",
    secondary_ic_color="#93c5fd",
    secondary_ic_opacity=0.45,
```

[5]在命令行中直接传入 --theme 时，只接受 colorblind 别名以外的六个主题。

```
    budget_color="#dc2626",
    eq_color="#16a34a",
)
```

更新：*v1.12.0*

Theme 保存图形元素的颜色、线宽与样式默认值。未指定的字段使用类默认值；绘图方法明确收到的样式参数优先于主题。

`name`
主题标识名称。

str
`background_color`
图形与坐标区域的背景色；`None` 保持透明。

str | *None* · 默认 `None`
`label_scale`
普通标签字号的倍数。

float · 默认 `1.0`
`axis_color`
坐标轴颜色。

str · 默认 `"#222222"`
`label_color`
坐标标签与原点文字颜色。

str · 默认 `"#222222"`
`ic_color`
无差异曲线颜色。

str · 默认 `"#377EB8"`
`ic_linewidth`
无差异曲线线宽。

float · 默认 `1.8`
`secondary_ic_color`
非焦点无差异曲线颜色；`None` 沿用 `ic_color`。

str | *None* · 默认 `None`
`secondary_ic_linewidth`
非焦点无差异曲线线宽。

float · 默认 `1.0`
`secondary_ic_opacity`
非焦点无差异曲线不透明度。

float · 默认 `0.45`
`path_color`
PCC 与 ICC 路径颜色。

str · 默认 `"#4DAF4A"`
`budget_color`
预算线颜色。

str · 默认 `"#984EA3"`
`budget_fill_alpha`
预算集阴影不透明度。

float · 默认 `0.08`
`eq_color`
均衡点与投影线颜色。

str · 默认 `"#E41A1C"`
`eq_markersize`
均衡点标记大小。

float · 默认 `4.0`

Theme 也包含射线、折点、补偿预算线与价格效应箭头的颜色和线宽。`axis_stroke`、`drop_stroke`、`projection_stroke`、`guide_stroke` 与 `box_stroke` 保存其他线条的默认值；标记、标签、填色及图例则由 `Marker`、`Label`、`Fill` 与 `Legend` 属性提供（详见第 4.3 节）。

9.3　配置文件

Config.load
Config.use
Config.reset

```python
from econ_viz import Config

Config.load("econ-viz.toml").use()
# 此后创建的图形使用该设置

Config.reset()
# 恢复内置默认值
```

新增：*v1.10.0*

`Config.load()` 读取 TOML 文件并返回 `Config`。`use()` 将其设为后续图形的默认配置；`reset()` 恢复内置默认值。

教程：以配置文件统一图形样式

同一份讲义或论文的图形，通常要共用相同的颜色与线条。与其在每个 `Canvas` 重复传入参数，不如把样式写在项目根目录的 `econ-viz.toml`，让 Python 与命令行读取同一份配置。

1. **创建模板**。在项目根目录运行下列命令，会生成含注释的 `econ-viz.toml`，列出每个配置节可用的名称与字段：

   ```
   econ-viz init
   ```

   模板中的设置默认都是注释；只保留需要修改的项目即可，其余可以删除。

2. **选择基础主题**。`base` 指定起始的内置主题（表 4），其余设置都在它之上覆盖：

   ```toml
   base = "paper"
   ```

3. **覆盖颜色与样式**。只写出要改的字段：

   ```toml
   [color]
   ic = "#1B4F72"
   eq = "#C0392B"

   [stroke.budget]
   width = 1.6
   style = "dashed"

   [marker.eq]
   size = 5

   [label.point]
   fontsize = 11
   position = "right"
   ```

   配置节名称对应主题属性：`[color]` 的 `ic` 对应 `ic_color`，`[stroke.budget]` 对应 `theme.budget_stroke`；配置节内的字段就是 `Stroke`、`Marker`、`Label`、`Fill`、`Legend` 的参数（详见第 4.3 节）。字体写在 `[font]` 的 `text` 与 `math`。省略的字段保留基础主题的值。

4. **在 Python 中应用**。在创建任何图形之前加载一次：

   ```python
   from econ_viz import Config

   Config.load().use()    # 默认读取当前目录的 econ-viz.toml
   ```

   此后创建的 `Canvas`、`Figure`、`DemandDiagram` 与 `EdgeworthBox` 都使用这组配置；调用 `Config.reset()` 可恢复内置默认值。

5. **在命令行应用**。以 `--config` 指定配置文件：

```
econ-viz plot --config econ-viz.toml --model cobb-douglas \
  --px 2 --py 3 --income 30 --output figure.png
```

配置的优先级由高到低为：直接传入 Canvas 或绘图方法的参数、配置文件、基础主题。例如 `Canvas(theme=themes.default)` 会略过配置文件中的主题。

配置文件写错时，`Config.load()` 会抛出 `InvalidParameterError`，并列出可用的名称。例如把 `[stroke.budget]` 误写成 `[stroke.budgt]`：

```
econ-viz.toml: [stroke.budgt]: no such setting (choose: axis, box,
budget, ...)
```

第 10 节　导出

10.1　输出格式

Canvas.save　`cvs.save(⟨路径⟩)`

根据 `path` 的扩展名决定输出格式。`Figure`、`DemandDiagram` 与 `EdgeworthBox` 使用相同接口。

- PNG：位图，分辨率由画布的 `dpi` 设置，默认为 300。
- PDF、SVG：向量图，可在打印或缩放时保留线条质量。
- TeX：TikZ 源代码（`.tex`），可嵌入 LaTeX 文件。通过关键字参数 `tikz_scale` 与 `tikz_standalone` 调整输出。

```
cvs.save("figure.png")   # PNG，依画布 DPI（默认 300）
cvs.save("figure.pdf")   # PDF（向量）
cvs.save("figure.svg")   # SVG（向量）
cvs.save("figure.tex")   # TikZ
```

`dpi` 仅影响位图。预览时可降低分辨率：

```
cvs = Canvas(x_max=20, y_max=15, dpi=150)   # 降低 DPI，加快预览
```

10.2　GIF 动画

使用 `Animator.save()` 导出 GIF，完整示例详见第 12 节。

```
from econ_viz.animation import Animator

Animator(draw_frame, frames=frames).save("animation.gif", fps=12,
dpi=120)
```

10.3　交互窗口

`cvs.show()` 打开 matplotlib 交互窗口，不会保存文件。命令行的 `plot` 省略 `--output` 时也会打开窗口（详见第 11.2 节）：

```
econ-viz plot --model cobb-douglas --px 2 --py 3 --income 30
```

第 11 节　命令行工具

命令行工具可绘图、批量处理及输出需求公式。以 `uv tool install econ-viz` 安装为独立工具，或在 uv 项目中于命令前加上 `uv run`（详见第 2.5 节）。

表 5
命令行命令

命令	说明
`econ-viz help [⟨命令⟩]`	显示命令行工具或特定命令的说明
`econ-viz models`	列出所有支持的效用模型
`econ-viz plot ...`	生成并导出图形
`econ-viz solve-tex ...`	以 TeX 格式输出 Marshall 需求的闭式解
`econ-viz init [path]`	创建 `econ-viz.toml` 配置模板

11.1　说明与模型

`econ-viz help`

`econ-viz help [⟨命令⟩]`

未指定参数时列出所有命令；指定命令名称时显示该命令的选项。

```
econ-viz help          # 所有命令
econ-viz help plot     # plot 选项
econ-viz help models   # models 选项
```

`econ-viz models`

列出命令行接口支持的模型名称与参数。`--model` 使用 kebab-case 名称，与 Python 类名称不同（参见表 6）。参数选项详见第 11.2 节。

表 6
命令行模型

模型	`--model` 值	必要参数	可选参数
Cobb-Douglas	`cobb-douglas`	`alpha`、`beta`	–
CES	`ces`	`rho`	`alpha`、`beta`
完全互补	`leontief`	`a`、`b`	–
完全替代	`perfect-substitutes`	`a`、`b`	–
饱和偏好	`satiation`	`bliss-x`、`bliss-y`	`a`、`b`
准线性	`quasi-linear`	–	`v-func`、`linear-in`
Stone-Geary	`stone-geary`	–	`alpha`、`beta`、`bar-x`、`bar-y`
Translog	`translog`	–	`alpha-0`、`alpha-x`、`alpha-y`、`beta-xx`、`beta-yy`、`beta-xy`

11.2　绘图

`econ-viz plot`

`econ-viz plot --model ⟨名称⟩⟨选项⟩`

`econ-viz plot --latex ⟨算式⟩⟨选项⟩`

使用 `--model` 指定模型，或使用 `--latex` 输入效用函数，两者择一。指定 `--output` 时导出文件，省略时打开交互窗口。

预算线与均衡点需要完整的价格与收入。`--no-budget`、`--no-equilibrium`、`--no-curves` 可分别省略对应图层，且可并用（参见表 7）。

表 7
图层控制选项

情况	结果
提供完整 `--px`、`--py`、`--income`	绘制预算线并求解均衡
省略任一价格或收入	仅绘制无差异曲线，不画预算线与均衡点
加上 `--no-equilibrium`	跳过求解与均衡标注，即使提供了价格与收入
加上 `--no-budget`	隐藏预算线，即使提供了价格与收入
`--no-curves` 搭配完整价格与收入	只呈现预算线与均衡点，不画无差异曲线

```
# 指定模型名称
econ-viz plot --model cobb-douglas --alpha 0.5 --beta 0.5 ...

# LaTeX 表达式
econ-viz plot --latex "x^{0.4} y^{0.6}" ...
```

例 6

```
# Cobb-Douglas，可行集合加阴影
econ-viz plot --model cobb-douglas --alpha 0.5 --beta 0.5 \
              --px 2 --py 3 --income 30 \
              --fill --output cobb_douglas.png

# LaTeX 输入，Nord 主题，扩张路径
econ-viz plot --latex "x^{0.4} y^{0.6}" \
              --px 2 --py 3 --income 30 \
              --theme nord --show-ray \
              --output cd_latex.png

# 完全互补，加大画布
econ-viz plot --model leontief --a 1 --b 2 \
              --px 2 --py 3 --income 30 \
              --x-max 20 --y-max 15 \
              --output leontief.png

# CES，只画无差异曲线
econ-viz plot --model ces --rho -0.5 \
              --x-max 20 --y-max 15 --n-curves 6 \
              --no-budget --no-equilibrium \
              --output ces.png

# 饱和（饱和点）
econ-viz plot --model satiation --bliss-x 6 --bliss-y 4 \
              --x-max 12 --y-max 10 \
              --no-budget --no-equilibrium \
              --output satiation.png

# 不加 --output：打开窗口
econ-viz plot --model cobb-douglas --px 2 --py 3 --income 30
```

选择模型

`--model`，`-m`　模型名称，完整列表可用 `econ-viz models` 查询。

`--latex`，`-l`　LaTeX 表达式，支持 Cobb-Douglas、完全互补、完全替代与 CES（详见第 14 节）。

价格与收入

`--px`　商品 x 的价格。

`--py`　商品 y 的价格。

`--income`　消费者的收入。

模型参数

`--alpha`　alpha 参数（Cobb-Douglas、CES）。

默认 `0.5`

`--beta` beta 参数（Cobb-Douglas、CES）。

默认 `0.5`

`--a` 参数 a（完全互补、完全替代、饱和）。

默认 `1.0`

`--b` 参数 b（完全互补、完全替代、饱和）。

默认 `1.0`

`--rho` 替代参数（CES）。

默认 `0.5`

`--bliss-x` 饱和点的 x 坐标（饱和）。

默认 `5.0`

`--bliss-y` 饱和点的 y 坐标（饱和）。

默认 `5.0`

准线性、Stone-Geary 与 Translog 另有下列选项，可按模型使用：

`--v-func` 准线性的非线性函数，接受 `log` 或 `sqrt`。

默认 `log`

`--linear-in` 准线性的线性商品，接受 `x` 或 `y`。

默认 `y`

`--bar-x` Stone-Geary 的最低消费量 $\bar{x}$。

默认 `1.0`

`--bar-y` Stone-Geary 的最低消费量 $\bar{y}$。

默认 `1.0`

`--alpha-0` Translog 的截距。

默认 `0.0`

`--alpha-x` Translog 中 $\ln x$ 的一次项权重。

默认 `0.5`

`--alpha-y` Translog 中 $\ln y$ 的一次项权重。

默认 `0.5`

`--beta-xx` Translog 中 x 方向的曲率。

默认 `0.0`

`--beta-yy` Translog 中 y 方向的曲率。

默认 `0.0`

`--beta-xy` Translog 中 x 与 y 的交叉项。

默认 `0.0`

```
econ-viz plot --model stone-geary --bar-x 2 --bar-y 2 \
              --px 2 --py 3 --income 30 \
              --x-max 20 --y-max 15 --output stone_geary.png
```

此例的最低消费支出为 $2\times 2+3\times 2=10$，收入 30 高于此限制。若将收入改为 10 或更低，求解会因不符合模型定义域而失败。

画布与图层

`--x-max` 横轴上限。

默认 `10`

`--y-max` 纵轴上限。

默认 `10`

`--x-label` 横轴标签。

默认 `x`

`--y-label` 纵轴标签。

默认 `y`

`--title` 图形标题。

`--theme` 主题名称：`default`、`nord`、`paper`、`monochrome`、`presentation` 或 `dark`（详见第 9 节）。

默认 `default`

`--config` TOML 配置文件路径。命令行参数的优先级高于配置文件。

`--n-curves` 无差异曲线的条数。

默认 `5`

`--dpi` 位图输出分辨率。

默认 `300`

`--fill` 为预算线下方的可行集合加上阴影。

`--show-ray` 画出通过最优点的扩张路径射线。

`--no-budget` 不画预算线。

`--no-equilibrium` 不画均衡点。

`--no-curves` 不画无差异曲线。

`--output`, `-o` 输出文件（`.png`、`.pdf`、`.svg`、`.tex`）；省略时打开交互窗口。
批量生成图形时，为每组参数指定不同文件名，并先创建输出目录。`--x-max` 与 `--y-max` 不会随价格自动调整；轴截距或最优点超出范围时，应同步增大画布上限。

11.3　创建配置文件

`econ-viz init` `econ-viz init [⟨路径⟩] [--force]`

新增：*v1.10.0*

创建带注释的 `econ-viz.toml` 模板。省略路径时写入当前目录；文件已存在时，必须使用 `--force` 才会覆盖。

```
econ-viz init
econ-viz init config/figures.toml
econ-viz plot --config econ-viz.toml --model cobb-douglas \
              --px 2 --py 3 --income 30 --output figure.png
```

11.4　需求公式

`econ-viz solve-tex` `econ-viz solve-tex --model ⟨名称⟩⟨选项⟩`

以 TeX 格式输出 Marshall 需求的闭式解，可用于支持 TeX 数学式的文件。

```
# 数值参数
econ-viz solve-tex --model cobb-douglas --alpha 0.4 --beta 0.6

# 符号参数
econ-viz solve-tex --model cobb-douglas --symbolic-params

# 自定义价格与收入符号
econ-viz solve-tex --model leontief --a 2 --b 3 \
                   --px-symbol p_1 --py-symbol p_2 --income-symbol M
```

支持 `cobb-douglas`、`leontief`、`perfect-substitutes` 及其 LaTeX 表达式。此命令只输出公式文字，不生成图像文件。

`--symbolic-params` 保留模型参数符号，例如 α、β；未使用时代入指定的参数值。

`--px-symbol` 商品 x 价格的符号。

默认 `p_x`

`--py-symbol` 商品 y 价格的符号。

默认 `p_y`

`--income-symbol` 收入的符号。

默认 `I`

第 12 节　动画

`Animator` 根据指定的参数序列调用绘图函数，将返回的 `Canvas` 或 `Figure` 导出为 GIF。使用前需安装 `animation` 可选依赖（详见第 2.4 节）。

`Animator` `Animator(⟨绘图函数⟩, frames=⟨数值⟩).save(⟨路径⟩, fps=12, dpi=120, loop=0)`

新增：*v1.4.0*

依次将 `frames` 的值传入 `draw`，每次返回的图形对应一个帧。

- `save()` 使用 Pillow，不需要 `ffmpeg`。
- 导出前将帧合成至白色背景，避免残影。
- `fps`、`dpi` 与 `loop` 分别设置帧率、分辨率与循环次数。

例 7

```python
import numpy as np

from econ_viz import Canvas, levels, solve
from econ_viz.animation import Animator
from econ_viz.models import CobbDouglas

def draw(px: float) -> Canvas:
    model = CobbDouglas(alpha=0.5, beta=0.5)
    eq = solve(model, px=px, py=2.0, income=20.0)
    lvls = levels.around(eq.utility, n=5)

    return (
        Canvas(
            x_max=14, y_max=12,
            x_label="X_1", y_label="X_2",
            title="Price sweep"
        )
        .add_utility(model, levels=lvls)
        .add_budget(px=px, py=2.0, income=20.0, fill=True)
        .add_equilibrium(eq, show_ray=True, drop_dashes=True)
    )

Animator(draw, frames=np.linspace(1.0, 6.0, 45)).save(
    "price_sweep.gif",
    fps=12,
    dpi=120,
)
```

`examples/scripts/animation.py` 提供四种示例[6]：

- 参数变动：固定价格与收入，调整效用函数参数。
- 价格变动：固定效用函数与 p_y，调整 p_x，呈现预算线旋转，写法见上例。
- 收入变动：固定效用函数与价格，调整收入，呈现预算线平移。
- 预算线变动：仅显示预算线，省略效用图层。

[6] 各示例均包含 Cobb-Douglas、CES、完全替代与完全互补模型。

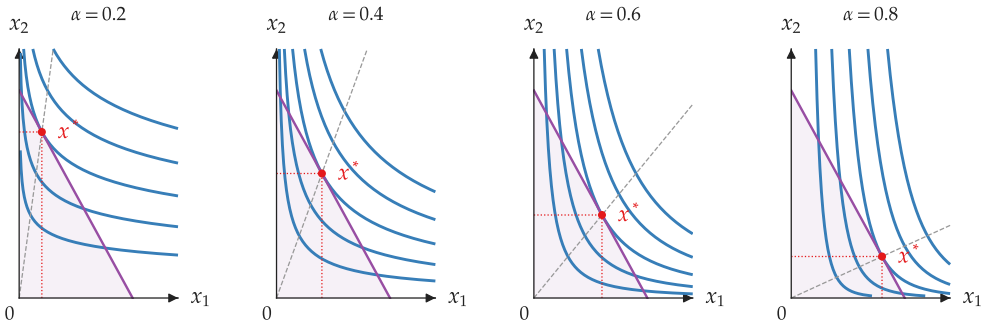

图 27
参数变动。

四种示例的差异只在于 `draw()` 每次固定哪些值、调整哪个值。以参数变动为例，将上例的 `draw()` 改为接收 `alpha`，价格固定为 $p_x = 2$：

```
def draw(alpha: float) -> Canvas:
    model = CobbDouglas(alpha=alpha, beta=1.0 - alpha)
    eq = solve(model, px=2.0, py=2.0, income=20.0)
    ...

Animator(draw, frames=np.linspace(0.1, 0.9,
45)).save("parameter_sweep.gif")
```

收入变动将 `draw()` 的参数改为 `income`，并把该值传入 `add_budget()` 与 `solve()`。只呈现预算线时，省略 `add_utility()` 与 `add_equilibrium()`。

图 27 至图 29 各节录四个帧。

第 13 节　交互组件

`WidgetViewer` 在 Jupyter 中提供滑块与数值输入字段，调整参数后会更新图形。使用前需安装 `interactive` 可选依赖（详见第 2.4 节）。

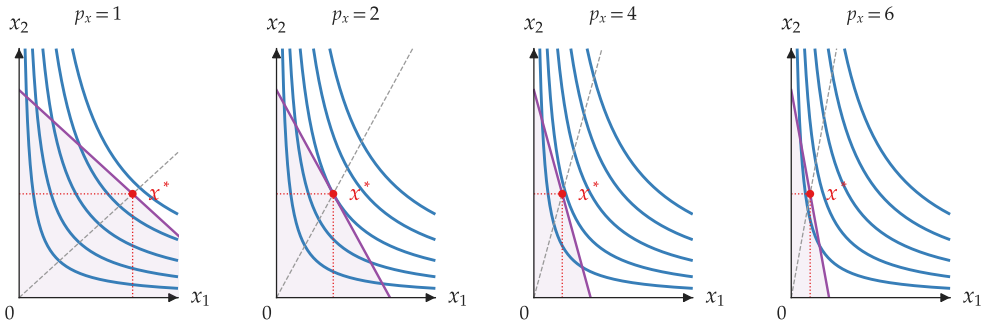

图 28
价格变动。

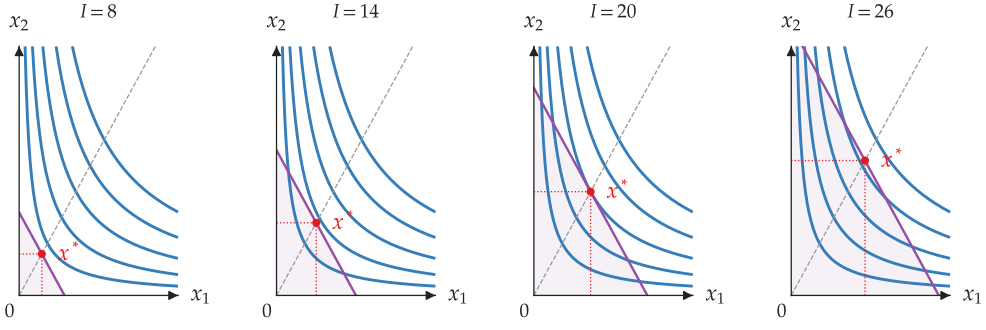

图 29
收入变动。

WidgetViewer　WidgetViewer(⟨绘图函数⟩, ⟨名称⟩=(⟨下限⟩, ⟨上限⟩, ⟨间距⟩), ...).show()

新增：*v1.4.0*

传入绘图函数 draw，并以（最小值，最大值，步长）指定各参数范围。每个参数会生成同步的 FloatSlider 与 FloatText；数值变更后重新调用 draw，并替换单元格中的旧图。draw 的写法与 Animator 相同（详见第 12 节），只是可同时接收多个参数。以第 12 节的价格变动示例为基础，让 draw() 另外接收 alpha：

例 8

```
from econ_viz.interactive import WidgetViewer

def draw(alpha: float, px: float) -> Canvas:
    model = CobbDouglas(alpha=alpha, beta=1.0 - alpha)
    ...  # 其余与动画示例相同

WidgetViewer(
    draw,
    alpha=(0.2, 0.8, 0.05),
    px=(1.0, 6.0, 0.25),
).show()
```

13.1　笔记本设置

在新的 Jupyter 或 Colab 环境中，按下列顺序运行[7]：

1. 运行安装单元格。
2. 若安装过程升级 ipywidgets、traitlets 或 IPython，重新启动运行环境。
3. 重新启动后，从导入软件包的单元格继续运行，无须再次安装。

13.2　选择交互组件或 GIF

需要由用户调整参数时，使用 WidgetViewer；需要在演示文稿或网页中播放固定的变动过程时，使用 Animator 导出 GIF（详见第 12 节）。

第 14 节　LaTeX 解析

parse_latex　parse_latex(⟨算式⟩)

将 LaTeX 效用函数解析为模型，返回值可用于 solve()、Canvas 或其他接受模型的 API。

```
from econ_viz import parse_latex

model = parse_latex(r"x^{0.4} y^{0.6}")  # CobbDouglas(alpha=0.4,
beta=0.6)
model = parse_latex(r"\min(2x, 3y)")     # Leontief(a=2.0, b=3.0)
model = parse_latex(r"2x + 3y")          # PerfectSubstitutes(a=2.0,
b=3.0)
```

14.1　支持的形式

支持的输入形式参见表 8。省略的系数或指数默认为 1。

[7] econ-viz 仓库中的 notebook/econ-viz Playground.ipynb 已采用此流程，可在 Jupyter、VS Code 或 Colab 打开；环境中已有 econ-viz 时，会自动跳过安装。

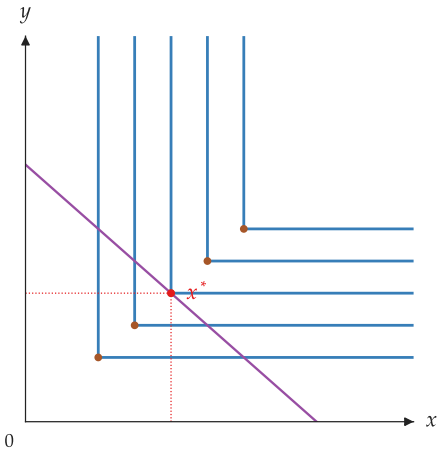

图 30
Leontief 示例。

表 8
支持的效用函数。

类型	格式	示例
Cobb-Douglas	`x^{α} y^{β}` 或 `x^α y^β`	`x^{0.3} y^{0.7}`
完全互补	`\min(ax, by)` 或 `min(ax, by)`	`\min(2x, y)`
完全替代	`ax + by`	`3x + 1.5y`
CES	`(αx^ρ + βy^ρ)^{1/ρ}`	`(0.4x^{-0.5}+0.6y^{-0.5})^{-2}`

可省略开头的 `U =` 或 `u(x, y) =`：

```
U(x,y) = x^{0.5} y^{0.5}
U = x^{0.5} y^{0.5}
x^{0.5} y^{0.5}
```

14.2　错误处理

无法解析时会抛出 `econ_viz.exceptions.ParseError`，错误信息包含支持的格式：

```python
from econ_viz import parse_latex
from econ_viz.exceptions import ParseError

try:
    model = parse_latex(r"x^2 + y^2")
except ParseError as e:
    print(e)
```

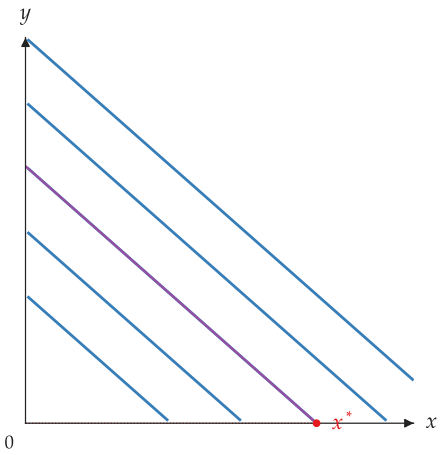

图 31
完全替代输入示例。

```
# Unrecognised LaTeX utility function: 'x^2 + y^2'
# Supported forms:
#   Cobb-Douglas       : x^{alpha} y^{beta}
#   Leontief           : \min(ax, by)
#   Perfect Substitutes: ax + by
#   CES                : (alpha x^{rho} + beta y^{rho})^{1/rho}
```

14.3　在命令行工具中使用

`econ-viz plot --latex` 使用相同的解析器，接受相同的输入格式（详见第 11.2 节）：

```
econ-viz plot --latex "x^{0.4} y^{0.6}" --px 2 --py 3 --income 30 -o
out.png
```

更新纪录

本章列出影响软件包使用的版本变更，不含仅修改文件的版本；各条目按 l3doc 惯例标注于它所描述的功能旁，页码即该功能的实际页码。

v1.12.0 (2026-09-26)

一般：添加无差异曲线主次层级、焦点曲线及数值或序数标签 ····· 2

`Canvas.add_utility`：添加焦点效用层级、次要曲线样式及序数标签 ····· 7

`Theme`：添加次要无差异曲线的颜色、线宽与不透明度设置 ····· 31

v1.11.0 (2026-09-26)

一般：完成主题系统，统一画布、多面板图、需求图、效应分解与 Edgeworth 箱形图的视觉设置 ····· 2

`Canvas`：背景、标签、线条与标记均由主题提供默认值 ····· 7

`Figure`：面板背景、标签、线条与标记完整应用主题 ····· 14

`EdgeworthBox`：箱框、契约曲线、核、禀赋、价格线与 Walras 均衡完整应用主题 ····· 19

`Theme`：完成主题系统，添加背景色、标签比例及所有经济图层的默认样式 ····· 31

`themes`：添加 `paper`、`monochrome`、`presentation` 与 `dark` ····· 31

`Config`：配置文件可选择所有内置主题，并在各种图形间保留相同的视觉语义 ····· 33

`econ-viz plot`：`--theme` 支持所有命令行内置主题 ····· 34

v1.10.1 (2026-09-26)

一般：软件包添加 `py.typed`，并在 CI 运行 Ruff 与 Mypy 检查 ····· 2

v1.10.0 (2026-09-26)

`econ-viz`：Python 3.10 使用 tomli 读取 TOML 配置文件 ····· 3

`Figure`：未指定主题与字体时，使用当前的 `Config` ····· 14

`EdgeworthBox`：未指定主题与字体时，使用当前的 `Config` ····· 19

`Config`：添加 `Config`、`econ-viz.toml`、`econ-viz init` 与 `plot --config` ····· 33

`econ-viz init`：添加配置文件模板命令与 `plot --config` ····· 34

v1.9.0 (2026-09-26)

`Canvas`：坐标轴、原点、标题与其他文字元素接受 `Label` ····· 7

`Stroke`：添加 `opacity` 透明度 ····· 12

`DemandDiagram`：支持 `Legend`、`Marker`、`Label` 与各图层的 `Stroke` ····· 16

`Canvas.add_decomposition`：默认绘制 A、B、C 所在的无差异曲线，并支持曲线标签与 `Legend` ····· 17

横轴下方的效应箭头依 A→B、B→C 方向绘制；效应为零时不绘制箭头 ····· 17

`Haagsma`：添加具有闭式解的 Haagsma 效用模型，用于劣等财与季芬财分析 ····· 24

v1.8.0 (2026-09-26)

`Canvas`：添加 `Axis`、`Marker`、`Label` 与 `Fill` 样式对象 ····· 7

`Stroke`：线条样式统一由 `Stroke` 表示；个别样式参数保留为简写 ····· 12

`Canvas.add_decomposition`：添加 `Effect`、`Marker` 与 `Label` 样式对象 ····· 17

`EdgeworthBox`：标记点、文字与坐标轴接受 `Marker`、`Label` 与 `Axis` ····· 19

v1.7.0 (2026-09-25)

一般：CI 测试 Python 3.10–3.13，强制分支覆盖率，并对可运行的示例做冒烟测试 ····· 2

版本标签须通过测试后才会触发发布 ····· 2

EdgeworthBox：Edgeworth 的内部实现拆成 compute、state 与 plotter 三个模块 ····· 19
models.registry：添加模型注册表（models.registry），供命令行工具创建模型 ····· 20
CobbDouglas：等高线水平的规则集中在 contours.level_policies ····· 20
Canvas.save：Canvas、Figure 与 EdgeworthBox 共享同一个图形导出器（io.exporter） ··· 34
econ-viz：命令行错误改为抛出 CliConfigError，结束处理集中在 cli.main ····· 34

v1.3.0 (2026-04-03)

一般：examples/edgeworth_box.py 涵盖常见的效用函数组合 ····· 2
添加 Edgeworth 测试 ····· 2
EdgeworthBox：添加 EdgeworthBox 与 EquilibriumFocusConfig，支持两人交换经济绘图 ····· 19
契约曲线、核、Walras 均衡叠图，以及聚焦于均衡的无差异曲线 ····· 19

v1.2.3 (2026-03-31)

slutsky_matrix：SlutskyMatrix 会检查对称性、负半定性与齐次性，条件不成立时发出警告 ····· 29
econ-viz models：命令行工具支持 QuasiLinear、StoneGeary 与 Translog ····· 34

v1.2.2 (2026-03-31)

一般：添加 PCC/ICC 示例生成器与测试 ····· 2
PricePath：PCC/ICC 路径默认经过平滑处理，端点略微延长，除非指定否则不显示标记 ··· 15
PCC/ICC 路径改用独立配色，并增加需求图商品空间的留白 ····· 15

v1.2.0 (2026-03-30)

Layout：多面板 Figure 版面与 Layout 枚举 ··· 15
PricePath：添加 PricePath、IncomePath 与 Canvas.add_path()，支持 PCC/ICC 绘图 ·· 15
DemandDiagram：添加 DemandDiagram，联动显示商品空间与 Marshall 需求 ····· 16

v1.1.0 (2026-03-30)

Translog：Translog 模型，以及图例与无差异曲线标签 ····· 20
comparative_statics：comparative_statics 辅助函数 ····· 28
HomogeneityAnalyzer：分析模块中的 HomogeneityAnalyzer 与 ReturnsToScale ··· 29

v1.0.2 (2026-03-29)

econ-viz：删除 NumPy 的版本上限，避免在 Colab 上发生冲突 ····· 3
Canvas：数学式的坐标轴标签不再被重复包装 · 12

v1.0.1 (2026-03-29)

econ-viz：固定 numpy<2，避免 Colab 与本机安装时 ABI 不兼容 ····· 3
放宽 Python 与 NumPy 的版本限制；pytest 固定在 8.x ····· 3

v1.0.0 (2026-03-28)

一般：首次发布 ····· 44

命令索引

W

参考文献

Arrow, K. J., Chenery, H. B., Minhas, B. S., & Solow, R. M. (1961). Capital-labor substitution and economic efficiency. *Review of Economics and Statistics*, *43*(3), 225–250.

Christensen, L. R., Jorgenson, D. W., & Lau, L. J. (1975). Transcendental logarithmic utility functions. *American Economic Review*, *65*(3), 367–383.

Cobb, C. W., & Douglas, P. H. (1928). A theory of production. *American Economic Review*, *18*(1), 139–165.

Edgeworth, F. Y. (1881). *Mathematical psychics*. C. Kegan Paul.

Geary, R. C. (1950). A note on "A constant-utility index of the cost of living". *Review of Economic Studies*, *18*(1), 65–66.

Haagsma, R. (2012). A convenient utility function with Giffen behaviour. *ISRN Economics*, *2012*, Article 608645. https://doi.org/10.5402/2012/608645

Hicks, J. R. (1939). *Value and capital*. Clarendon Press.

Kugler, P., & Andrews, K. (1996). Graphical analysis and the visually impaired in undergraduate economics courses. *The Journal of Economic Education*, *27*(3), 224–228. https://doi.org/10.1080/00220485.1996.10844911

Slutsky, E. (1915). Sulla teoria del bilancio del consumatore. *Giornale degli Economisti e Rivista di Statistica*, *51*, 1–26.

Stone, R. (1954). Linear expenditure systems and demand analysis: An application to the pattern of British demand. *Economic Journal*, *64*(255), 511–527.

thriveth. (2014, January 22). *A color blind/friendly color cycle for Matplotlib line plots* [Source code]. https://gist.github.com/thriveth/8560036