MagicPanel 用户手册（0.1.1）

zyfcode@outlook.com

最后更新：2022 年 4 月 26 日

目录

1 MagicPanel 简介

1.1 功能简介

社会科学或计量经济学研究常常需要用到面板数据库，同一项研究使用的面板数据一般包括多个变量，而这些变量常常来自多个不同的数据源，格式也不尽相同。将这些数据手动合并到一起，费时费力且容易出错。

MagicPanel 库为合并不同格式的面板数据提供了工具。虽然 R 和 Stata 等专业统计软件提供了类似的数据合并选项，但它们一般要求待合并的两个数据库有相同的格式，而 MagicPanel 为不同格式的面板数据提供了可行方案。

1.2 MagicPanel 的面板数据格式规范

表1示例了 MagicPanel 的面板数据格式规范。我们将表格分成四个基本部分：

1. **变量名（index / varname）**：即“表头”，在表1中为黄色的行。
2. **样本名（unit）**：如“国家名”“城市名”等，在表1中为红色的列。
3. **时间点（timepoint）**：如“年份”“季度”等，在表1中为蓝色的列。
4. **变量值（value）**：确定样本名、时间点、变量名，可以在面板数据中确定唯一的一个单元格，该单元格中的内容就是“变量值”，在表1中为白色的单元格。

表 1: 面板数据格式规范示例（数据无实际意义）

城市名	年份	专利申请量	人均 GDP	人才政策
A 市	2011	603	1.768	0
A 市	2012	611	1.895	0
A 市	2013	1614	1.936	1
A 市	2014	1444	1.357	1
A 市	2015	1554	1.872	1
B 市	2011	641	1.367	0
B 市	2012	668	1.452	0
B 市	2013	644	1.432	0
B 市	2014	545	1.118	0
B 市	2015	617	1.317	0
C 市	2011	1256	3.879	0
C 市	2012	1310	4.208	0
C 市	2013	1373	4.378	0
C 市	2014	2161	3.779	1
C 市	2015	2299	4.257	1

表2和表3示例了一些格式不规范的面板数据。MagicPanel 提供了一系列方法，将这些格式各异的数据转化成统一的面板数据规范格式。

表 2: 不规范的面板数据示例一（数据无实际意义）

专利申请量	2011	2012	2013	2014	2015
A 市	603	611	1614	1444	1554
B 市	641	668	644	545	617
C 市	1256	1310	1373	2161	2299

表 3: 不规范的面板数据示例二（数据无实际意义）

城市名	变量名	2011	2012	2013	2014	2015
A 市	专利申请量	603	611	1614	1444	1554
B 市	专利申请量	641	668	644	545	617
C 市	专利申请量	1256	1310	1373	2161	2299
A 市	人均 GDP	1.768	1.895	1.936	1.357	1.872
B 市	人均 GDP	1.367	1.452	1.432	1.118	1.317
C 市	人均 GDP	3.879	4.208	4.378	3.779	4.257
A 市	人才政策	0	0	1	1	1
B 市	人才政策	0	0	0	0	0
C 市	人才政策	0	0	0	1	1

如果您是在浏览 **gen_panel()** 方法详解时跳转到这里的，可点击这里返回：gen_panel。

1.3 QuasiPanel 类简介

```
__init__(self, contents=[[]])
```

QuasiPanel 类用于对“准面板”进行操作。

一些格式不太规范的面板数据，可以先声明为 QuasiPanel 类的一个实例，使用 QuasiPanel 类的一些方法进行标准化之后，再转换为 Panel 类的一个实例，以便进行进一步的操作。

- **contents** 参数指定矩阵的初始内容，数据类型应当是一个 list，且这个 list 的内容应当是若干个长度相同的 list。默认值为一个空矩阵。

1.4 Panel 类简介

```
__init__(self, units: dict={"country": ["Aruba"]}, time: dict={"year": [2001, 2022]})
```

Panel 类用于对“面板”进行操作。

Panel 类的方法大多是为了进一步处理标准格式的面板。这些方法如果应用于格式不规范的数据，往往会造成错误，所以它们是 Panel 类独有的方法。

- **units** 参数指定面板数据第一列的内容。需要传入一个字典，键为表头（如“国家名”），值为包含所有样本名称（如国家名）的列表。默认值为 **{"country": ["Aruba"]}**。

- **units** 参数指定面板数据第二列的内容。需要传入一个字典，键为表头（如“年份”），值为一个二元列表，分别指定了面板数据的起止时间点（如起止年份）。默认值为 **{"year": [2001, 2022]}**。

1.5 链式调用

MagicPanel 推荐使用**链式调用**的方式调用类方法，这可以使代码更清晰可读：

```
pn1 = (
    MagicPanel
    .QuasiPanel()
    .import_from_csv(path=path, filename="gdppc-horizonal-gbk", encoding="gb18030")
    .standardize_hor(index_row=0, unit_col=0, var_col=1, first_time_col=2)
    .gen_panel()
)
```

如果不习惯链式调用，您依旧可以逐行单独调用方法：

```
pn1 = MagicPanel.QuasiPanel()
pn1.import_from_csv(path=path, filename="gdppc-horizonal-gbk", encoding="gb18030")
pn1.standardize_hor(index_row=0, unit_col=0, var_col=1, first_time_col=2)
pn1.gen_panel()
```

这两种写法是等效的。

2 QuasiPanel 类方法详解

2.1 contents

```
contents(self)
```

返回值为准面板的当前内容。返回值的数据类型是 list。

2.2 gen_panel

```
gen_panel(self)
```

将一个 QuasiPanel 声明为 Panel。返回值为 Panel 类的一个实例。

注意：**必须确保 QuasiPanel 的内容已经符合面板数据的规范形式，才可以使用此方法。**参见MagicPanel 的面板数据格式规范。

通过 **standardize_hor**、**standardize_ver** 或 **standardize_policy** 方法生成的 QuasiPanel，原则上都应该符合面板数据的规范，可以放心使用本方法声明为 Panel。

2.3 import_from_csv

```
import_from_csv(self, path: str, filename: str, encoding: str="utf-8", optimize: bool=
    True)
```

从指定的 csv 文件中读取数据，并覆盖矩阵的内容。返回值为更新后的 Matrix 实例。

- **path** 参数指定文件所在的文件夹。
- **filename** 参数指定文件名（不含后缀“.csv”）。
- **encoding** 参数指定 csv 文件的编码，默认为 utf-8 格式。
- **optimize** 参数指定是否要在导入时自动优化格式。默认值为**True**。格式优化包括：
  - 如果 csv 文件的末尾有空行，则会自动删去这些空行。这里的“空行”包括两种情况：长度为 0 的行；每一个单元格长度都为 0 的行。
  - 如果 csv 文件的各行长度不全相等，则会自动用空字符串补齐在较短的行的末尾，使得各行长度相等。
  - 如果 csv 文件的最右侧有空列，则会自动删去这些空列。

2.4 standardize_hor

```
standardize_hor(self, index_row: int=0, unit_col: int=0, var_col=1, var_name="",
    first_time_col: int=2)
```

适用于**时间线横向排开、样本名和变量名纵向排列**的原始数据。显然这种排列方式不符合 MagicPanel 的格式规范。**standardize_hor** 方法将这种格式的数据转变为 MagicPanel 的规范格式。

返回值是更新内容后的一个 QuasiPanel 的实例。

- **index_row** 参数指定索引行（即“表头”）的行号，行号从 0 开始计。默认值为 0。计算时，空行也包括在内。经过 Excel 渲染的 csv 文件，显示的行号可能不是真实的行号；因此，最好是在记事本中打开 csv 文件，查看表头的行号。在图1中，**index_row** 参数应设为 4。
- **unit_col** 参数指定样本名称（如“国家名”“城市名”等）所在列的列号，列号从 0 开始计。默认值为 0。计算时，空列也包括在内。在图1中，**unit_col** 参数应设为 0 或 1。
- **var_col** 参数指定变量名称所在列的列号，列号从 0 开始计。默认值为 1。计算时，空列也包括在内。在图1中，**var_col** 参数应设为 2 或 3。如果原始数据中没有指定变量名的列，则该参数可设为 **None**，相应地，需要指定 **var_name** 参数。
- **var_name** 参数指定变量名。仅当 **var_col** 的值为 **None** 时有效。默认值为空字符串。

	A	B	C	D	E	F	G	H	I
1	Data Source	World Development Indicators							
2									
3	Last Updated Date	2021/12/16							
4									
5	Country Name	Country Code	Indicator Name	Indicator Code	1960	1961	1962	1963	1964
6	Aruba	ABW	Taxes on income, profits and capital gains (% of revenue)	GC.TAX.YPKG.RV.ZS					
7	Africa Eastern and Southern	AFE	Taxes on income, profits and capital gains (% of revenue)	GC.TAX.YPKG.RV.ZS					
8	Afghanistan	AFG	Taxes on income, profits and capital gains (% of revenue)	GC.TAX.YPKG.RV.ZS					
9	Africa Western and Central	AFW	Taxes on income, profits and capital gains (% of revenue)	GC.TAX.YPKG.RV.ZS					
10	Angola	AGO	Taxes on income, profits and capital gains (% of revenue)	GC.TAX.YPKG.RV.ZS					
11	Albania	ALB	Taxes on income, profits and capital gains (% of revenue)	GC.TAX.YPKG.RV.ZS					
12	Andorra	AND	Taxes on income, profits and capital gains (% of revenue)	GC.TAX.YPKG.RV.ZS					
13	Arab World	ARB	Taxes on income, profits and capital gains (% of revenue)	GC.TAX.YPKG.RV.ZS					
14	United Arab Emirates	ARE	Taxes on income, profits and capital gains (% of revenue)	GC.TAX.YPKG.RV.ZS					
15	Argentina	ARG	Taxes on income, profits and capital gains (% of revenue)	GC.TAX.YPKG.RV.ZS					
16	Armenia	ARM	Taxes on income, profits and capital gains (% of revenue)	GC.TAX.YPKG.RV.ZS					

图 1: standardize_hor 方法应用示例

	A	B
1	省份	是否实验组
2	A省	1
3	B省	1
4	C省	0
5	D省	1
6	E省	0
7	F省	0
8	G省	0
9	H省	1

图 2: 同时施行的政策

- **first_time_col** 参数指定起始年份的数据所在列的列号，列号从 0 开始计。默认值为 2。计算时，空列也包括在内。在图1中，**first_time_col** 参数应设为 4。

2.5 standardize_policy

```
standardize_policy(self, start, end, mode: str="sync", varname: str="", treat_time=0)
```

这个方法适用于处理一类特殊的数据，即**政策变量**。

这类变量往往连续多年取同样的值，并在个别时间节点发生变化。因此，原始数据常常仅指出了政策变化的时间节点，而没有标注每一年的变量值。**standardize_policy** 方法可以将这类原始数据转化成规范的面板数据形式。

本方法可以处理三类原始数据：

	A	B
1	省份	政策施行年份
2	A省	2005
3	B省	
4	C省	2004
5	D省	2017
6	E省	
7	F省	2009
8	G省	
9	H省	2013

图 3: 不同时施行的政策

	A	B	C	D
1	**省份**	**年份**	**人才资助**	**医保起付线**
2	A省	2010	10000	600
3	A省	2013	30000	300
4	B省	2010	0	200
5	B省	2017	10000	200
6	C省	2010	0	0
7	D省	2010	30000	1000
8	D省	2016	30000	600
9	D省	2018	50000	0
10	E省	2010	20000	500
11	F省	2010	20000	100
12	F省	2014	30000	100
13	F省	2019	50000	100
14	G省	2010	30000	300
15	H省	2010	40000	300

图 4: 多次变化的政策

1. **同时施行的**（synchronic）政策变量。即：控制组始终不施行政策，实验组的所有样本在同一时间施行政策。此时，传入的原始数据只需要如图2所示即可。处理这类数据时，需要在 `standardize_policy()` 的参数中额外指定**政策施行年份**。

2. **不同时施行的**（diachronic）政策变量。即：控制组始终不施行政策，实验组的样本施行政策有时间先后之别。此时，传入的原始数据只需要如图3所示即可（控制组的“政策施行年份”一列值为空）。

3. **复杂的**（complex）政策。该模式允许原始数据中有多个变量，并且变量的值可以多次变化。此时，传入的原始数据需要如图4所示（**时间必须按先后顺序排列**），明确指定每次变化的时间点，以及变化后政策变量的值。

- `start` 参数指定面板数据从哪一个时间点开始记录。
- `end` 参数指定面板数据在哪一个时间点结束记录。
- `mode` 参数指定处理模式，分别对应上述的三类原始数据。值为 `"sync"` 时，处理同时施行的（synchronic）政策；值为 `"diac"` 时，处理不同时施行的（diachronic）政策；值为 `"complex"` 时，处理复杂的（complex）政策。默认值为 `"sync"`。
- `varname` 参数指定政策变量的名称。仅在 `mode="sync"` 或 `mode="diac"` 时适用。
- `treat_time` 参数指定政策（同时）施行的时间点。默认值为 0。仅在 `mode="sync"` 时适用。

注意：在所有模式中，变量值的变化在**政策施行当年**即发生。

2.6 standardize_ver

```
standardize_ver(self, index_row: int=0, unit_col: int=0, time_col: int=1, first_var_col:
    int=2)
```

适用于**样本名和时间线纵向排列、变量名横向排列**的原始数据。这种排列方式基本符合 MagicPanel 的格式规范。`standardize_ver` 方法对这种格式的数据进行一些微调，如删除多余的行和列等。

返回值是更新内容后的一个 QuasiPanel 的实例。

- `index_row` 参数指定索引行（即“表头”）的行号，行号从 0 开始计。默认值为 0。计算时，空行也包括在内。经过 Excel 渲染的 csv 文件，显示的行号可能不是真实的行号；因此，最好是在记事本中打开 csv 文件，查看表头的行号。
- `unit_col` 参数指定样本名称（如“国家名”“城市名”等）所在列的列号，列号从 0 开始计。默认值为 0。计算时，空列也包括在内。
- `time_col` 参数指定变量名称所在列的列号，列号从 0 开始计。默认值为 1。计算时，空列也包括在内。
- `first_var_col` 参数指定第一个变量所在列的列号，列号从 0 开始计。默认值为 2。计算时，空列也包括在内。

3 Panel 类方法详解

3.1 absorb

```
absorb(self, new_panel: "Panel", mode: str="new", mapping: dict={})
```

将一个新的面板数据合并到原来的面板中。

注意：合并时以原来的面板数据为基础。合并后保持原面板数据的样本顺序；**当新面板中有原面板没有的样本（如原面板时间只到 2010 年为止，但新面板存在 2011 年的数据）时，新面板中多出来的样本会被舍弃**。

`absorb` 方法提供两种模式：

- **新数据（new）模式**，即合并一个新的数据库进来，添加为新的变量。
- **补充数据（complementary）模式**，即用新的数据填补原有数据中的缺失值（如果不是缺失值，则不会填充，而是保留原有数据）。

应慎用“补充数据模式”，因为不同来源的数据很可能有单位不一致、统计口径有差异等问题，贸然合并会影响数据质量。

- `new_panel` 参数指定新的面板，需要传入一个 Panel 类的实例。
- `mode` 参数指定合并的模式：`"new"` 为新数据模式，`"complementary"` 为补充数据模式。默认值为 `"new"`。

- **mapping** 参数指定补充数据模式中变量名的对应情况。对于新数据来说，只有在 **mapping** 中指定的变量会用于数据合并，其他的变量将被忽略。**在补充数据模式下，必须一一指明变量名的对应情况，即使原变量名和对应的变量名相同**。传入数据的格式为：**{"原变量名A": "对应变量名A", "原变量名B": "对应变量名B", ...}**。

3.2 add_var

```
add_var(self, varname: str)
```

给数据库添加一个新变量，变量的初始值为空字符串。

- **varname** 参数指定新变量的名称。

3.3 change

```
change(self, unit: str, timepoint, var: str, value: str)
```

通过指定样本名、时间点和变量名确定单元格，并改写该单元格的值。
如果找不到对应的单元格，则不对数据作任何改动。

- **unit** 参数指定样本名（如具体国家名）。
- **timepoint** 参数指定时间点（如具体年份）。
- **var** 参数指定变量名。
- **value** 参数指定为该单元格赋的新值。

3.4 contents

```
contents(self)
```

返回值为面板的当前内容。返回值的数据类型是 list。

3.5 extract

```
extract(self, varlist: list)
```

通过指定变量，提取当前数据集的子数据集。返回值为 Panel 类的一个实例。
如果指定的变量中有原数据集中不存在的变量，则忽略之。

- **varlist** 参数指定需要提取的变量名（表的前两列无需指定）。

3.6 export

```
export(self, path: str, filename: str, encoding: str="utf-8")
```

将当前面板数据导出为 csv 文件。

- **path** 参数指定导出文件所在的文件夹。
- **filename** 参数指定文件名（不含后缀“.csv”）。
- **encoding** 参数指定 csv 文件的编码，默认为 utf-8 格式。如果使用 Excel 打开导出文件时乱码，可以尝试声明 **encoding="gb18030"**。

3.7 get

```
get(self, unit: str, timepoint, var: str)
```

通过指定样本名、时间点和变量名，获取面板数据中的一个单元格的值。
返回值为一个字符串，即要查找的单元格的值。
如果找不到对应的单元格，则返回**None**。

- **unit** 参数指定样本名（如具体国家名）。
- **timepoint** 参数指定时间点（如具体年份）。
- **var** 参数指定变量名。

3.8 get_syn_from_csv

```
get_syn_from_csv(path: str, filename: str, encoding: str="utf-8")
```

从 csv 文件中获取同义词词典。返回值为一个字典。
关于该字典的形式和用途，参见paraphrase。

- **path** 参数指定 csv 文件所在的文件夹。
- **filename** 参数指定文件名（不含后缀“.csv”）。
- **encoding** 参数指定 csv 文件的编码，默认为 utf-8 格式。

3.9 get_var_col

```
get_var_col(self, varname: str)
```

返回值为变量名对应的列号。
如果数据中有重复的变量名（这种情况应尽量避免），则返回第一个匹配到的列号。如果找不到该变量，则返回 **None**。

- **varname** 参数指定要查找的变量名。

3.10 locate

```
locate(self, unit: str, timepoint, var: str)
```

通过指定样本名、时间点和变量名，锁定面板数据中的一个单元格。

返回值为一个元组，包含要查找的单元格的行号和列号（行号和列号从 0 开始计）。

如果找不到对应的单元格，则返回`(None, None)`。

- `unit` 参数指定样本名（如具体国家名）。
- `timepoint` 参数指定时间点（如具体年份）。
- `var` 参数指定变量名。

3.11 paraphrase

```
paraphrase(self, synonyms: dict)
```

该方法可以解决一个常见的问题，即不同的数据库对同一个样本采用不同的称呼方式。如“Cambodia”和“Kampuchea”都是指柬埔寨、“Cote d'Ivoire”和“Ivory Coast”都是指科特迪瓦等。

- `synonyms` 参数指定一个同义词词典。这个词典需要逐个指明每个样本的别名对应的标准名称。传入数据的格式为：`{"别名1": "标准名1", "别名2": "标准名2", ...}`。

3.12 sort

```
sort(self, varlist: list=[], reverse=False)
```

将面板数据重新排序。先按照用户设置排序，对于按照用户设置无法分出顺序的，按照样本名称和时间升序排列。

- `varlist` 参数指定排序依据的变量。放在前面的变量优先级更高。默认值为空列表。当 `varlist` 参数为默认值时，即对原始数据按照样本名称和时间升序排列。
- `reverse` 参数指定是否降序排列。默认值为 `False`。

在 Python 中，字符串排序是按照字典顺序，因此会出现 `"345" > "1234"` 的情况。为了避免这种情况，对于所有数字型的变量，MagicPanel 会将其自动转成浮点数（float）再进行排序。

3.13 time

```
time(self)
```

返回值为面板包含的所有时间点（如所有年份），从小到大排序。

3.14 units

```
units(self)
```

返回值为面板包含的所有样本名（如所有国家名），按字母顺序排序。

3.15 variables

```
variables(self)
```

返回值为面板包含的所有变量名（即“表头”的各项），按原始顺序排序。

4 示例代码

您可以点击此处下载下列示例的源代码。

```
# 以下为MagicPanel库的示例代码。
# 访问 https://github.com/pku-zyf/MagicPanel/，可在“example-data”文件夹获取本示例代码
    用到的原始数据。
# 将“example-data”文件夹复制到本示例代码所在的路径，即可运行。
# 所有数据仅供示例使用，无任何实际意义。

import MagicPanel
from os.path import dirname, join, realpath

def main():
    # 获取原始数据所在路径。
    path = join(dirname(realpath(__file__)), "example-data")
    # 导入“人均GDP”数据（横向）。
    pn1 = (
        MagicPanel
        .QuasiPanel()
        .import_from_csv(path=path, filename="gdppc-horizonal-gbk", encoding="gb18030")
        .standardize_hor(index_row=0, unit_col=0, var_col=1, first_time_col=2)
        .gen_panel()
    )
    # 导入“人口”数据（纵向）。
    pn2 = (
        MagicPanel
        .QuasiPanel()
        .import_from_csv(path=path, filename="pop-vertical-utf8", encoding="utf8")
        .standardize_ver(index_row=2, unit_col=0, time_col=1, first_var_col=2)
        .gen_panel()
    )
```

```
    # 导入“人才政策”数据（虚拟变量）。
    pn3 = (
        MagicPanel
        .QuasiPanel()
        .import_from_csv(path=path, filename="policy-big5", encoding="big5")
        .standardize_policy(varname="人才政策", start=2011, end=2020, mode="diac")
        .gen_panel()
    )
    # 导入“专利数量”数据（横向）。
    pn4 = (
        MagicPanel
        .QuasiPanel()
        .import_from_csv(path=path, filename="patent-horizonal-gbk", encoding="gbk")
        .standardize_hor(index_row=1, unit_col=0, var_col=None, var_name="专利数量",
            first_time_col=1)
        .gen_panel()
    )
    # 导入“专利数量”补充数据（横向）。
    pn5 = (
        MagicPanel
        .QuasiPanel()
        .import_from_csv(path=path, filename="patent-horizonal-sup-gbk", encoding="gbk")
        .standardize_hor(index_row=1, unit_col=0, var_col=None, var_name="专利数量",
            first_time_col=1)
        .gen_panel()
    )
    # 合并数据。
    pn = (
        pn1
        .absorb(pn2)
        .absorb(pn3)
        .absorb(pn4)
        .absorb(pn5, mode="complementary", mapping={"专利数量": "专利数量"})
        .sort()
    )
    # 导出数据。导出为gb18030格式，可在Windows系统的Excel中直接打开。
    pn.export(path=path, filename="result", encoding="gb18030")

if __name__ == "__main__":
    main()
```