1、基础知识：array\_a.ndim 维度

array\_a.shape 维度，几行几列

array\_a.reshape(变形，3行4列可以转化为2行6列或者其他行列相乘为12 的数组）

array\_a.size 获取数组中元素的个数。

array\_a.itemsize 数组中每个元素占的内存大小

2、深拷贝和浅拷贝

浅拷贝，a1.view（）就是只拷贝栈区，深拷贝，a1.copy()既拷贝栈区也拷贝堆区

3、np.save（“名字”，数组) 保存 np.load（“名字”，数组）加载。

不可以设置header，但是可以操作三维以上的数组。

4、np.random.seed()种子，固定随机数

np.random.rand()产生0-1之间一个随机数。

np.random.randn()产生标准正态分布的随机数

.choice()随机获取几个数，对数组中随机取样。

.shuffle() 直接跟新数组。

5 np.argsort()返回排序的下标值。

6 np.apply\_along\_axis:沿着某个轴执行指定的代码。

np.linspace:用来将指定的区间内的值平均分成多少份

np.unique：返回数组中的唯一值

series:

# 核心：series相比于ndarray，是一个自带索引index的数组 → 一维数组 + 对应索引

# 所以当只看series的值的时候，就是一个ndarray

# series和ndarray较相似，索引切片功能差别不大

# series和dict相比，series更像一个有顺序的字典（dict本身不存在顺序），其索引原理与字典相似（一个用key，一个用index）

# .head()查看头部数据

# .tail()查看尾部数据

# 默认查看5条

# fill\_value参数：填充缺失值的值

# Dataframe是一个表格型的数据结构，“带有标签的二维数组”。Dataframe带有index（行标签）和columns（列标签）

parse('5/1/2014', dayfirst = True),'\n', # 国际通用格式中，日在月之前，可以通过dayfirst来设置

# pd.to\_datetime()：如果是单个时间数据，转换成pandas的时刻数据，数据类型为Timestame

print(t3)# 多个时间数据，将会转换为pandas的DatetimeIndex

date3 = ['2017-2-1','2017-2-2','2017-2-3','hello world!','2017-2-5','2017-2-6']

t3 = pd.to\_datetime(date3, errors = 'ignore')

print(t3,type(t3))

# 当一组时间序列中夹杂其他格式数据，可用errors参数返回

# errors = 'ignore':不可解析时返回原始输入，这里就是直接生成一般数组

# errors = 'coerce':不可扩展，缺失值返回NaT（Not a Time），结果认为DatetimeIndex

rng1 = pd.date\_range('1/1/2017','1/10/2017', normalize=True)

rng2 = pd.date\_range(start = '1/1/2017', periods = 10)

rng3 = pd.date\_range(end = '1/30/2017 15:00:00', periods = 10) # 增加了时、分、秒

print(rng1,type(rng1))

print(rng2)

print(rng3)

print('-------')

# 直接生成DatetimeIndex

# pd.date\_range(start=None, end=None, periods=None, freq='D', tz=None, normalize=False, name=None, closed=None, \*\*kwargs)

# start：开始时间

# end：结束时间

# periods：偏移量

# freq：频率，默认天，pd.date\_range()默认频率为日历日，pd.bdate\_range()默认频率为工作日

# tz：时区

# M：每月最后一个日历日

# Q-月：指定月为季度末，每个季度末最后一月的最后一个日历日

# A-月：每年指定月份的最后一个日历日

# 月缩写：JAN/FEB/MAR/APR/MAY/JUN/JUL/AUG/SEP/OCT/NOV/DEC

# 所以Q-月只有三种情况：1-4-7-10,2-5-8-11,3-6-9-12

# BM：每月最后一个工作日

# BQ-月：指定月为季度末，每个季度末最后一月的最后一个工作日

# BA-月：每年指定月份的最后一个工作日

# M：每月第一个日历日

# Q-月：指定月为季度末，每个季度末最后一月的第一个日历日

# A-月：每年指定月份的第一个日历日

# BM：每月第一个工作日

# BQ-月：指定月为季度末，每个季度末最后一月的第一个工作日

# BA-月：每年指定月份的第一个工作日

# asfreq：时期频率转换

ts = pd.Series(np.random.rand(4),

index = pd.date\_range('20170101','20170104'))

print(ts)

print(ts.asfreq('4H',method = 'ffill'))

# 改变频率，这里是D改为4H

# method：插值模式，None不插值，ffill用之前值填充，bfill用之后值填充

# 重采样：.resample()

# 创建一个以天为频率的TimeSeries，重采样为按2天为频率

rng = pd.date\_range('20170101', periods = 12)

ts = pd.Series(np.arange(12), index = rng)

print(ts)

ts\_re = ts.resample('5D')

ts\_re2 = ts.resample('5D').sum()

print(ts\_re, type(ts\_re))

print(ts\_re2, type(ts\_re2))

print('-----')

# ts.resample('5D')：得到一个重采样构建器，频率改为5天

# ts.resample('5D').sum():得到一个新的聚合后的Series，聚合方式为求和

# freq：重采样频率 → ts.resample('5D')

# .sum()：聚合方法

升采样

print(ts.resample('15T').asfreq())

print(ts.resample('15T').ffill())

print(ts.resample('15T').bfill())

# 低频转高频，主要是如何插值

# .asfreq()：不做填充，返回Nan

# .ffill()：向上填充

# .bfill()：向下填充

plt.pie(x, explode=None, labels=None, colors=None,autopct=None, pctdistance=0.6, shadow=False,labeldistance=1.1, startangle=None,radius=None, counterclock=True, wedgeprops=None,textprops=None, center=(0, 0), frame=False)

x：指定绘图的数据；

explode：指定饼图某些部分的突出显示，即呈现爆炸式；

labels：为饼图添加标签说明，类似于图例说明；

colors：指定饼图的填充色；

autopct：自动添加百分比显示，可以采用格式化的方法显示；

pctdistance：设置百分比标签与圆心的距离；

shadow：是否添加饼图的阴影效果；

labeldistance：设置各扇形标签（图例）与圆心的距离；

startangle：设置饼图的初始摆放角度；

radius：设置饼图的半径大小；

counterclock：是否让饼图按逆时针顺序呈现；

wedgeprops：设置饼图内外边界的属性，如边界线的粗细、颜色等；

textprops：设置饼图中文本的属性，如字体大小、颜色等；

center：指定饼图的中心点位置，默认为原点

frame：是否要显示饼图背后的图框，如果设置为True的话，需要同时控制图框x轴、y轴的范围和饼图的中心位置；

plt.boxplot(x, notch=None, sym=None, vert=None,whis=None, positions=None, widths=None,patch\_artist=None, meanline=None, showmeans=None,showcaps=None, showbox=None, showfliers=None,boxprops=None, labels=None, flierprops=None,medianprops=None, meanprops=None,capprops=None, whiskerprops=None)

x：指定要绘制箱线图的数据；

notch：是否是凹口的形式展现箱线图，默认非凹口；

sym：指定异常点的形状，默认为+号显示；

vert：是否需要将箱线图垂直摆放，默认垂直摆放；

whis：指定上下须与上下四分位的距离，默认为1.5倍的四分位差；

positions：指定箱线图的位置，默认为[0,1,2…]；

widths：指定箱线图的宽度，默认为0.5；

patch\_artist：是否填充箱体的颜色；

meanline：是否用线的形式表示均值，默认用点来表示；

showmeans：是否显示均值，默认不显示；

showcaps：是否显示箱线图顶端和末端的两条线，默认显示；

showbox：是否显示箱线图的箱体，默认显示；

showfliers：是否显示异常值，默认显示；

boxprops：设置箱体的属性，如边框色，填充色等；

labels：为箱线图添加标签，类似于图例的作用；

filerprops：设置异常值的属性，如异常点的形状、大小、填充色等；

medianprops：设置中位数的属性，如线的类型、粗细等；

meanprops：设置均值的属性，如点的大小、颜色等；

capprops：设置箱线图顶端和末端线条的属性，如颜色、粗细等；

whiskerprops：设置须的属性，如颜色、粗细、线的类型等；

# 检查年龄是否有缺失

any(titanic.Age.isnull())

plt.hist(x, bins=10, range=None, normed=False,weights=None, cumulative=False, bottom=None,histtype='bar', align='mid', orientation='vertical',rwidth=None, log=False, color=None,label=None, stacked=False)

x：指定要绘制直方图的数据；

bins：指定直方图条形的个数；

range：指定直方图数据的上下界，默认包含绘图数据的最大值和最小值；

normed：是否将直方图的频数转换成频率；

weights：该参数可为每一个数据点设置权重；

cumulative：是否需要计算累计频数或频率；

bottom：可以为直方图的每个条形添加基准线，默认为0；

histtype：指定直方图的类型，默认为bar，除此还有’barstacked’, ‘step’, ‘stepfilled’；

align：设置条形边界值的对其方式，默认为mid，除此还有’left’和’right’；

orientation：设置直方图的摆放方向，默认为垂直方向；

rwidth：设置直方图条形宽度的百分比；

log：是否需要对绘图数据进行log变换；

color：设置直方图的填充色；

label：设置直方图的标签，可通过legend展示其图例；

stacked：当有多个数据时，是否需要将直方图呈堆叠摆放，默认水平摆放；

# 生成正态曲线的数据

x1 = np.linspace(titanic.Age.min(), titanic.Age.max(), 1000)

normal = mlab.normpdf(x1, titanic.Age.mean(), titanic.Age.std())

# 绘制正态分布曲线

line1, = plt.plot(x1,normal,'r-', linewidth = 2)

# 生成核密度曲线的数据

kde = mlab.GaussianKDE(titanic.Age)

x2 = np.linspace(titanic.Age.min(), titanic.Age.max(), 1000)

# 绘制

line2, = plt.plot(x2,kde(x2),'g-', linewidth = 2)

# 去除图形顶部边界和右边界的刻度

plt.tick\_params(top='off', right='off')

plt.hist(x,y,linestyle,

linewidth,color,marker,

markersize,markeredgecolor,

markerfactcolor,label,alpha)

x：指定折线图的x轴数据；

y：指定折线图的y轴数据；

linestyle：指定折线的类型，可以是实线、虚线、点虚线、点点线等，默认文实线；

linewidth：指定折线的宽度

marker：可以为折线图添加点，该参数是设置点的形状；

markersize：设置点的大小；

markeredgecolor：设置点的边框色；

markerfactcolor：设置点的填充色；

label：为折线图添加标签，类似于图例的作用；

利用魔法函数的，可以不用plt.show()函数来展示图形。

% matplotlib qt5 #魔法函数，弹出matplotlib控制台

% matplotlib notebook #魔法函数，弹出可交互的matplotlib窗口

% matplotlib inline #魔法函数，嵌入图表

lable 标签,lim 边界。

plt.grid(True, linestyle = "--",color = "gray", linewidth = "0.5",axis = 'x')

# 显示网格

# linestyle：线型

# color：颜色

# linewidth：宽度

# axis：x，y，both，显示x/y/两者的格网

matplotlib.rcParams['xtick.direction'] = 'out'

matplotlib.rcParams['ytick.direction'] = 'inout'

# 设置刻度的方向，in,out,inout

# linestyle参数

plt.plot([i\*\*2 for i in range(100)],

linestyle = '-.')

# '-' solid line style

# '--' dashed line style

# '-.' dash-dot line style

# ':' dotted line style

# marker参数

s = pd.Series(np.random.randn(100).cumsum())

s.plot(linestyle = '--',

marker = '.')

# '.' point marker

# ',' pixel marker

# 'o' circle marker

# 'v' triangle\_down marker

# '^' triangle\_up marker

# '<' triangle\_left marker

# '>' triangle\_right marker

# '1' tri\_down marker

# '2' tri\_up marker

# '3' tri\_left marker

# '4' tri\_right marker

# 's' square marker

# 'p' pentagon marker

# '\*' star marker

# 'h' hexagon1 marker

# 'H' hexagon2 marker

# '+' plus marker

# 'x' x marker

# 'D' diamond marker

# 'd' thin\_diamond marker

# '|' vline marker

# '\_' hline marker

# 刻度

from matplotlib.ticker import MultipleLocator, FormatStrFormatter

t = np.arange(0.0, 100.0, 1)

s = np.sin(0.1\*np.pi\*t)\*np.exp(-t\*0.01)

ax = plt.subplot(111) #注意:一般都在ax中设置,不再plot中设置

plt.plot(t,s,'--\*')

plt.grid(True, linestyle = "--",color = "gray", linewidth = "0.5",axis = 'both')

# 网格

#plt.legend() # 图例

xmajorLocator = MultipleLocator(10) # 将x主刻度标签设置为10的倍数

xmajorFormatter = FormatStrFormatter('%.0f') # 设置x轴标签文本的格式

xminorLocator = MultipleLocator(5) # 将x轴次刻度标签设置为5的倍数

ymajorLocator = MultipleLocator(0.5) # 将y轴主刻度标签设置为0.5的倍数

ymajorFormatter = FormatStrFormatter('%.1f') # 设置y轴标签文本的格式

yminorLocator = MultipleLocator(0.1) # 将此y轴次刻度标签设置为0.1的倍数

ax.xaxis.set\_major\_locator(xmajorLocator) # 设置x轴主刻度

ax.xaxis.set\_major\_formatter(xmajorFormatter) # 设置x轴标签文本格式

ax.xaxis.set\_minor\_locator(xminorLocator) # 设置x轴次刻度

ax.yaxis.set\_major\_locator(ymajorLocator) # 设置y轴主刻度

ax.yaxis.set\_major\_formatter(ymajorFormatter) # 设置y轴标签文本格式

ax.yaxis.set\_minor\_locator(yminorLocator) # 设置y轴次刻度

ax.xaxis.grid(True, which='both') #x坐标轴的网格使用主刻度

ax.yaxis.grid(True, which='minor') #y坐标轴的网格使用次刻度

# which：格网显示

#删除坐标轴的刻度显示

#ax.yaxis.set\_major\_locator(plt.NullLocator())

#ax.xaxis.set\_major\_formatter(plt.NullFormatter())

# dpi是分辨率

# bbox\_inches：图表需要保存的部分。如果设置为‘tight’，则尝试剪除图表周围的空白部分。

# facecolor，edgecolor： 图像的背景色，默认为‘w’（白色）

# wspace,hspace：用于控制宽度和高度的百分比，比如subplot之间的间距

# Series.plot()：series的index为横坐标，value为纵坐标

# kind → line,bar,barh...（折线图，柱状图，柱状图-横...）

# label → 图例标签，Dataframe格式以列名为label

# style → 风格字符串，这里包括了linestyle（-），marker（.），color（g）

# color → 颜色，有color指定时候，以color颜色为准

# alpha → 透明度，0-1

# use\_index → 将索引用为刻度标签，默认为True

# rot → 旋转刻度标签，0-360

# grid → 显示网格，一般直接用plt.grid

# xlim,ylim → x,y轴界限

# xticks,yticks → x,y轴刻度值

# figsize → 图像大小

# title → 图名

# legend → 是否显示图例，一般直接用plt.legend()

plt.pie 饼图

plot.area,面积图

plt.scatter散点图

s.hist直方图

plt.figure极坐标图

# 雷达图1 - 极坐标的折线图/填图 - plt.plot()

plt.axis('equal') # 保证长宽相等

s.hist(bins = 20,##直方图

histtype = 'bar',

align = 'mid',

orientation = 'vertical',

alpha=0.5,

normed =True)

# bin：箱子的宽度

# normed 标准化

# histtype 风格，bar，barstacked，step，stepfilled

# orientation 水平还是垂直{‘horizontal’, ‘vertical’}

# align : {‘left’, ‘mid’, ‘right’}, optional(对齐方式)

# stacked：是否堆叠

# diagonal：({‘hist’, ‘kde’})，必须且只能在{‘hist’, ‘kde’}中选择1个 → 每个指标的频率图

# range\_padding：(float, 可选)，图像在x轴、y轴原点附近的留白(padding)，该值越大，留白距离越大，图像远离坐标原点

箱线图：

④ 内限 → T形的盒须就是内限，最大值区间Q3+1.5IQR,最小值区间Q1-1.5IQR （IQR=Q3-Q1）

⑤ 外限 → T形的盒须就是内限，最大值区间Q3+3IQR,最小值区间Q1-3IQR （IQR=Q3-Q1）

# whiskers → 分位数与error bar横线之间竖线的颜色

# medians → 中位数线颜色

# caps → error bar横线颜色

def color\_neg\_red(val):

if val < 0:

color = 'red'

else:

color = 'black'

return('color:%s' % color)

df.style.applymap(color\_neg\_red)

# 创建样式方法，使得小于0的数变成红色

# 通过pd.IndexSlice[]调用切片

format表格的表示方式都可以设置

Stripplot的本质就是把数据集中具有quantitative属性的变量按照类别去做散点图(Scatterplot)。

Swarmplot

另一个解决stripplot中点重叠的办法就是绘制swarmplot,它的本质就是用通过算法，在类别坐标轴的方向上去‘延展’绘制这些原本重合的点。

rename 重命名 ，

.unique() 输出独一的数据

from scipy import stats, integrate

data['Date'] = pd.to\_datetime(data['Date'])

data['Year'] = data['Date'].dt.year #转化成日期类型

data['Month'] = data['Date'].dt.month

test = data.groupby([data['Year'],data['Month']],as\_index=False).count()#组合数据

#做线性回归拟合

plt.figure(figsize=(12,6))

plt.subplot(121)

sns.regplot(x="Year", y="ID", data=temp,order=1) # default by 1

plt.ylabel(' ')

plt.title('Regression fit of earthquake records by year,order = 1')

plt.subplot(122)

sns.residplot(x="Year", y="ID", data=temp)

plt.ylabel(' ')

plt.title('Residual plot when using a simplt regression model,order=1')#order = 2 二阶拟合

plt.show()

**pyecharts**

条形图

WordCloud(width=1000,height=620)

width=1000 宽度

height=620 高度

wordcloud.add('', name, values, word\_size\_range=[20,100])

word\_size\_range=[20,100] 字体大小范围

shape='diamond' # 不倾斜形式

bar.add('学生A', labels, cj1, is\_stack=False, is\_more\_utils=True)

is\_stack = False不堆叠排列; True 堆叠排列

is\_more\_utils=True显示更多工具;False 不显示更多工具

mark\_point=["average", "max", "min"];标记平均值、最大、最小值

mark\_line=["max"]标记一条虚线来表示最大值所在区域

is\_convert=True横向柱形图; False 纵向柱形图

bar.render()没设置路径保存则不做任何保存处理

饼图

pie.add('', labels, cj1, is\_label\_show=True, legend\_orient='vertical',legend\_pos='left', radius=[40,75])

s\_label\_show = True 代表显示标签

label\_text\_color=None # 设置标签字体颜色

legend\_orient='vertical' 代表图例垂直排列

legend\_pos='left/30%' 代表图例的位置

center=[10,85], # 调整子图的位置，[调整列与列的间隙，调整行与行间隙]，如[10, 30]

radius=[40,75] 设置内径大小、外径大小，使其变成圆环图

线性图

line = Line('3、线形图', '副标题', width=1000, height=400)

line.add('正数', labels, L1, mark\_point=['min','max'], mark\_line=['average'], is\_smooth=True)

mark\_point 标记要点

mark\_point\_symbol='arrow' 标记要点的符号形状（箭头型）

mark\_point\_symbolsize=30 标记要点的符号形状的大小

mark\_point\_textcolor='#fff' 标记要点的字体颜色

symbol=None 符号形状

mark\_line 标记虚线线条，average 表示平均值

is\_smooth 绘制平滑曲线

is\_step 是否设置为台阶型线形图

is\_fill=True 是否填充

line\_opacity=0.2 线的不透明度

area\_opacity=0.4 填充区域的不透明度

散点图

scatter = Scatter('4、散点图', '副标题')

scatter.add('A', s1, s2)

es =EffectScatter("动态散点图")

es.add("", es1, es2, symbol\_size=18)

symbol\_size# 符号大小

effect\_scale# 缩放大小

effect\_period# 动态周期，越大越慢

symbol="triangle", # 什么类型符号，有 "pin/rect/roundRect/diamond/arrow/triangle"

effect\_brushtype='fill' # 是否填充动态缩放区

仪表盘

from pyecharts import Gauge

gauge=Gauge('5、小车速度表--')

gauge.add('速度指标','km/h',80)

Funnel漏斗图

funnel = Funnel('6、学生成绩分布漏斗图')

funnel.add('成绩', labels, cj1, is\_label\_show=True, label\_pos='inside')

is\_label\_show=True 代表显示标签

label\_pos='inside' 代表标签在在内部

label\_text\_color='#fff' 代表标签文本的颜色，这里是白色

Geo地图

Geo('8、地图', '副标题', title\_color='#fff', title\_pos='center', width=1200, height=600, background\_color='#404a59')

title\_color 标题颜色

title\_pos 标题位置

width 图像宽度

height 图像高度

background\_color 背景颜色

geo.add('', attr, value, visual\_range=[0, 200], visual\_text\_color='#fff',

symbol\_size=15, is\_visualmap=True, is\_piecewise=True, visual\_split\_number=6)

visual\_range 可见区域范围

visual\_text\_color 可见区域文字颜色

symbol\_size 符号大小

is\_visualmap 是否展示地图

is\_piecewise 是否分段

visual\_split\_number 视觉分割数

overlap = Overlap()

overlap.add(line)

overlap.add(scatter)组合图

Timeline

关系图：节点，节点之间的联系 graph