代码建议 二 （截止到章节 2.1.3）

1. 对章节名称的建议
   2.1.1 和 2.1.2 的名称（2.1.1 数据 POI 与描述性统计和正态分布+ 2.1.2 数据 OSM 与核密度估计），可以改成 POI 数据，OSM 数据

2. 2.1.1 中标准计分特征的描述和举例的勘误

原文为：

```
标准计分的特征：

1. 无论作为变量的满分为几分，其标准计分的平均数势必为 0， 而其标准差势必为 1；
2. 无论作为变量的单位是什么，其标准计分的平均数势必为 0， 而其标准差势必为 1.
```

上面的举例中，所有科目的满分都为 100，且单位相同（分）。为防止引起困扰和不解，建议做出以下改动

```
标准计分的特征：

1. 无论作为变量的取值范围是多少（比如，科目满分为 100 或者 150 分），其标准计分的平均数势必为 0， 而其标准差势必为 1；
2. 无论作为变量的单位是什么（比如分、公斤和小时等），其标准计分的平均数势必为 0， 而其标准差势必为 1.
```

3. 函数功能和参数介绍的建议
文件：2.1.2 数据 OSM 与核密度估计

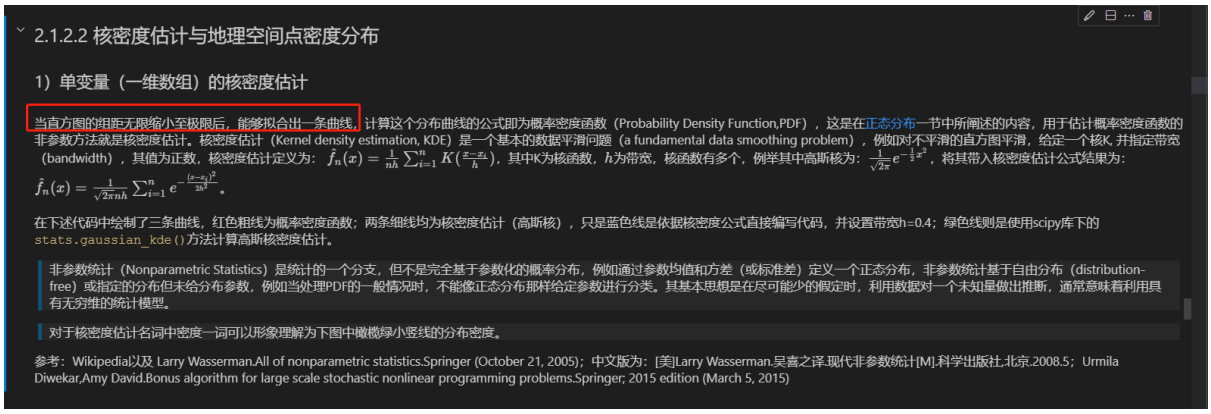
函数功能和参数介绍，放在函数定义 def 的下面会更好
如果函数无返回值，也应该写一下类似的代码
Outputs: None

```python
def save_osm(osm_handler,osm_type,save_path=r"./data/",fileType="GPKG"):
    import geopandas as gpd
    import os
    import datetime

    a_T=datetime.datetime.now()
    print("start time:",a_T)
    '''
    function-根据条件逐个保存读取的osm数据(node, way and area)

    Paras:
        osm_handler - osm返回的node,way和area数据
        osm_type - 要保存的osm元素类型
        save_path - 保存路径
        fileType - 保存的数据类型, shp, GeoJSON, GPKG
    '''
    def duration(a_T):
```

```python
def ptsKDE_geoDF2raster(pts_geoDF,raster_path,cellSize,scale):
    from osgeo import gdal,ogr,osr
    import numpy as np
    from scipy import stats
    '''
    function - 计算GeoDaraFrame格式的点数据核密度估计,并转换为栅格数据

    Paras:
        pts_geoDF - GeoDaraFrame格式的点数据
        raster_path - 保存的栅格文件路径
        cellSize - 栅格单元大小
        scale - 缩放核密度估计值
    '''
    #定义空值（没有数据）的栅格数值 Define NoData value of new raster
    NoData_value=-9999
    x_min, y_min,x_max, y_max=pts_geoDF.geometry.total_bounds

    #使用GDAL库建立栅格 Create the destination data source
    x_res=int((x_max - x_min) / cellSize)
    y_res=int((y_max - y_min) / cellSize)
    target_ds=gdal.GetDriverByName('GTiff').Create(raster_path
    target_ds.SetGeoTransform((x_min, cellSize, 0, y_max, 0,
    outband=target_ds.GetRasterBand(1)
    outband.SetNoDataValue(NoData_value)

    #配置投影坐标系统
    spatialRef = osr.SpatialReference()
    epsg=int(pts_geoDF.crs.srs.split(":")[-1])
    spatialRef.ImportFromEPSG(epsg)
    target_ds.SetProjection(spatialRef.ExportToWkt())

    #向栅格层中写入数据
    #print(x_res,y_res)
    X, Y = np.meshgrid(np.linspace(x_min,x_max,x_res), np.lins
    positions=np.vstack([X.ravel(), Y.ravel()])
    values=np.vstack([pts_geoDF.geometry.x, pts_geoDF.geometry
    print("Start calculating kde...")
    kernel=stats.gaussian_kde(values)
    Z=np.reshape(kernel(positions).T, X.shape)
    print("Finish calculating kde!")
    #print(values)

    outband.WriteArray(np.flip(Z,0)*scale) #需要翻转数组，写栅格单元
    outband.FlushCache()
    print( "conversion has completed! " )
    return gdal.Open(raster_path).ReadAsArray()
```

```python
import rasterio
dataset_gpd=rasterio.open(raster_path_gpd)

from rasterio.plot import show
import matplotlib.pyplot as plt
#plt.figure(figsize=(8,8))
fig, axs=plt.subplots(1,2,figsize=(10, 10))

show((dataset_gpd,1),contour=True,cmap='Greens',ax=axs[0]) #开启等高线模式
show((dataset_gpd,1),cmap='Greens',ax=axs[1])
axs[0].tick_params(axis='x', rotation=90)
axs[1].tick_params(axis='x', rotation=90)
plt.show()
```

4. 2.1.2.2 核密度估计与地理空间点密度分布 中对核密度概念的引入



2.1.2.2 核密度估计与地理空间点密度分布

1）单变量（一维数组）的核密度估计

当直方图的组距无限缩小至极限后，能够拟合出一条曲线，计算这个分布曲线的公式即为概率密度函数（Probability Density Function,PDF），这是在正态分布一节中所阐述的内容，用于估计概率密度函数的非参数方法就是核密度估计。核密度估计（Kernel density estimation, KDE）是一个基本的数据平滑问题（a fundamental data smoothing problem），例如对不平滑的直方图平滑，给定一个核K, 并指定带宽（bandwidth），其值为正数，核密度估计定义为：$\hat{f}_n(x) = \frac{1}{nh}\sum_{i=1}^{n} K(\frac{x-x_i}{h})$，其中K为核函数，$h$为带宽，核函数有多个，例举其中高斯核为：$\frac{1}{\sqrt{2\pi}}e^{-\frac{1}{2}x^2}$，将其带入核密度估计公式结果为：

$\hat{f}_n(x) = \frac{1}{\sqrt{2\pi}nh}\sum_{i=1}^{n} e^{-\frac{(x-x_i)^2}{2h^2}}$。

在下述代码中绘制了三条曲线，红色粗线为概率密度函数；两条细线均为核密度估计（高斯核），只是蓝色线是依据核密度公式直接编写代码，并设置带宽h=0.4；绿色线则是使用scipy库下的`stats.gaussian_kde()`方法计算高斯核密度估计。

> 非参数统计（Nonparametric Statistics）是统计的一个分支，但不是完全基于参数化的概率分布，例如通过参数均值和方差（或标准差）定义一个正态分布，非参数统计基于自由分布（distribution-free）或指定的分布但未给分布参数，例如当处理PDF的一般情况时，不能像正态分布那样给定参数进行分类。其基本思想是在尽可能少的假定时，利用数据对一个未知量做出推断，通常意味着利用具有无穷维的统计模型。

> 对于核密度估计名词中密度一词可以形象理解为下图中橄榄绿小竖线的分布密度。

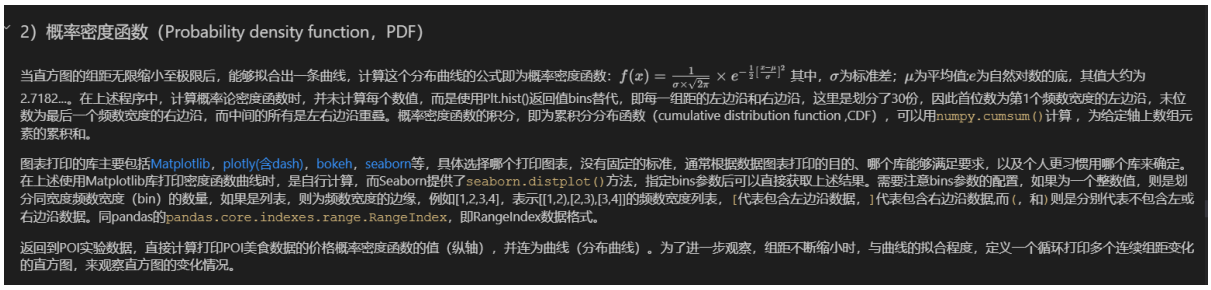
参考：Wikipedia以及 Larry Wasserman.All of nonparametric statistics.Springer (October 21, 2005)；中文版为：[美]Larry Wasserman.吴喜之译.现代非参数统计[M].科学出版社.北京.2008.5；Urmila Diwekar,Amy David.Bonus algorithm for large scale stochastic nonlinear programming problems.Springer; 2015 edition (March 5, 2015)

直接提到“直方图的组距”，会有略突兀的感觉，随后后面提到了“正态分布一节”，但是直接插图或者明确引用，会更好的

可以插入一下对“2.1.1.4 正态分布与概率密度函数，异常值处理” 2）中的那幅图，

或者简单的提一下这幅图的标号

PS：“其中 K 为核函数”，K 应该和后面的“h 为带宽”一样是斜体的公式字母样式

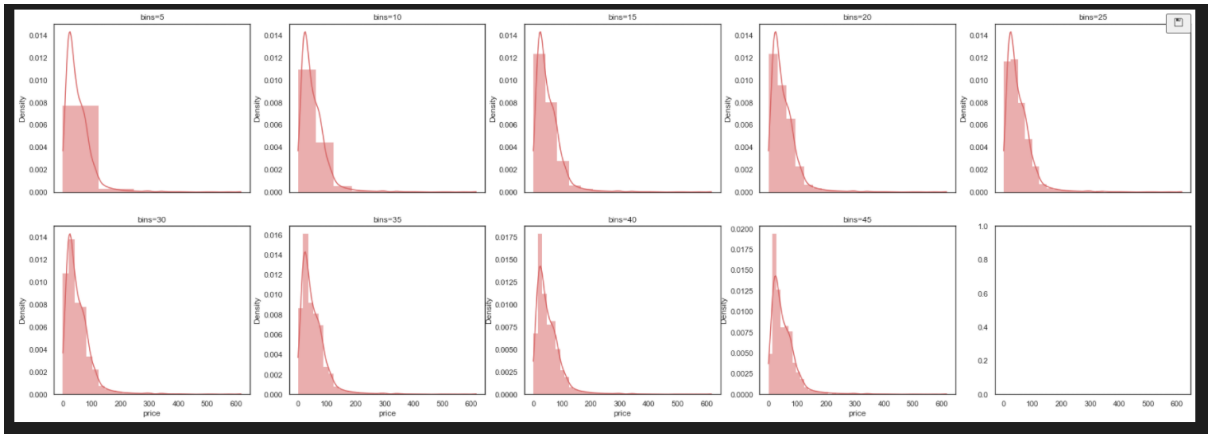


2）概率密度函数（Probability density function，PDF）

当直方图的组距无限缩小至极限后，能够拟合出一条曲线，计算这个分布曲线的公式即为概率密度函数：$f(x) = \frac{1}{\sigma\times\sqrt{2\pi}} \times e^{-\frac{1}{2}[\frac{x-\mu}{\sigma}]^2}$ 其中，σ为标准差；μ为平均值;e为自然对数的底，其值大约为2.7182....。在上述程序中，计算概率论密度函数时，并未计算每个数值，而是使用Plt.hist()返回值bins替代，即每一组距的左边沿和右边沿，这里是划分了30份，因此首位数为第1个频数宽度的左边沿，末位数为最后一个频数宽度的右边沿，而中间的所有是左右边沿重叠。概率密度函数的积分，即为累积分分布函数（cumulative distribution function ,CDF），可以用`numpy.cumsum()`计算，为给定轴上数组元素的累积和。

图表打印的库主要包括Matplotlib，plotly(含dash)，bokeh，seaborn等，具体选择哪个打印图表，没有固定的标准，通常根据数据图表打印的目的、哪个库能够满足要求，以及个人更习惯用哪个库来确定。在上述使用Matplotlib库打印密度函数曲线时，是自行计算，而Seaborn提供了`seaborn.distplot()`方法，指定bins参数后可以直接获取上述结果。需要注意bins参数的配置，如果为一个整数值，则是划分同宽度频数宽度（bin）的数量，如果是列表，则为频数宽度的边缘，例如[1,2,3,4]，表示[[1,2),[2,3),[3,4]]的频数宽度列表，[代表包含左边沿数据，)代表包含右边沿数据,而(，和)则是分别代表不包含左或右边沿数据。同pandas的`pandas.core.indexes.range.RangeIndex`，即RangeIndex数据格式。

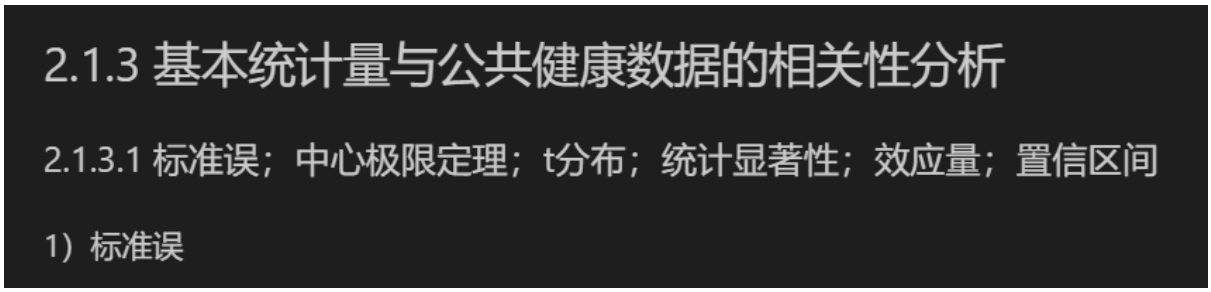
返回到POI实验数据，直接计算打印POI美食数据的价格概率密度函数的值（纵轴），并连为曲线（分布曲线）。为了进一步观察，组距不断缩小时，与曲线的拟合程度，定义一个循环打印多个连续组距变化的直方图，来观察直方图的变化情况。



5. 代码块中，应该 import 在最上面，然后是定义的常量，最后是变量和相应的处理代码

```
restaurant_df=amenity_poi.loc[amenity_poi.amenity=='restaurant']
restaurant_coordinates=np.array([restaurant_df.geometry.x,restaurant_df.geometry.y])
restaurant_kernel=stats.gaussian_kde(restaurant_coordinates)
restaurant_df['restaurant_kde']=restaurant_kernel(restaurant_coordinates)

import plotly.express as px
mapbox_token='pk.eyJ1IjoicmljaGllYmFvIiwiYSI6ImNrYjB3N2NyMzBlMG8yc254dTRzNnMyeHMifQ.QT7MdjQKs9Y6OtaJaJAn0A'
px.set_mapbox_access_token(mapbox_token)
fig=px.scatter_mapbox(restaurant_df,lat=restaurant_df.geometry.y, lon=restaurant_df.geometry.x,color='amenityKDE',colo
fig.update_layout(autosize=False,width=800,height=800,)
fig.show()
```

6. 对章节名称的建议 2



2.1.3.1 标准误；中心极限定理；t 分布；统计显著性；效应量；置信区间

可改为

2.1.3.1 基本统计量：标准误、中心极限定理、t 分布、统计显著性、效应量和置信区间

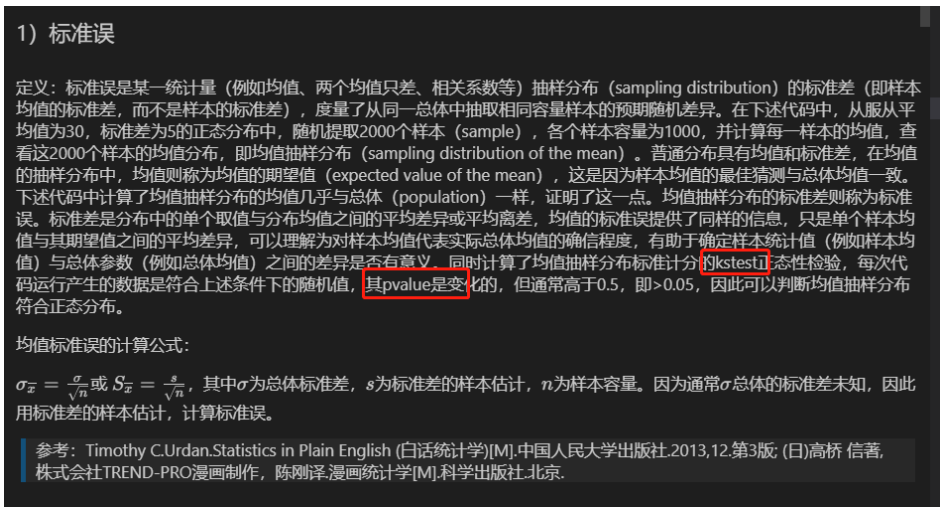
7. 统计学术语修改

1）标准误部分

计算机和统计科学中通常是用修改后的说法

kstest –> K-S test

pvalue → p-value



2.1.3.2 相关性部分

相关系数有关的 Variance 和 Covariance, 一般翻译为方差和协方差（not 变异数和共变异数）

另外补充说明

协方差（Covariance）在概率论和统计学中用于衡量两个变量的总体误差。 而方差是协方差的一种特殊情况，即当两个变量是相同的情况。 协方差表示的是两个变量的总体的误差，这与只表示一个变量误差的方差不同。
来源： 协方差_百度百科 https://baike.baidu.com › item › 协方差 (google.com)

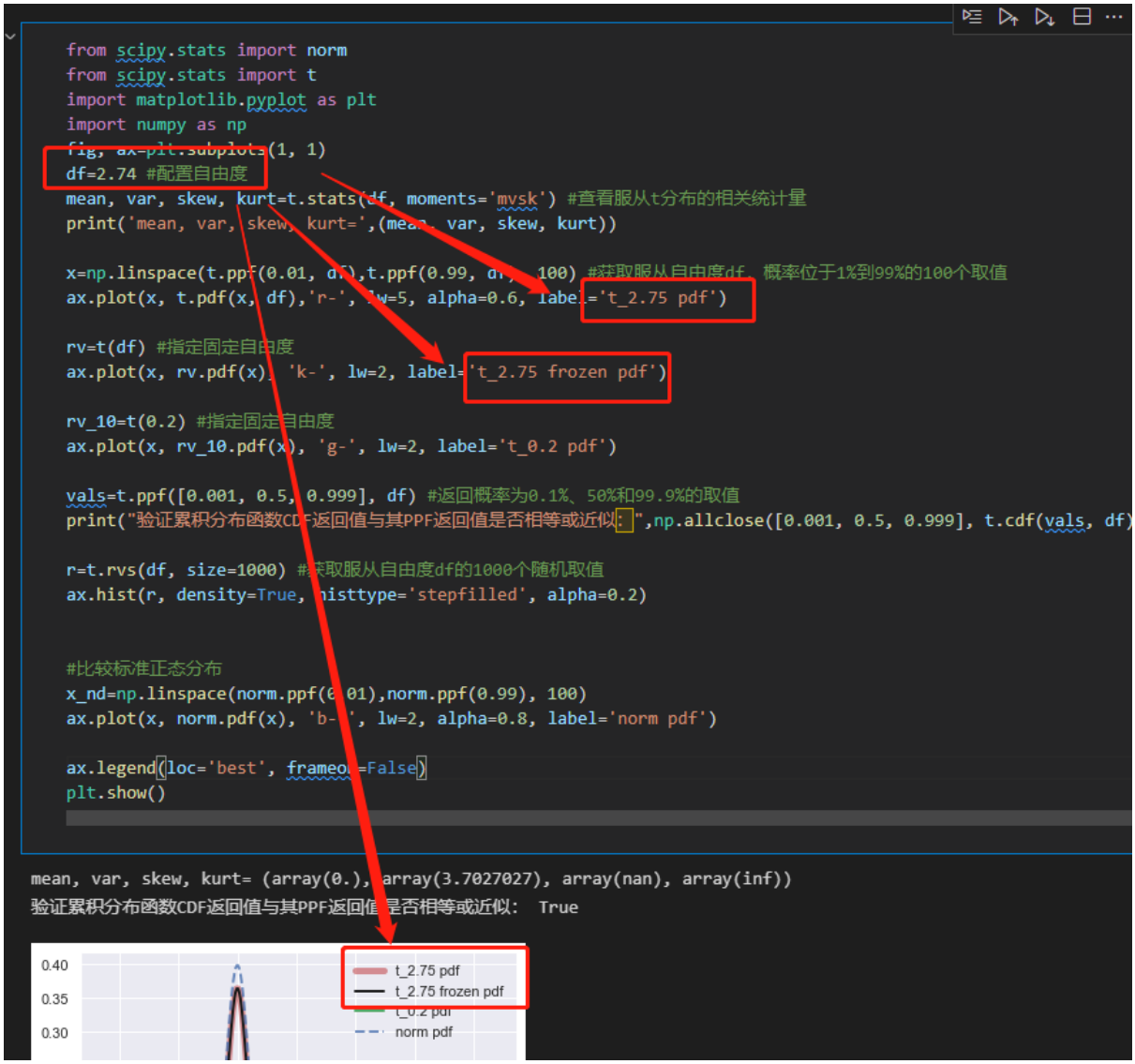
克莱姆相关系数的介绍中，没有介绍提及卡方统计量χ^2（又称为皮尔森卡方统计量）

卡方统计量是指数据的分布与所选择的预期或假设分布之间的差异的度量。 在 1900 年由英国统计学家 pearson 提出，是用于卡方检验中的一个统计量。 它可用于检验类别变量之间的独立性或确定关联性。

克莱姆 V（Cramer's V），又称为克莱姆相关系数、克莱姆关联系数、独立系数等，是双变量相关分析的一种方法，专门用于衡量分类数据与分类数据之间相关程度。该系数取值范围为 0 到 1，0 表示两个变量无关，1 表示完全相关。
https://blog.csdn.net/jbb0523/article/details/92794650

8. 实验中的数值确认与修改

2.1.3 3） t 分布



此处配置自由度时，是否应该为 df=2.75?

此处变量名也不太合适，例如 rv，rv_10,r,等

变量 x_nd 没有用过，疑似变量有误

修改如下，红色标出修改部分

原始代码	修改后
from scipy.stats import norm from scipy.stats import t import matplotlib.pyplot as plt import numpy as np fig, ax=plt.subplots(1, 1) df=2.74 #配置自由度 mean, var, skew, kurt=t.stats(df, moments='mvsk') #查看服从 t 分布的相关	from scipy.stats import norm from scipy.stats import t import matplotlib.pyplot as plt import numpy as np fig, ax=plt.subplots(1, 1) df=2.75 #配置自由度 mean, var, skew, kurt=t.stats(df, moments='mvsk') #查看服从 t 分布的相关

```
统计量
print('mean, var, skew,
kurt=',(mean, var, skew, kurt))

x=np.linspace(t.ppf(0.01,
df),t.ppf(0.99, df), 100) #获取服从自
由度 df，概率位于 1%到 99%的 100 个取值
ax.plot(x, t.pdf(x, df),'r-', lw=5,
alpha=0.6, label='t_2.75 pdf')

rv=t(df) #指定固定自由度
ax.plot(x, rv.pdf(x), 'k-', lw=2,
label='t_2.75 frozen pdf')

rv_10=t(0.2) #指定固定自由度
ax.plot(x, rv_10.pdf(x), 'g-', lw=2,
label='t_0.2 pdf')

vals=t.ppf([0.001, 0.5, 0.999], df)
#返回概率为 0.1%、50%和 99.9%的取值
print("验证累积分布函数 CDF 返回值与其
PPF 返回值是否相等或近似：
",np.allclose([0.001, 0.5, 0.999],
t.cdf(vals, df)))

r=t.rvs(df, size=1000) #获取服从自由
度 df 的 1000 个随机取值
ax.hist(r, density=True,
histtype='stepfilled', alpha=0.2)


#比较标准正态分布
x_nd=np.linspace(norm.ppf(0.01),norm
.ppf(0.99), 100)
ax.plot(x, norm.pdf(x), 'b--', lw=2,
alpha=0.8, label='norm pdf')

ax.legend(loc='best', frameon=False)
plt.show()
```

```
统计量
print('mean, var, skew,
kurt=',(mean, var, skew, kurt))

x=np.linspace(t.ppf(0.01,
df),t.ppf(0.99, df), 100) #获取服从自
由度 df，概率位于 1%到 99%的 100 个取值
ax.plot(x, t.pdf(x, df),'r-', lw=5,
alpha=0.6, label='t_2.75 pdf')

rv=t(df) #指定固定自由度的随机变量
（random variable）序列
ax.plot(x, rv.pdf(x), 'k-', lw=2,
label='t_2.75 frozen pdf')

rv_df_02=t(0.2) #指定固定自由度为 0.2
的随机变量序列
ax.plot(x, rv_df_02.pdf(x), 'g-',
lw=2, label='t_0.2 pdf')

vals=t.ppf([0.001, 0.5, 0.999], df)
#返回概率为 0.1%、50%和 99.9%的取值
print("验证累积分布函数 CDF 返回值与其
PPF 返回值是否相等或近似：
",np.allclose([0.001, 0.5, 0.999],
t.cdf(vals, df)))

rv_1000=t.rvs(df, size=1000) #获取服
从自由度 df 的 1000 个随机取值
ax.hist(rv_1000, density=True,
histtype='stepfilled', alpha=0.2)


#比较标准正态分布
x_nd=np.linspace(norm.ppf(0.01),norm
.ppf(0.99), 100)
ax.plot(x_nd, norm.pdf(x_nd), 'b--',
lw=2, alpha=0.8, label='norm pdf')

ax.legend(loc='best', frameon=False)
plt.show()
```

另外，此处之后的一部分

```python
#如果需要计算服从t分布的概率则指定df，以及loc（均值，默认为0）和scale(标准差，默认为1)
print("用.cdf计算值小于或等于113的概率为: ",t.cdf(113,df=999,loc=100,scale=12))
print("用.sf计算值大于或等于113待概率为: ",t.sf(113,df=999,loc=100,scale=12))
print("可以观察到.cdf（<=113）概率结果+.sf(>=113)概率结果为: ",t.cdf(113,df=999,loc=100,scale=12)+t.sf(113,df=999,lo
print("用.ppf找到给定概率值为98%的数值为: ",t.ppf(0.81766,df=999,loc=100,scale=12))
```

Python

```
用.cdf计算值小于或等于113的概率为:  0.8605390547558804
用.sf计算值大于或等于113待概率为:  0.13946094524411956
可以观察到.cdf（<=113）概率结果+.sf(>=113)概率结果为:  1.0
用.ppf找到给定概率值为98%的数值为:  110.88276310459135
```

用到 sf，ppf，这两个函数[Survival function (1-cdf — sometimes more accurate) Survival function 和 Percent point function 百分比点函数函数 (inverse of cdf — percentiles)]并未在书中提及，可以添加一个介绍 scipy t 分布官方文档的链接

scipy.stats.t — SciPy v0.13.0 Reference Guide

https://docs.scipy.org/doc/scipy-0.13.0/reference/generated/scipy.stats.t.html

9. 4）统计显著性中的歧义表述

原文："绘制样本容量为 1000，2000 个样本的均值抽样分布"

可以修改为

绘制样本容量为 1000，样本数为 2000 的均值抽样分布

10. 公式字母的格式问题

2.1.3.2 相关性部分

相关比	0~1	$WSS = \sum_{k=1}^{K} S_{x^k} = \sum_{k=1}^{K} \sum_{i=1}^{n_i} (x_i^k - \overline{x^k})^2$ $BSS = \sum_{k=1}^{K} n_k (\overline{x^k} - \overline{x})^2$ $cr = \frac{BSS}{BSS+WSS}$	设共有n个数值数据，它们被分成了K个类别，$x_1^1, x_2^1, \cdots, x_{n_1}^1; x_1^2, x_2^2, \cdots, x_{n_2}^2; \cdots; x_1^K, x_2^K, \cdots, x_{n_K}^K$，其中$n_K$代表第K个类别的数值的个数，记$\overline{x^k}$为第K个类别数值的均值，$\overline{x}$为所有数据的均值，BSS（between sum of squares）为组间变异，WSS（within sum of squares）组内变异
克莱姆相关系数	0~1	$\chi^2 = \sum_{i,j} \frac{(n_{ij} - \frac{n_i n_j}{n})^2}{\frac{n_i n_j}{n}}$ $V = \sqrt{\frac{\chi^2/n}{min(k-1, r-1)}}$	设有两个类别变量A和B，观测样本总数为n，对于$i = 1, \cdots, r; j = 1, \cdots, k$，$n_{ij}$为$(A_i, B_j)$的观测次数（频数），$k$为观察次数表列，$r$为其行。$\frac{n_i n_j}{n}$为计算期望次数

11.