附录一 软件与数据

一、Anaconda3安装

之前写过一篇R语言可视化数据挖掘工具Rattle的离线安装，这几天把Python开源数据科学套件离线安装到工作用的PC上，也整理一下。内网不能上网，所以要离线安装。最新Anaconda3安装好后有301个包5.4G，这几个月在笔记本上学习Python，陆续安装的包有70个454M。如果在线安装，conda install会自动下载安装依赖的包，而离线最有效的方法，就是从安装好的笔记本上直接拷贝过去，否则逐个下载安装是比较繁琐的。Python安装包就是解压拷贝到site-packages目录下，不会改注册表，因为它要保持Windows、Linux、MacOS跨平台的兼容，拷贝展开是最好的办法。

（一）笔记本在线安装  
1、先下载安装最新anaconda3 Windows python3.7单独发行版2020.02，477M， python2.7已停止更新了，装3.7或3.8版。  
地址：https://www.anaconda.com/distribution/  
2、在线安装其它需要的包。Spyder最新的4.0.1版已装好，可以启动Anaconda Navigator在面板上选择安装Orange3.23.1及Glueviz0.15.2，也可以启动Anaconda Prompt命令行窗口，在其中用 “conda install 包名”安装。用conda安装比用pip安装要好，它会管理好虚拟环境及版本的关系。  
3、下载安装好Graphviz，解压即可，然后把它的bin目录加入到系统环境变量的路径变量path中。参阅：https://graphviz.gitlab.io/\_pages/Download/Download\_s.html 及https://github.com/xflr6/graphviz/blob/master/docs/manual.rst 。  
4、在Anaconda Prompt命令行窗口中用conda list --export --json>d:\temp\python-packages.txt输出此时的包列表。

（二）PC离线安装  
1、同上面一样离线装好Anaconda3。  
2、在Anaconda Prompt命令行窗口中用conda list --export --json>d:\temp\python-packages2.txt输出此时的包列表。  
3、编一段python程序输出两个包清单的差异。

import pandas as pd

import numpy as np

import os

packages = pd.read\_csv("d:/temp/python-packages.txt",sep="=")

packages = packages[["Name","Version"]]

packages2 = pd.read\_json("d:/temp/python-packages2.txt")

packages2 = packages2[["name","version"]]

packages2.rename(columns = {"name":"Name","version":"Version2"}, inplace = True)

packages3 = pd.merge(packages,packages2, how = "right")

packages4 = packages3[packages3.isnull().T.any()]

packages4 = packages4.reset\_index(drop=True)

packages4.to\_csv("packages\_to\_install.txt")

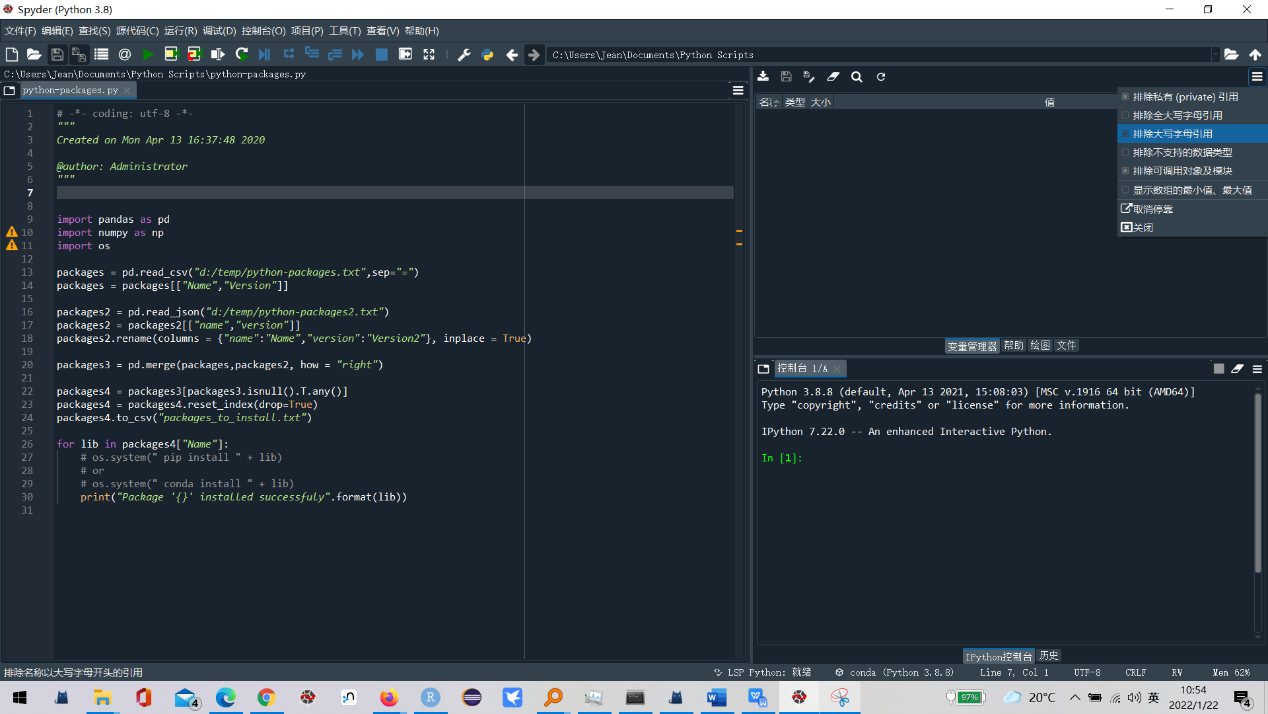
4、建立一个临时目录，把笔记本上在D:\Anaconda3\Lib\site-packages目录中上述差异清单中包的目录都拷贝进去，把笔记本与PC上述目录下差异的文件如easy-install.pth等拷贝进去，一共有十几个文件。然后把这些目录及文件拷贝到PC上相应的目录下。  
5、同上面一样安装好Graphviz。

（三）在拷贝笔记本的安装版本之前，可以按前面文章的介绍在笔记本上hack Orange,并安装我开发的orange3-glueviz包，以实现Spyder、Orange系列、Glueviz的之间无缝集成，充分发挥套件的威力。可以用 conda install先更新到Orange的最新版本3.24.1。

（四） Spyder的设置。如果要在Spyder中调用Orange，要更改Spyder的 工具->偏好设置->IPython控制台->绘图->图形后端 的后端为“行内”,即不使用QuickTime后端。因为Orange会使用它并打开一个QuickTime主程序界面，而QuickTime每个主进程中只允许有一个主程序界面。如果IPython也使用，则在Orange退出后，会让Spyder所连接的Python kernel进程挂起，只能重启，从而丢掉调用Orange之前的状态。具体参阅：  
https://github.com/spyder-ide/spyder/issues/4349及https://github.com/ipython/ipython/blob/master/IPython/lib/guisupport.py#L112 。



（五） Spyder变量管理器的设置。默认的“排除私有引用”、“排除全大写引用”、“排除可调用对象及模块”是比较合适的，这样每次执行语句后，变量管理器中显示的变量比较少，主要是脚本中自己建立的变量。



（六）连接Oracle，用rc\_Oracle包。

import cx\_Oracle

import pandas as pd

conn = cx\_Oracle.connect("db\_jscx/\*\*\*\*\*\*\*@150.28.20.172:1521/zhjsxf")

cursor = conn.cursor()

def Select(cursor,sql):

result = cursor.execute(sql)

all\_data = cursor.fetchall()

data = pd.DataFrame(all\_data)

colNames = {}

columns = cursor.description

for i in range(len(columns)):

colNames[i] = columns[i][0]

data.rename(columns = colNames,inplace = True)

return data

sql = "select zsxm\_dm,zsxmmc from hx\_dm\_qg.dm\_gy\_zsxm order by zsxm\_dm"

data = Select(cursor,sql)

sql = "select a.zsxm\_dm,b.zsxmmc,sum(a.ybtse) as ybtse \

from hx\_zs.zs\_yjsf a \

left join hx\_dm\_qg.dm\_gy\_zsxm b on a.zsxm\_dm=b.zsxm\_dm \

where \

a.skcllx\_dm='1' \

and a.yjskzt\_dm not in('12','13') \

and a.czlx\_dm not in ('19','49','54','70','99') \

and a.tzlx\_dm in ('1','4') \

and a.sjgsdq like '24404%' \

and a.SKSSQQ>= date '2014-12-01' and a.skssqz< date '2015-12-01' \

group by a.zsxm\_dm,b.zsxmmc \

order by a.zsxm\_dm \

"

data = Select(cursor,sql)

cursor.close()

conn.commit()

conn.close()

二、Spyder、Orange、Glueviz集成

现在大数据分析主要用R、Python、Java等语言，R与Python更简单易用一点，渐渐成为客户端的主流。开源工具中，Rstudio、Rattle、Gobi是R语言一个不错的组合，Spyder、Orange、Glueviz是Python一个不错的组合。各大数据云计算平台基本都支持Python，也有同时支持R语言的，对于数据工作者来说，熟练掌握一套工具，连接各个云平台开展数据分析工作是更有效的策略。一方面学习曲线较短，便于积累提高（这很重要，数据挖掘是金字塔尖的技能，需要长期的学习与实践积累）；另一方面，软件开发是个高投入，重积累的行业，好的产品需要时间成长和筛选，通用工具一般代表了行业共同的选择，功能更强水平更高，国内的同类产品还是有不小的差距。当然各云平台必定也有自己特色的开发平台与工具，尤其是在云端的绑定会更紧密。当工作具体到某个云平台时，也要同时掌握运用其自带的工具。Anaconder是目前世界上主流的Python与R语言发行平台，全世界有1800万的数据科学用户，这些工具反映了大家共同的选择，比较有代表性。经过近两个月的了解，觉得Python上的这几个工具确实不错，并且开源社区的推动也会让它们有更广阔的未来。

Orange的可视化数据挖掘确实方便，也通过Python Script Widget带来灵活性，但这个组件缺乏调试的功能，只适合运行完成状态的脚本。数据挖掘前期的数据抽取、清洗、整合、加工等要占工作量的一半以上，交互性很强，不断迭代，Spyder是更合适的工具；Orange数据可视化展示探索的能力有所不足，Glueviz则可以补充。所以，让数据在三者之间无缝流动是很必要的。Glueviz可以在Spyder中传入数据集启动，已经可以支持，Orange则需要稍加改造。

Spyder启动Glueviz并传入数据集，这是Glueviz自带的例子：

from glue.core.data\_factories import load\_data

from glue.core import DataCollection

from glue.core.link\_helpers import LinkSame

from glue.app.qt.application import GlueApplication

#load 2datasets from files

image =load\_data('D:/迅雷下载/w5/w5.fits')

catalog =load\_data('D:/迅雷下载/w5/w5\_psc.vot')

dc =DataCollection([image, catalog])

# linkpositional information

dc.add\_link(LinkSame(image.id['RightAscension'], catalog.id['RAJ2000']))

dc.add\_link(LinkSame(image.id['Declination'],catalog.id['DEJ2000']))

#startGlue

app =GlueApplication(dc)

app.start()

本文主要讲怎样hack Orange，以从Spyder传入数据。Orange工作流中，各节点间传递的是数据，Orange的一个主要优点是，节点数据的变化，会触发讯号并把变化传递给后续的所有节点，即时更新整个工作流的输出。数据在节点间流动的主要载体是数据表，Orange.data.table.Table，在Orange可视化界面上是个Data Table Widget，所以传进数据要做的工作是：

1、传进Table

2、在主界面空白画布上创建DataTable Widget

3、把数据绑定到DataTable Widget

具体可参阅[我的GitHub项目文档](https://github.com/icejean/orange3-glueviz/blob/master/docs/ChangeList-of-Orange3.24.1.txt)下面来看看怎样完成，要魔改定制Orange的可以参考。这里是改好的3.24.1版程序，[\_main\_.py](https://github.com/icejean/orange3-glueviz/blob/master/spyder-orange3/orange/canvas/__main__.py)，[widgetsscheme.py](https://github.com/icejean/orange3-glueviz/blob/master/spyder-orange3/orangewidget/workflow/widgetsscheme.py), 可以参考阅读下文。

1、找到主程序入口点。

从Orange的命令行窗口启动命令：

python –m Orange.canvas

可以找到Orange的主程序D:\Anaconda3\Lib\site-packages\Orange\canvas\\_\_main\_\_.py及\_\_init\_\_.py，打开浏览，找到主程序\_\_main\_\_.py中的main()函数，可以看到默认是没有入口参数的，但可以接受命令行的选项参数：

def main(argv=None):

更改为：

# Modified by Jean 2020/01/10, add two parameters:

# datasets: the Orange table list passed in from Spyder

# logfile: the file name for Orange log messages,

# since the windows stdout & stderr problem for sub process

def main(argv=None, datasets=None, logfile=None):

后面要把datasets = [Table List]传进去，把日志重定向到logfile=某文件，来找到要改哪里。

2、更改日志输出，增加日志重定向到文件的选项。

找到main()中初始化日志的语句：

logging.basicConfig(

level=levels[options.log\_level],

handlers=[make\_stdout\_handler(levels[options.log\_level])]

)

改为当有日志文件参数传入时，重定向到该文件，记录到最详细的debug级：

# modified by Jean2020/01/10,to redirect log message to a file

if logfile is notNone:

logging.basicConfig(

level=logging.DEBUG,

filename=logfile

)

else:

logging.basicConfig(

level=levels[options.log\_level],

handlers=[make\_stdout\_handler(levels[options.log\_level])]

)

前面有一行要注释掉，也是有关日志输出的，Python3.7.5在windows上已经解决了这个bug。

# comment out by Jean 2020/01/09, this bug is fixed in python3.7.5

# fix\_win\_pythonw\_std\_stream()

3、指定一个日志文件，从Spyder启动Orange,点击Widget工具区的DataTable Widget在空白画布上添加一个Data Table Widget，然后查看日志文件的输出，Orange执行了哪些操作。

import sys

from Orange.canvas import \_\_main\_\_ as OrangeMain

sys.exit(OrangeMain.main(logfile="orange.log"))

用其它编辑器打开日志文件，可以看到最后有两行：

INFO:orangecanvas.scheme.scheme:Added node'Data Table' to scheme ''.

INFO:orangewidget.workflow.widgetsscheme:WidgetManager:Creating 'Orange.widgets.data.owtable.OWDataTable' instance 'Data Table'.

确定添加Data Table Widget到画布的是orangewidget.workflow.widgetsscheme.py。

退出Orange回到Spyder。

4、打开D:\Anaconda3\Lib\site-packages\orangewidget\workflow\widgetsscheme.py，找到输出的日志“WidgetManager: Creating”，它只有1处，在Widget的基础实现类:

class OWWidgetManager(\_WidgetManager):

"""

OWBaseWidget instancemanager class.

This class handles thelifetime of OWBaseWidget instances in a

:class:`WidgetsScheme`.

"""

def create\_widget\_instance(self, node):

# type: (SchemeNode) -> OWBaseWidget

"""

Create a OWBaseWidget instance for the node.

"""

……

try:

# Lookup implementation class

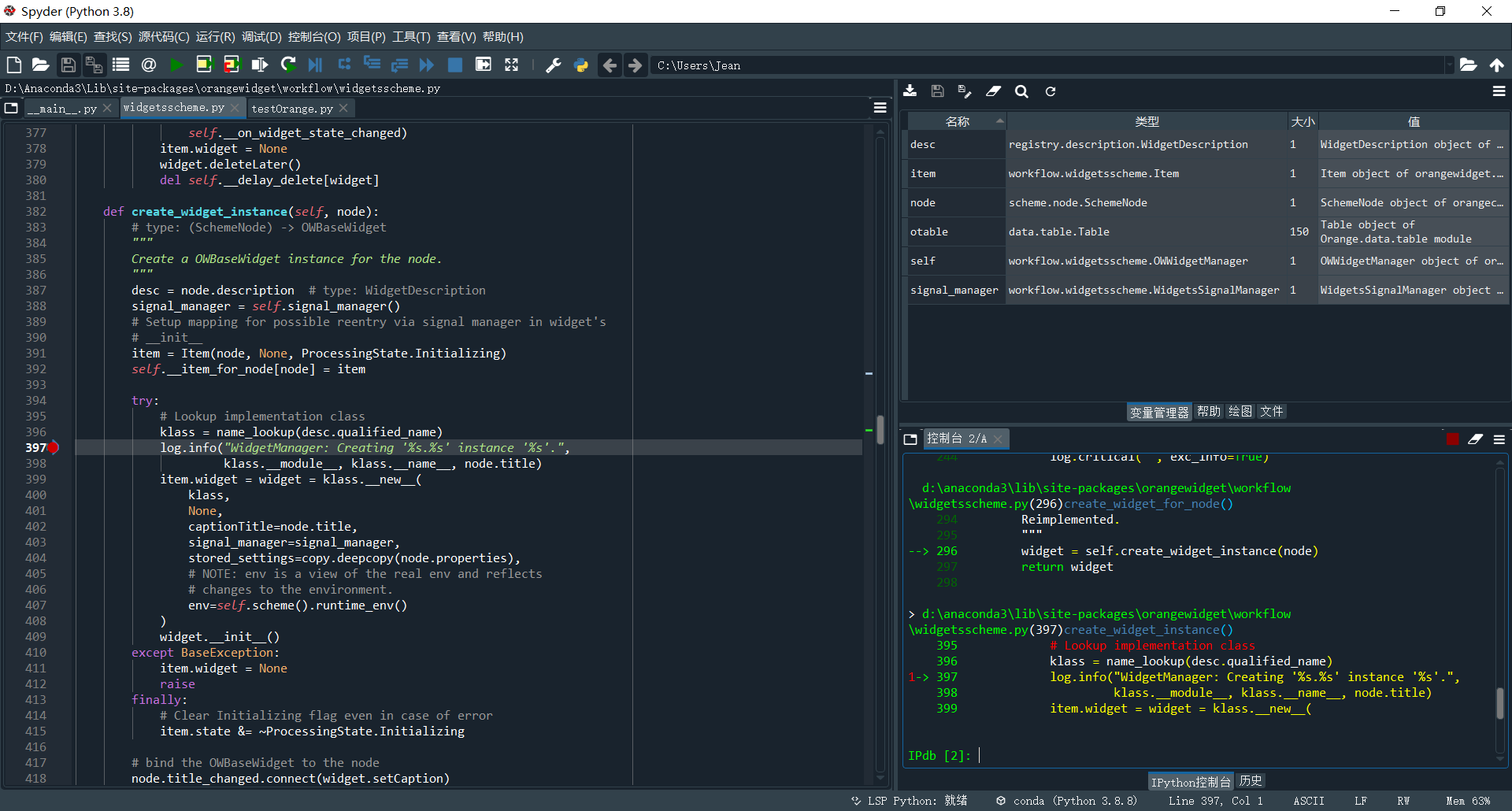
klass = name\_lookup(desc.qualified\_name)

log.info("WidgetManager: Creating '%s.%s' instance '%s'.",

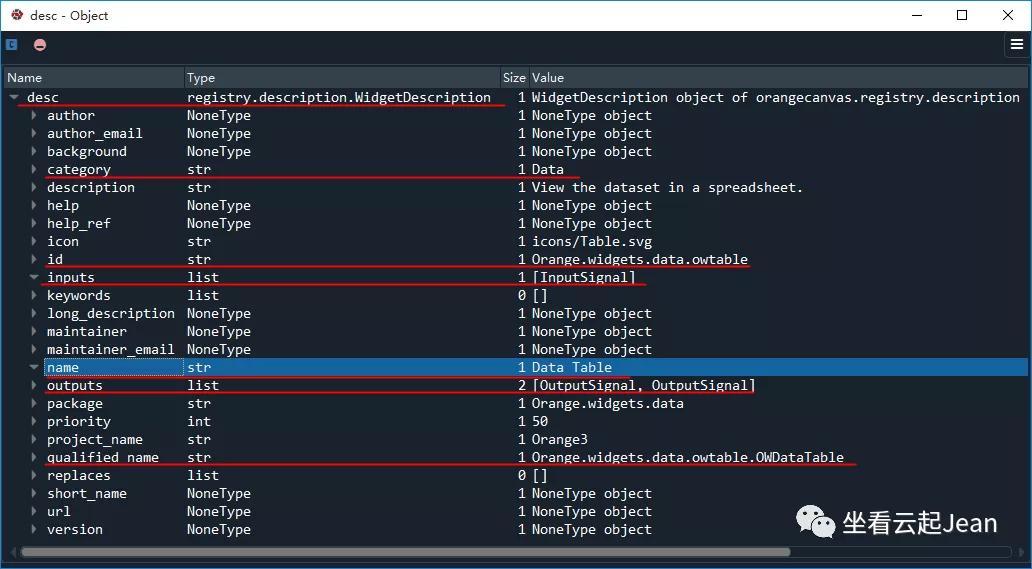
klass.\_\_module\_\_, klass.\_\_name\_\_, node.title)

……

这是创建Widget实例的地方，create\_widget\_instance（），在log.info那里设置一个断点，重新以debug的方式调用Orange，点击Data Table Widget添加一个到空白画布中，Spyder在断点处停下。



通读该函数，可以看到该函数的入参是一个节点对象node，从中提取节点描述参数desc来实例化具体的Widget类，desc是一个registry.description.WidgetDescription类。参阅开头的import语句找到orangecanvas.registry.description.py，可以了解WidgetDescription类中各属性的意义，可知这是一个Data Table Widget，它的唯一标识符是Orange.widgets.data.owtable，由Orange.widgets.data.owtable.OWDataTable类来实现，归属Data大类，有1个输入和2个输出。



5、在Spyder控制台ipdb>提示符（这是debug模式的提示符）后输入 where命令，输出直到断点的整个堆栈调用列表，查看调用及参数传递的过程。在ipdb命令行提示符后通过输入up命令上移一级堆栈，down命令下移一级堆栈，Spyder编辑器会自动定位到该级调用堆栈对应的代码，变量管理器会更新到显示该级堆栈的变量（注意，Spyder变量管理器只显示当前堆栈的变量，不会显示各级累积的所有变量）。

IPdb [5]: up

> d:\anaconda3\lib\site-packages\orangecanvas\scheme\widgetmanager.py(378)\_\_process\_init\_queue()

376 log.debug("\_\_process\_init\_queue: '%s'", node.title)

377 try:

--> 378 self.ensure\_created(node)

379 finally:

380 if self.\_\_init\_queue:

找到参数来源的尽头，可知Orange通过将动作压入及弹出队列的异步方式来处理添加Data Table的操作，需要从上往下跟踪来找到入队的代码，了解添加动作入队时Orange执行的操作，然后根据上面变量管理器显示的desc对象各属性的值来重构一个desc对象自动模拟点击添加的动作。

def \_\_process\_init\_queue(self):

if self.\_\_init\_queue:

node = self.\_\_init\_queue.popleft()

assert self.\_\_workflow is not None

assert node in self.\_\_workflow.nodes

log.debug("\_\_process\_init\_queue: '%s'", node.title)

try:

self.ensure\_created(node)

finally:

if self.\_\_init\_queue:

self.\_\_init\_timer.start()

再往上一级，已到达Orange的主程序入口：

IPdb [6]: up

> d:\anaconda3\lib\site-packages\orange\canvas\\_\_main\_\_.py(703)main()

701 patch('sys.stderr', stderr),\

702 patch('sys.stdout', stdout):

--> 703 status = app.exec\_()

704 except BaseException:

705 log.error("Error in main event loop.", exc\_info=True)

从log.info可知，程序在这里完成初始化，进入了主循环：

log.info("Entering main event loop.")

excepthook = ExceptHook(stream=stderr)

excepthook.handledException.connect(handle\_exception)

try:

with closing(stdout),\

closing(stderr),\

closing(stream), \

patch('sys.excepthook', excepthook),\

patch('sys.stderr', stderr),\

patch('sys.stdout', stdout):

status = app.exec\_()

except BaseException:

log.error("Error in main event loop.", exc\_info=True)

模拟点击并自动添加一个Data Table Widget的操作应该在程序进入主循环之前的此处进行。入队操作是在status= app.exec\_()主循环里触发的（在主界面点击了Data Table Widget）。

通读main()函数，可知Orange主窗口画布的句柄对象是canvas\_window，它是一个Orange.canvas.mainwindow.py中的MainWindow对象。要操作该对象添加Data Table Widget。通读该源码，可知它继承了orangewidget.workflow.mainwindow.py中的OWCanvasMainWindow类，该类又继承了orangecanvas.application.canvasmain.py中的CanvasMainWindow类，有2千多行程序，是画布的主要实现类。

6、浏览orangecanvas.application.canvasmain.py的CanvasMainWindow类，在直接的类属函数中，只有这个函数最接近处理点击Widget动作的函数：

def on\_tool\_box\_widget\_activated(self, action):

"""A widget action in the widget toolbox has been activated.

"""

widget\_desc = action.data()

if isinstance(widget\_desc, WidgetDescription):

scheme\_widget = self.current\_document()

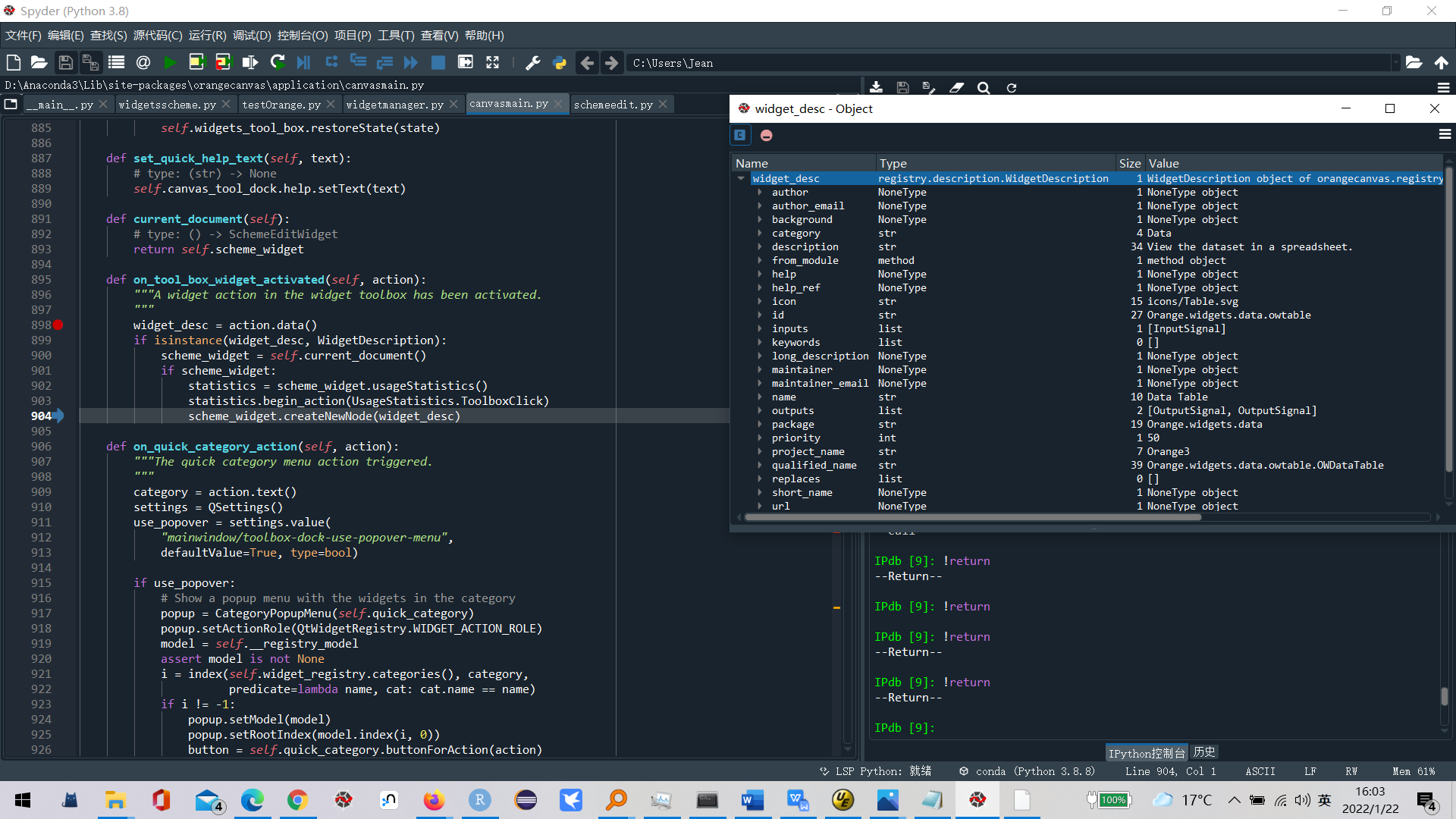
if scheme\_widget:

statistics = scheme\_widget.usageStatistics()

statistics.begin\_action(UsageStatistics.ToolboxClick)

scheme\_widget.createNewNode(widget\_desc)

在widget\_desc = action.data()处设置断点，在Orange中点击Data Table Widget，Spyder停在了断点处，确定这是在画布上添加Widget的入口。查看widget\_desc的内容，确定这是一个Data Table widget的WidgetDescription。单步执行并跟踪进下级函数，找到添加动作入队的代码。



# d:\anaconda3\lib\site-packages\orangecanvas\document\schemeedit.py

def createNewNode(self, description, title=None, position=None):

# type: (WidgetDescription, Optional[str], Optional[Pos]) -> SchemeNode

"""

Create a new :class:`.SchemeNode` and add it to the document.

The new node is constructed using :func:`~SchemeEdit.newNodeHelper`

method

"""

node = self.newNodeHelper(description, title, position)

self.addNode(node)

return node

def addNode(self, node):

# type: (SchemeNode) -> None

"""

Add a new node (:class:`.SchemeNode`) to the document.

"""

if self.\_\_scheme is None:

raise NoWorkflowError()

command = commands.AddNodeCommand(self.\_\_scheme, node)

self.\_\_undoStack.push(command)

7、在D:\Anaconda3\Lib\site-packages\Orange\canvas\\_\_main\_\_.py主函数main()进入主循环前插入模拟点击添加DataTable Widget的代码：

……

# Added by Jean 2020/01/10, insert initial data table widgets if it's not None

insertTableWidget(canvas\_window, datasets)

log.info("Entering main event loop.")

……

然后在末尾的if语句之前加上以下的变量定义及函数完成插入Data Table的操作：

# Added by Jean 2020/01/10, for passing in initial tables to data table widgets

init\_tables = None

def insertTableWidget(canvas\_window,datasets):

if datasets is not None:

# save reference to the tables passed in for creating instances of OWDataTable latter

global init\_tables

init\_tables = datasets

# process the table list passed in

for table in datasets:

# create a WidgetDescription for creating a node for the passed in table latter

init\_widget = newDataTableDesc(table)

if init\_widget is not None:

# get the schema of the current blank canvas

scheme\_widget = canvas\_window.current\_document()

if scheme\_widget:

# create a node for it,

# and let Orange create an instance for the OWDataTable latter,

# through it's popup queue mechanism

# Orange 3.24.0

# scheme\_widget.usageStatistics().set\_node\_type(UsageStatistics.NodeAddClick)

# node = scheme\_widget.createNewNode(init\_widget)

# Orange 3.24.1

statistics = scheme\_widget.usageStatistics()

statistics.begin\_action(UsageStatistics.ToolboxClick)

scheme\_widget.createNewNode(init\_widget)

log.debug("Node from spyder added: ",init\_widget.name)

def newDataTableDesc(table):

from orangecanvas.registry.description import WidgetDescription

from orangecanvas.registry.description import InputSignal

from orangecanvas.registry.description import OutputSignal

widget = WidgetDescription("FromSpyder-"+table.name,"Orange.widgets.data.owtable",\

qualified\_name = "Orange.widgets.data.owtable.OWDataTable")

# Strings

widget.category = "Data"

widget.description = "View the dataset in a spreadsheet."

widget.icon = "icons/Table.svg"

widget.package = "Orange.widgets.data"

widget.project\_name = "Orange3"

widget.priority = 1

# Lists

widget.keywords = []

# One InputSignal for a data table widget

in\_c = InputSignal(name='Data', type=('Orange.data.table.Table',), handler='set\_dataset')

widget.inputs = [in\_c]

# Two OutputSignals for a data table widget

out\_c1 = OutputSignal(name='Selected Data', type=('Orange.data.table.Table',))

out\_c2 = OutputSignal(name='Data', type=('Orange.data.table.Table',))

widget.outputs = [out\_c1, out\_c2]

widget.output = []

widget.replaces = []

# Nones

widget.author = None

widget.author\_email = None

widget.background = None

widget.help = None

widget.help\_ref = None

widget.long\_description = "This data table is come from Spyder by calling Orange."

widget.maintainer = None

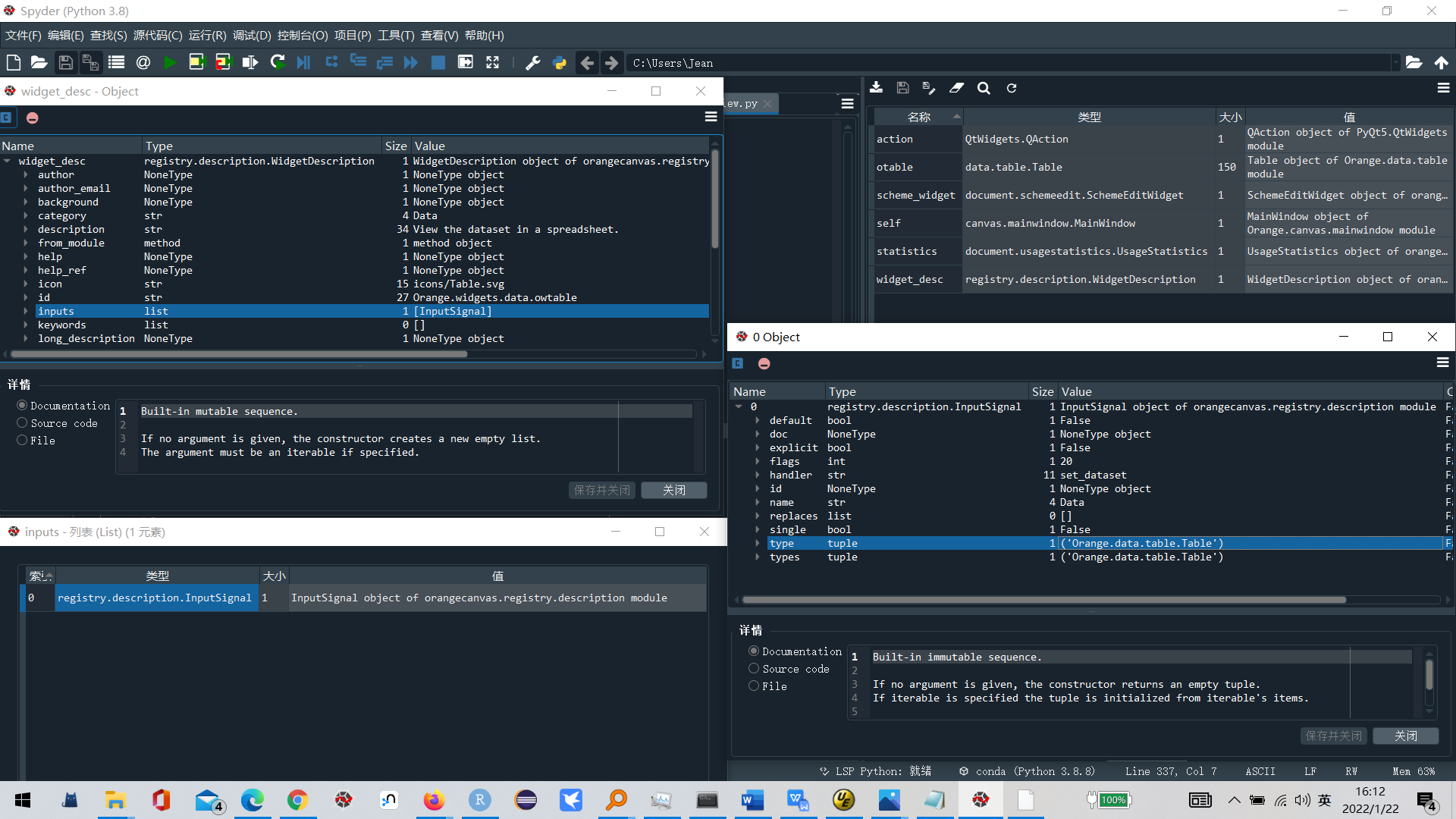
widget.maintainer\_email = None

widget.url = None

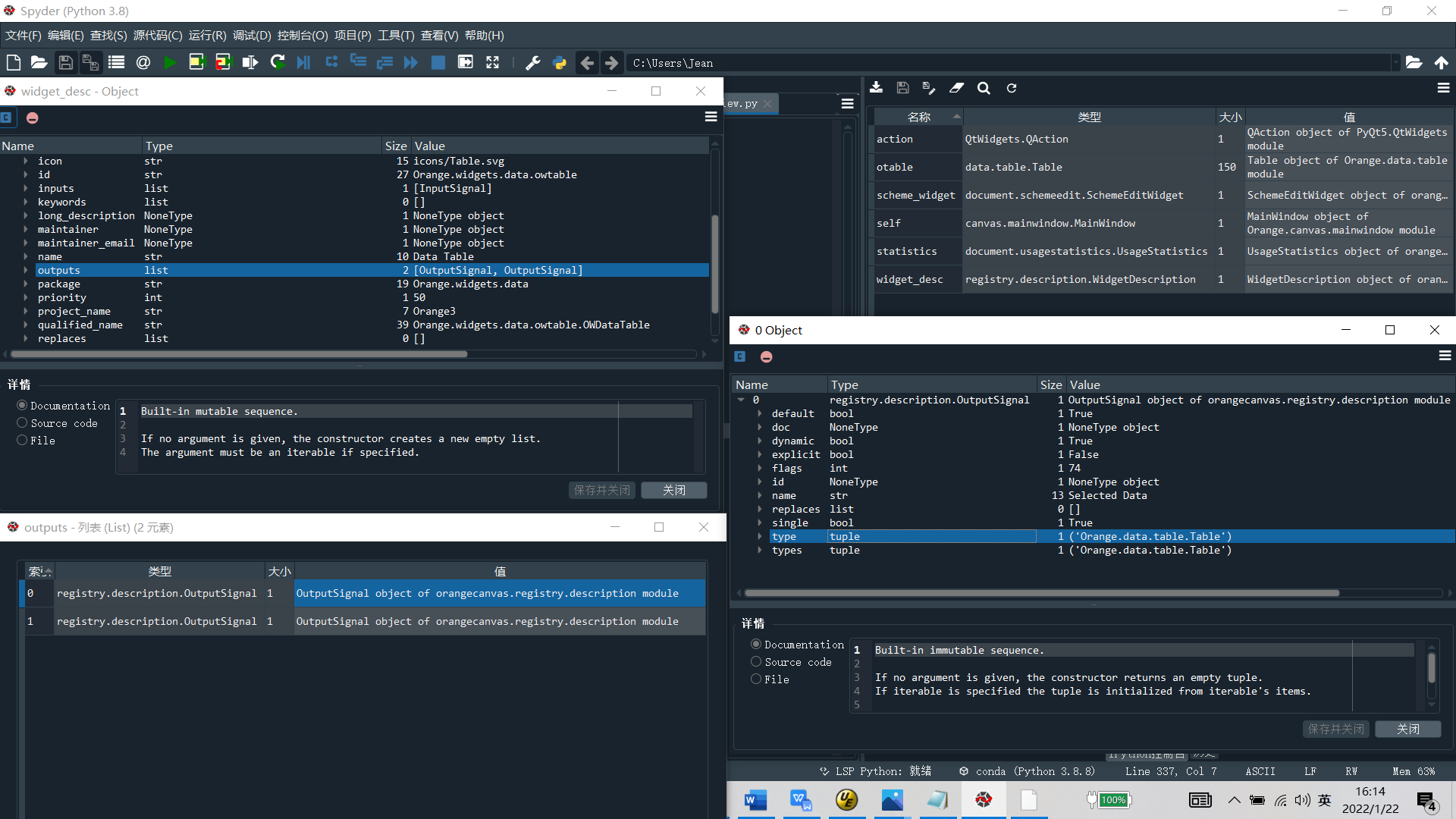
widget.version = None

return widget

Data Table Widget Input Signal的定义,它有1个输入：



Data Table Widget Output Signal的定义，它有2个输出：



8、绑定传入的 datatable到Data Table Widget。在前面D:\Anaconda3\Lib\site-packages\orangewidget\workflow\widgetsscheme.py的create\_widget\_instance()函数中，新建的Data TableWidget是没有数据的，绑定传入的数据应在函数返回前完成。前面提到Data Table Widget由Orange.widgets.data.owtable.OWDataTable类实现，打开D:\Anaconda3\Lib\site-packages\Orange\widgets\data\owtable.py浏览，可知OWDataTable类有set\_dataset()来设置。

@Inputs.data

def set\_dataset(self, data, tid=None):

"""Set the input dataset."""

在D:\Anaconda3\Lib\site-packages\orangewidget\workflow\widgetsscheme.py的create\_widget\_instance()函数返回前插入代码调用set\_dataset()绑定传入的数据即可：

……

# Added by Jean 2020/01/10, bind a passed in table to an OWDataTable widget

# import the orange table list passed in from Orange.canvas.\_\_main\_\_

# and set the dataset to it for a data table widget passed in from Spyder

from Orange.canvas.\_\_main\_\_ import init\_tables

isSpyder = desc.name[:11]

# only data table passed in from Spyder is initialized

if isSpyder == "FromSpyder-" and \

desc.qualified\_name == "Orange.widgets.data.owtable.OWDataTable" and\

init\_tables is not None:

wtable = None

for table in init\_tables:

name = "FromSpyder-"+table.name

if name == desc.name:

wtable = table

break

if wtable is not None:

widget.set\_dataset(wtable)

log.info("Data bind to the data table widget: ", desc.name)

return widget

……

9、在Spyder中调用并传入数据：

# -\*- coding: utf-8 -\*-

"""

Created on Sun Jan 12 10:32:46 2020

@author: Jean

"""

import pandas as pd

from odps.inter import enter

import time

# Read data from ODPS， 取得odps入口

if 'o' not in globals():

room = enter()

odps = o = room.odps

# 不使用云端的 data frame，把数据下载到客户端，逐行读入，建立各字段列表，初始化空列表

djxh = [] ; kzztdjlx\_dm = [] ; djzclx\_dm = [] ; hy\_dm = []

ns = [] ; pjrs = [] ; gzze = [] ; gsze = [] ; pjgz = []

ymin = [] ; rsmin = [] ; gzmin = [] ; gsmin = [] ; pjmin = []

ymax = [] ; rsmax = [] ; gzmax = [] ; gsmax = [] ; pjmax = []

yb = [] ; rsb = [] ; gzb = [] ; gsb = [] ; pjb = []

ye = [] ; rse = [] ; gze = [] ; gse = [] ; pje = []

risk = [] ; perc = [] ; amount = [] ; isRisk = []

t1 = time.time()

sql = 'select \* from udaftest3'

# 逐行载入数据

with odps.execute\_sql(sql).open\_reader() as reader:

for record in reader:

djxh.append(record[0]) ; kzztdjlx\_dm.append(record[1])

djzclx\_dm.append(record[2]) ; hy\_dm.append(record[3])

ns.append(record[4]) ; pjrs.append(record[5]) ; gzze.append(record[6])

gsze.append(record[7]) ; pjgz.append(record[8])

ymin.append(record[9]) ; rsmin.append(record[10]) ; gzmin.append(record[11])

gsmin.append(record[12]) ; pjmin.append(record[13])

ymax.append(record[14]) ; rsmax.append(record[15]) ; gzmax.append(record[16])

gsmax.append(record[17]) ; pjmax.append(record[18])

yb.append(record[19]) ; rsb.append(record[20]) ; gzb.append(record[21])

gsb.append(record[22]) ; pjb.append(record[23])

ye.append(record[24]) ; rse.append(record[25]) ; gze.append(record[26])

gse.append(record[27]) ; pje.append(record[28])

risk.append(record[29]) ; perc.append(record[30]) ; amount.append(record[31])

# 生成是否有风险的标签列，用于后面机器学习的各种算法

if record[29]>0 :

isRisk.append(1)

else:

isRisk.append(0)

t2 = time.time()

print("time: ",t2-t1)

# 从 list建立 pandas data frame，先不包括标签列

import pandas as pd

data = pd.DataFrame({"djxh":djxh,\

"ns":ns,"pjrs":pjrs,"gzze":gzze,"gsze":gsze,"pjgz":pjgz,\

"ymin":ymin,"rsmin":rsmin,"gzmin":gzmin,"gsmin":gsmin,"pjmin":pjmin,\

"ymax":ymax,"rsmax":rsmax,"gzmax":gzmax,"gsmax":gsmax,"pjmax":pjmax,\

"yb":yb,"rsb":rsb,"gzb":gzb,"gsb":gsb,"pjb":pjb,\

"ye":ye,"rse":rse,"gze":gze,"gse":gse,"pje":pje})

# print(data)

# 转换 pandas data frame 到 Orange table, 这个 table不包含 target 列，即标签列，它是离散型的

# Orange要求所有建模变量都是数字或离散型的，对于字符串会自动判断是离散型还是meta型，具体见转换函数。

# meta型元数据列类似ID列等标识或附加说明数据的列， 比如文本内容，会以单独的矩阵存储，不参与建模。

# 这个转换函数会自动处理字符与数字列，判断是元数据meta型还是离散型，但认为没有target列

# 离散型即R语言中的因子factor类型，Python中以tuple元组类型实现。

# 先用Orange自带的转换函数转换，后面再补上target列

from Orange.data.pandas\_compat import table\_from\_frame

otable = table\_from\_frame(data)

# print(otable)

# 补上指示是否有风险的标签列

from Orange.data import DiscreteVariable

from Orange.data.table import Table

from Orange.data import Domain

import numpy as np

# print(isRisk)

# 定义一个分类的 target变量， 是否有风险的标签，相当于R语言中的factor因子类型。

target = DiscreteVariable("isRisk",["False","True"])

# 新建一个domain， 补上 target标签

domain2= Domain(otable.domain.attributes,target,otable.domain.metas)

# 新建一个带 target变量的table

otable2 = Table.from\_numpy(domain2,otable.X,Y=np.array(isRisk),\

metas=otable.metas,W=otable.W)

otable2.name = "个税汇总"

# print(otable2)

# 删除第一个表回收内存空间，对于千万条记录级别的大表很有必要。

del otable

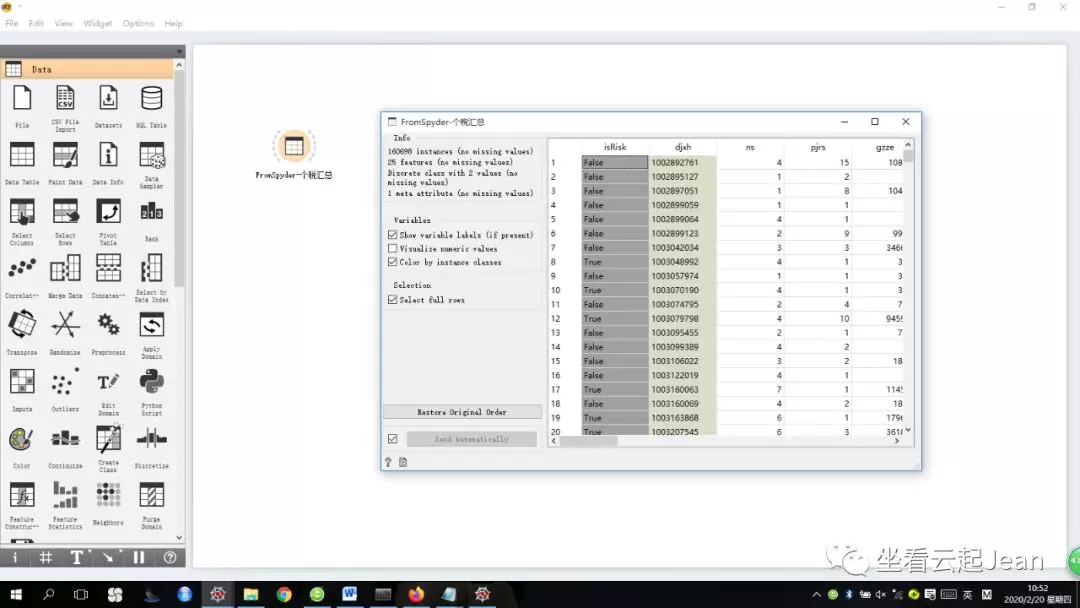
# 调用 Orange，把经过预处理的待分析数据传递过去，日志记录在指定的日志文件中

import sys

from Orange.canvas import \_\_main\_\_ as OrangeMain

sys.exit(OrangeMain.main(datasets = [otable2], logfile="orange.log"))

运行结果：



10、Orange需要改进的地方。

A、可以把这个传入数据集的功能作为标准功能添加进去。

B、增加国际化的支持，象Spyder一样，可以选择界面支持的语言。

C、Text组件增加国际化的支持，改善对中文等不以空格分词的非字母语言的支持（橙现智能已经开发了汉化版）。

这样就可以成为更通用的成熟产品。开源社区的一个原则是，从开源社区得到了帮助，要回馈开源社区，以促进社区的发展。经联系斯洛文尼亚卢布尔雅那大学(Univerza v Ljubljani)的生物科学实验室沟通，开发团队不同意合并这个功能到Orange发行版中，应该是有他们的考虑。

11、我也开发了一个Widget，从Orange中调用Glueviz并传入数据作可视化的探索，这样Spyder、Orange、Glueviz就紧密集成在一起了。具体可看我的[GitHub项目文档](https://github.com/icejean/orange3-glueviz)。只下载[setup.py](https://github.com/icejean/orange3-glueviz/blob/master/setup.py)及./glueviz目录，从源码安装即可，其它文件是其它项目的，不相关。

三、 Rstudio Rattle离线安装

注意，这是旧文档，附上仅作参考。

（一）R环境安装。最新版3.2.0已经合并32位与64位版本。国内镜像下载地址： <http://mirror.bjtu.edu.cn/cran/>，Widows XP 安装32位R，它的最大可用内存是3.2G；Win7、Win8安装64位，它的最大可用内存没有限制。可以用memory.limit()函数看看最大可用内存以确定Rstudio启动的R版本。

（二）Rstudio安装

最流行的R集成开发环境，最新版本RStudio Desktop 0.99.467，没有32位64位之分，下载地址，下Windows desktop 免费版：

<http://www.rstudio.com/products/rstudio/download/>

（三）安装ROracle等常用包。通过OCI访问oracle，需要用 oracle instant client。ROracle包需要自己用Rtools工具编译。我已编译好1.1-11版，包括32位及64位的。PL/SQL+INSTANTCLIENT的安装配置请看Oracle的文档。

Oracle已提供编译好的64位1.1-12版下载，地址：<http://www.oracle.com/technetwork/database/database-technologies/r/roracle/downloads/index.html>

A.下载 instant client 11.2 windows SDK，解压到instant client目录下

B.下载安装 R的 DBI包

C.下载安装 ROracle ，Windows XP装我编译好的1.1-11.zip版（32位），Win7、8装Oracle站上编译好的1.1-12.zip版（64位）。

（四）Rattle安装

1. 安装最新版R3.6.1，Windows7、10上只装64位的版本。
2. 安装相匹配的编译工具Rtools3.5，在默认的目录C:\Rtools下，很多扩展包从源码编译安装时，会指定到这个目录找gcc编译器。
3. 安装Gtk2运行时，gtk2-runtime-2.22.0-2010-10-21-ash.exe。
4. 安装Ggobi，ggobi-2.1.10a.exe。
5. 更改系统环境变量path，加入C:\Rtools\bin;C:\Program Files (x86)\GTK2-Runtime\bin;C:\Program Files\ggobi，以便R及Rstudio可以找到它们。更新rJava引用的路径至新的安装路径D:\R-3.6.1\library\rJava\libs\x64。
6. 更改Rstudio默认的R语言引擎到R3.6.1。
7. 笔记本电脑上先在线装好Rattle环境，再通过R语言脚本批量下载所有扩展包的源码。下载安装脚本及扩展包清单见后面附件。
8. 源码拷贝到内网PC上，再通过R语言脚本批量编译安装。
9. 软件包统计：R3.6.1自带软件包31个，Rattle直接依赖或引用扩展包64个，其他间接依赖或引用扩展包200多个，共306个。
10. 特殊处理的扩展包。具体见脚本，有些软件包在Windows7上编译有问题，因为它要联网下载资源文件，或者找不到相关C的 .h头文件，直接从CRAN上下载已经编译好的Windows二进制文件安装，其中RGtk2\_2.20.36使用中有问题，但RGtk2\_2.20.35版在Windows7上编译不了，直接copy笔记本Windows10上编译好的RGtk2\_2.20.35版使用（复制./library/RGtk2目录到相同目录下）。

#下载软件包

getPackages <- function(packs){

packages <- unlist(

tools::package\_dependencies(packs, available.packages(),which=c("Depends", "Imports"), recursive=TRUE)

)

packages <- union(packs, packages)

packages

}

#Rattle及其依赖的包，64个

myPackages <- c("rattle","sqldf")

packages <- getPackages(myPackages)

packages1<-as.data.frame(packages)

#笔记本上已装好的Rattle环境，共306个包

packages2<-library()

packages2<-as.data.frame(packages2$results)

#从CRAN上下载这些包的源码

download.packages(packages2$Package, destdir="D:/R/RattleSource", type="source")

#成功下载的包清单，共287个

packages3<-read.csv("d:/temp/rattle\_packages.csv",header=FALSE,sep="",row.names = 1)

packages3$V3<-packages3$V5<-NULL

#没找到的包清单，共19个，多数是R语言基本环境自带的包（3.6有31个）

packages4<-sqldf("select \* from packages2 where Package not in(select V2 from packages3)")

#输出三类包清单,在安装时读入批量安装。

write.csv(packages3,"d:/temp/rattle\_downloaded.csv")

write.csv(packages4,"d:/temp/rattle\_notfound.csv")

write.csv(packages1,"d:/temp/rattle\_dependencies.csv")

#离线安装

library(tools)

#Rattle及相关包源码目录

path<-"D:/Rsource/RattleSource"

#生成包索引，以便自动安装依赖的包

write\_PACKAGES(path,type="source",verbose = TRUE)

#这几个包在Windows7上编译有问题，直接安装已编译好的Windows二进制版，

#其中RGtk2\_2.20.36使用中有问题，但RGtk2\_2.20.35版在windows7上编译不了，直接copy笔记本Windows10上编译好的RGtk2\_2.20.35版使用。

install.packages("D:/Rsource/RattleSource/stringi\_1.4.3.zip",repos = NULL,type="win.binary",INSTALL\_opts = "--no-multiarch")

install.packages("D:/Rsource/RattleSource/XML\_3.98-1.20.zip",repos = NULL,type="win.binary",INSTALL\_opts = "--no-multiarch")

#install.packages("D:/Rsource/RattleSource/RGtk2\_2.20.35.tar.gz",repos = NULL,type="source",INSTALL\_opts = "--no-multiarch")

#install.packages("D:/Rsource/RattleSource/RGtk2\_2.20.36.zip",repos = NULL,type="win.binary",INSTALL\_opts = "--no-multiarch")

#自动编译安装依赖的包

install.packages("rattle",contriburl=paste("file:",path,sep=""),type="source",INSTALL\_opts = "--no-multiarch")

install.packages("sqldf",contriburl=paste("file:",path,sep=""),type="source",INSTALL\_opts = "--no-multiarch")

library(rattle)

library(rggobi)

#Rattle汉化有问题，设置语言环境为英文，

load("d:/temp/GeShuiDaShuJu2.RData")

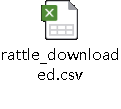
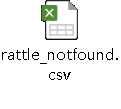
Sys.setenv(LANGUAGE="en")

rattle(dataset="riskindexs")

#安装连接数据库的包

install.packages("D:/Rsource/RattleSource/ROracle\_1.2-1.tar.gz",repos = NULL,type="source",INSTALL\_opts = "--no-multiarch")

install.packages("D:/Rsource/RattleSource/RODPS.tar.gz",repos = NULL,type="source",INSTALL\_opts = "--no-multiarch")

R语言另有一个图形界面Rcommander可以交互式的探索分析数据，统计、作图等，有兴趣的话可以安装了解一下。

四、Rstudio Linux Server 安装配置

**说明：这是篇2016年初写的旧文档，虽然旧一点，有需要的人还是可以参考一下。全国或全省的顶层规划落地实施的周期比较长，对于基层的一些需求，有时候远水救不了近火，这种开源的中小型数据挖掘解决方案可以作为过渡方案备用，解决眼前急需的问题，除了人工，没有额外的开销，也不影响统一的顶层规划。另外也为大量基于J2EE的现有业务系统提供了扩展与集成深度数据分析能力的途径，落地应用效益较高，可操作性强。我装在内网华为虚拟机上的Rstudio Linux Server，现在还在用着。具体与J2EE的集成开发，请参阅第五章《社保明细数据分析》第四节《扩面分析：个税社保比对模型》的第九小节《Rstudio Linux Server与J2EE的集成》。**

在互联网+大数据的潮流下，R语言以其开源免费和强大全面，成为热门的统计分析、数据挖掘工具，迅速在全球各领域普及。Rstudio是其中一个优秀的集成开发环境，提供了Windows上的桌面版及Linux上的服务器版。其中Linux server版通过浏览器使用，客户端无须安装其它软件，极为简便，也便于集中安装管理软件包，是理想的中小型团队网络协作开发环境。另外，安装在Linux上的R，通过Rserve对外提供基于tcp/ip的服务，可在J2EE环境中调用R脚本，支持并发的访问，提供了中小型规模的统计分析、数据挖掘生产环境。

Rstudio linux server如果安装在小型机或虚拟机上，配置较多的CPU及较大的内存，则成为具有较强处理能力的生产环境。

本文通过安装配置Rstudio linux server，对此作了实际测试，并通过一个运行在Tomcat上的实例，演示了J2EE与 R通过Rserve的集成。具体章节如下：

1. 安装Redhat enterprise linux server7.2。
2. 配置Redhat提供telnet服务，以便远程登录管理服务器。
3. 配置Redhat提供ftp服务，以便上传安装R有关的软件包，上传下载R脚本，运行产生的分析结果等。
4. 安装Oracle instantclient linux版，以便访问Oracle数据库。
5. 在Redhat上从源码编译安装R。
6. 安装ROracle以便在R中连接Oracle数据库。
7. 安装配置Rstudio server linux。
8. 安装其它常用的R软件包，形成完整的R开发环境。
9. 安装配置Rserve，提供对R脚本的调用。
10. 安装配置Tomcat，发布测试一个通过Rserve调用R的J2EE应用程序。

（一）linux环境安装

1. 制作U盘安装介质
2. 下载Redhat enterprise linux server7.2的安装介质映象文件rhel-server-7.2-x86\_64-dvd.iso及其安装指南Red\_Hat\_Enterprise\_Linux-7-安装指南.pdf。下载地址：

<http://pan.baidu.com/s/1gdrLC8F#path=%252Fredhat_linux>

1. Windows上下载安装生成U盘安装介质映象的工具Fedora liveusb-creator，按照安装手册第2.2.2节的说明生成U盘安装介质映象。
2. 安装Redhat enterprise linux server7.2

从U盘引导启动，进入安装程序。选择第一项“install redhat enterprise linux 7.2”进入安装，这时安装程序会报错，找不到安装源及/root，安装程序会进入紧急模式。

#cd /dev

# ls –l

看/dev/sdb后的数字（sda一般是硬盘），比如/dev/sdb1，这里sdb1就是U盘所在分区。重新引导系统进入安装程序，选择“press tab for full configuration options on menu items”, 将 “vmlinuzinitrd=initrd.img inst.stage2=hd:LABEL=CentOS\x207\x20x86\_64 quiet”改为 “vmlinuzinitrd=initrd.img inst.stage2=hd:/dev/sdb1 quiet”，这里sdb1为U盘所在分区，具体盘符可能有所不同，可根据实际情况修改，然后按enter执行修改，正常情况下，将进入安装界面。

注意，每台电脑的usb的device的名称都不一样，我的是在/dev/sdb1。如果不确定usb的device名，可以将文字改为

>vmlinuz initrd=initrd.img linux dd quiet

,回车后，屏幕会出现选择的的界面，可以一个个去试。

具体参考：

<http://www.2cto.com/os/201411/354914.html>

<http://blog.csdn.net/gaohuaid/article/details/38750283>

<http://www.centoscn.com/CentOS/2014/0903/3667.html>

选择图形界面安装，按提示作出选择。

注意：

1. 网络主机名，安装完后，要把网络主机名对应IP的解释加入/etc/hosts文件中，比如：

Rserver 150.28.85.189

Oracle instantclient解释TNS服务名时需要解释主机名到IP。

1. 软件安装选择“GUI服务器”，很多软件安装程序只安装软件库文件，而不会安装它们的C头文件，R中很多软件包安装编译时会提示找不到头文件，到时要用yum工具补装需要的软件包及其头文件包。
2. 安装目标选择要安装的硬盘，一般选择自动分区。
3. WIFI是即插即用的，可以以后再装。R的作图软件包rCharts需要在线安装。

设置好后执行安装即可。安装完后，要创建一个除root外的用户用于测试。

1. 为yum设置本地源到U盘

U盘的安装映像中包含了大量的软件包及其头文件，yum安装工具可以解决包之间的依赖关系，自动安装相关的包，非常好用。

<http://jingyan.baidu.com/article/1e5468f90fd265484961b7bf.html>

1. 在 /etc/yum.repos.d/中删除原来repo文件，新建一个扩展名为.repo的文件，文件名自定义，如mylocal.repo。

内容如下，注意替换baseurl中文件路径：

vim /etc/yum.repos.d/mylocal.repo

[](http://jingyan.baidu.com/album/1e5468f90fd265484961b7bf.html?picindex=2)

文件内容含义解释：

**[]：**

中括号中的是repository id，id可以随意命名，不过要注意的是不能存在相同的id，因为id是用来标识不同容器的；

**name：**

后接repository name，用来说明容器，随意命名；

**baseurl：**

repository的地址，如果是网络地址，就用http://，如果本地地址，就用file://。我们这里用的就是本地地址。注意上面的之所以是三个“///”，是因为第三个“/”表示根目录。

**enable：**

表示这个容器是否启用。启用就设置为1，不启用就设置为0。

**gpgcheck：**

表示是否检查rpm文件的数字签名。检查就设置为1，不检查就设置为0。

**gpgkey：**

就是数字签名的公钥文件所在位置。如果gpgcheck值为1，此处就需要指定gpgkey文件的位置，如果gpgcheck值为0 ，就不需要此项了。上面的gpgcheck设置为0，此处可以没有gpgkey。

U盘即插即用，本例中本地U盘软件源的位置是：

/run/media/root/LIVE/Packages

1. 修改yumRepo.py 文件

命令如下：

vim /usr/lib/python2.6/site-packages/yum/yumRepo.py

找到如下行

remote = url + '/' + relative

修改为

remote = "/run/media/root/LIVE/Packages " + '/' + relative

如下图所示:

[RHEL6设置本地YUM源](http://jingyan.baidu.com/album/1e5468f90fd265484961b7bf.html?picindex=3)

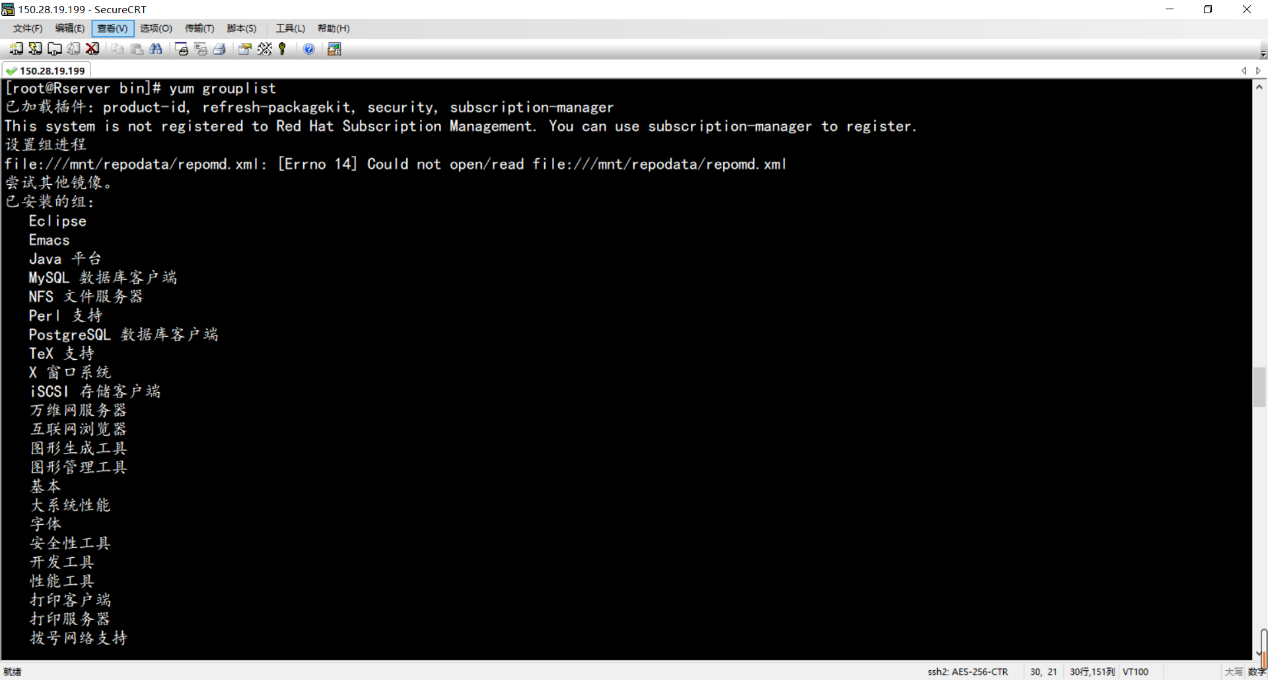
至此，配置就完成了。接下来测试一下配置是否成功。

1. 测试配置

[命令如下：](http://jingyan.baidu.com/album/1e5468f90fd265484961b7bf.html?picindex=5" \t "_self)

[yum grouplist](http://jingyan.baidu.com/album/1e5468f90fd265484961b7bf.html?picindex=5" \t "_self)

[成功返回组列表，表示配置成功，如下图所示：](http://jingyan.baidu.com/album/1e5468f90fd265484961b7bf.html?picindex=5" \t "_self)



1. 安装配置telnet

<http://www.linuxidc.com/Linux/2015-02/113957.htm>

1)开终端并输入下面的命令来安装telnet：

# cd /run/media/root/LIVE/Packages

# ls –l |grep telnet

# yum install telnet telnet-server –y

2)让这个服务在每次重启时都会启动：

#systemctl start telnet.socket

#systemctl enable telnet.socket

3)打开防火墙23号telnet端口

#firewall-cmd --permanent --add-port=23/tcp

#firewall-cmd–reload

**4)安装telnet客户端**

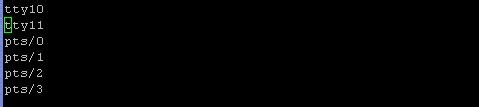
#yum install telnet

1. 本机测试telnet

#telnet 150.28.85.189

1. 其它机器测试telnet
2. 配置root可以telnet，Redhat默认禁止root telnet

**修改配置文件 /etc/securetty，**在文件末尾添加pts/0……字样，如图所示：



<http://jingyan.baidu.com/article/c74d6000668e440f6a595df3.html>

**为了安全，此功能慎用，获取root功能的方法有：**

A、su root，切换到root用户，如果知道密码；

B、sudo 加命令，如 sudo vi /etc/securetty

C、sudosu root，暂时获得root权限

8）Windowstelnet客户端

Putty，支持汉字显示，不用做设置。

Xwinlogon, windows上的X window 图形界面终端

它们设置linux服务器的IP就可以连上。

1. 配置ftp，用vsftpd

Rstudio server可以直接在浏览器界面上传下载个人使用的文件，这里配置ftp server主要是用于放置公用的文件，比如一些工作中常用的较大的软件。

<http://www.wyzc.com/forum/57649.html>

<http://yuanbin.blog.51cto.com/363003/129071/>

Redhat7.2 使用了SELinux来增强安全，但是使用起来有时候太繁琐，关闭它以免影响ftp连接。修改/etc/selinux/config文件中的SELINUX=””为 disabled ，然后重启。

<http://blog.chinaunix.net/uid-26727991-id-3235275.html>

* 1. 安装

# cd /run/media/root/LIVE/Packages

//使用yum安装

# yum –yinstall ftp vsftpd

//或者使用rpm安装以下两个包

ftp-0.17-66.el7.x86\_64

vsftpd-3.0.2-9.el7.x86\_64

//另外需要安装db包，用来加密虚拟用户的账户信息

//这个包在centos7中默认已经安装了

# rpm -qf/usr/bin/db\_load

libdb-utils-5.3.21-17.el7\_0.1.x86\_64

1. 配置

# mkdir /ftp/open

做几个测试文件

# echo opentest> /ftp/open/open.txt

# echo filetest> /tmp/filetest1.txt

# touch /ftp/open/anontest.txt

//查看配置文件所在路径

# rpm -qc vsftpd

/etc/logrotate.d/vsftpd  
/etc/pam.d/vsftpd  
/etc/vsftpd/ftpusers  
/etc/vsftpd/user\_list  
/etc/vsftpd/vsftpd.conf

# cd /etc/vsftpd/

//备份原有配置文件

# cp vsftpd.conf vsftpd.conf.origin

//创建密码明文文件,格式：用户名一行，密码一行

# vi /etc/vsftpd/vftpuser.txt

alice  
[P@ssw0rd](mailto:P@ssw0rd)  
jack  
[P@ssw0rd](mailto:P@ssw0rd)  
tom  
[P@ssw0rd](mailto:P@ssw0rd)  
  
//根据明文创建密码DB文件

# db\_load -T -t hash -f /etc/vsftpd/vftpuser.txt \

/etc/vsftpd/vftpuser.db  
  
//查看密码数据文件

# file /etc/vsftpd/vftpuser.db

/etc/vsftpd/vftpuser.db: Berkeley DB (Hash, version9, native byte-order)

//创建vftpd的guest账户

# useradd -d/ftp/private -s /sbin/nologinvftpuser

# vi /etc/pam.d/vsftpd

将auth及account的所有配置行行均注释掉，添加如下内容：

auth required pam\_userdb.so db=/etc/vsftpd/vftpuser

account required pam\_userdb.sodb=/etc/vsftpd/vftpuser

//打开配置文件

# vi /etc/vsftpd/vsftpd.conf

//在最后添加

anon\_root=/ftp/open  
virtual\_use\_local\_privs=YES  
guest\_enable=YES  
guest\_username=vftpuser  
chroot\_local\_user=YES  
allow\_writeable\_chroot=YES  
//设置自动启动

# systemctl enable vsftpd

ln –s'/usr/lib/systemd/system/vsftpd.service''/etc/systemd/system/multi-user.target.wants/vsftpd.service'  
  
# systemctl start vsftpd

//查看目前的状态

# systemctl status vsftpd.

1. **测试**

//测试匿名账户

# ftp localhost

ftp> ls.

ftp>cd /tmp

ftp> getopen.txt

ftp> bye

//测试本地账户

# ftp localhost

ftp> ls

ftp> ls /tmp

ftp>cd /tmp

ftp> bye

221 Goodbye.

4）远程ftp连接

打开防火墙21号ftp端口

#firewall-cmd --permanent --add-port=21/tcp

#firewall-cmd –reload

Cuteftp 使用passive port方式连接。

1. 安装Oracle instantclient
   1. 下载 basic-linux-x64及对应的sqlplus，devel软件包。

下载地址：

<http://www.oracle.com/technetwork/topics/linuxx86-64soft-092277.html>

* 1. 用 rpm -ivh或yum install 安装rpm软件包。

<http://www.educity.cn/wenda/612023.html>

* 1. 修改/etc/profile，加入oracle环境变量：

export ORACLE\_HOME=/usr/lib/oracle/12.1/client64

export LD\_LIBRARY\_PATH=/usr/lib/oracle/12.1/client64/lib:$LD\_LIBRARY\_PATH

export TNS\_ADMIN=$ORACLE\_HOME/network/admin

export PATH=$PATH:$ORACLE\_HOME/bin

export NLS\_LANG = ’SIMPLIFIED CHINESE\_CHINA.UTF-8’

* 1. 配置tnsnames.ora
  2. 在/etc/hosts中加入本机的机器名解释，TNS服务名解释需要解释主机名到IP，如：

Rserver 150.28.85.189

* 1. 用sqlplus测试连接及中文支持。

sqlplus scott/tiger@CACTESTDB

(二)R安装

1. 下载R源码

<http://mirror.bjtu.edu.cn/cran/src/base/R-3/R-3.2.3.tar.gz>

1. 解压

tar –xzvfR-3.2.3.tar.gz

1. 安装

#cd R-3.2.3

#./configure --prefix=”/home/R-3.2.3” --enable-R-shlib --with-tcltk

#make

#make install

说明：

--prefix=”/home/R-3.2.3”，安装R到该目录下。

--enable-R-shlib，Rserve要求该选项打开

--with-tcltk，Rcmdr等图形界面需要。

从./configure运行的结果看R需要的功能，JPEG，tcl/tk，X11这些应该列出，如果不是，要看./configure输出的信息，看linux系统安装时缺了那些库文件及头文件，缺的就用yum install补装，R重新配置编译一下即可。

1. 测试

#cd /home/R-3.2.3/bin

#./R

>capabilities()

这个函数显示R编译好后支持的功能，要求R支持JPEG，X11，tcl/tk，这些项目输出应该是TRUE。

（三）ROracle安装

ORACLE Instant Client安装完后， 相关的头文件在 /usr/include/oracle/12.1/client64/下；库文件在/usr/lib/oracle/12.1/client64/下(client目录下有 lib 和 bin两个目录) ，头文件需要转移到目录/usr/lib/oracle/12.1/client64/rdbms/public下，如果不复制过去，安装ROracle时就提示~/rdbms/public目录不存在， ROracle安装文档没有提及这个问题。

#mkdir -p /usr/lib/oracle/12.1/client64/rdbms/public

#cd /usr/lib/oracle/12.1/client64/rdbms/public

#cp /usr/include/oracle/12.1/client/. .

# cd /home/R-3.2.3/bin

#./R

>install.packages(“/home/ftp/priv/ROracle\_1.1-12.tar.gz”,repos=NULL,type=”source”)

说明：

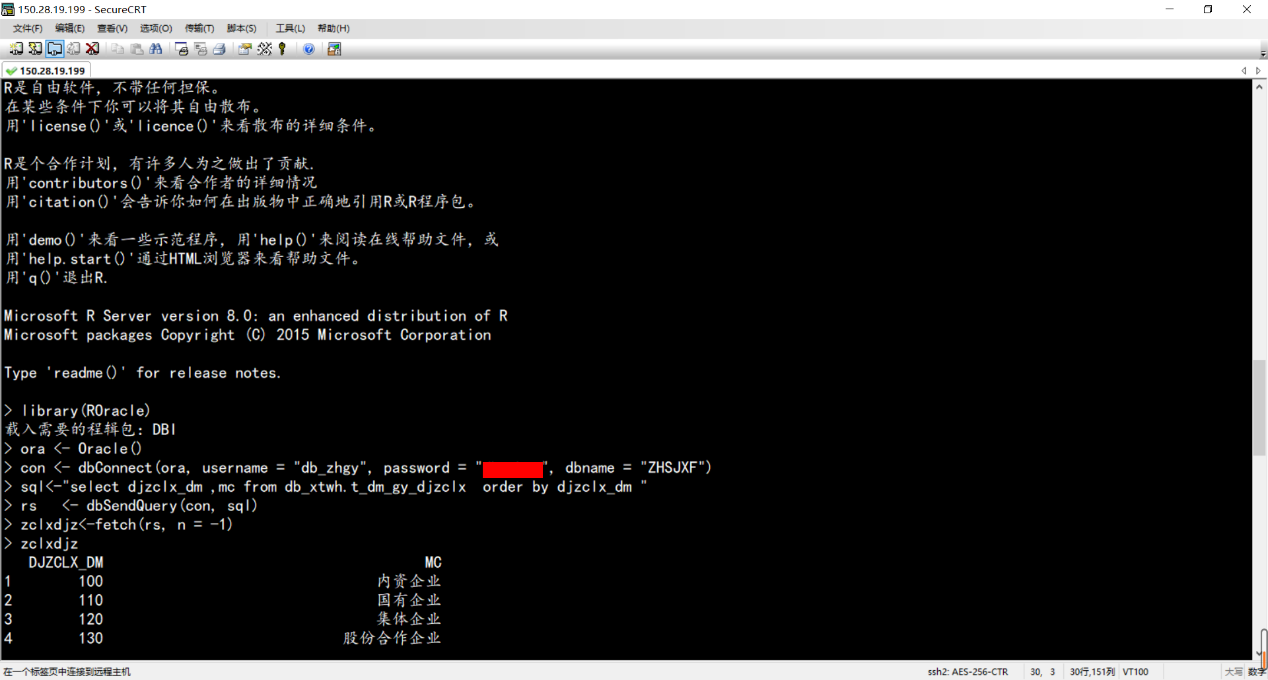
从本地安装R软件包，第一个参数是软件包的路径，第二个参数repos=NULL指明从本地安装，第三个参数type=”source”指明从源码安装。

ROracle也需要设置Oracle的环境变量，在命令行状态下使用R，它会引用shell中设置的环境变量，之前在安装Oracle instantclient时已经在 /etc/profile中设置好了，它使用底层的instantclient动态库。

测试，在R中输入下图中的命令联接Oracle：

#cd /home/R-3.2.3/bin

#./R



（四）Rstudio Linux Server安装

1. 下载地址：

<https://www.rstudio.com/products/rstudio/#Server>

1. 安装

yum install rstudio-server-rhel-0.99.489-x86\_64.rpm

或

rpm –uvh rstudio-server-rhel-0.99.489-x86\_64.rpm

1. 配置

<http://www.07net01.com/program/305533.html>

<http://www.codes51.com/article/detail_112922.html>

* + 1. 指定R版本

默认情况下，RStudio server 运行时使用的R是系统PATH中指定的R，这个位置可以使用which R确定，如果我们的系统上有多个版本的R，那么可以使用rsession-which-r参数指定RStudio 使用哪个版本的R。 如：

rsession-which-r=/home/R-3.2.3/bin/R

在/etc/rstudio/rserver.conf文件里添加这个参数之后，必须重启rstudio server 才能生效。

* + 1. 将 Oracle instantclient 路径加入Rstudio server配置文件，以便可以使用ROracle，不知为何，shell中的环境变量没有带到Rstudio中。

#cd /etc/rstudio

#virserver.conf

加入：

# Server Configuration File

rsession-ld-library-path=/usr/local/lib64/R/lib:/usr/local/lib64:/usr/java/jdk1.6.0\_45/jre/lib/amd64/server:/usr/local/lib:/usr/lib/oracle/12.1/client64/lib

* + 1. 将ORACLE环境变量加入R环境文件，以便Rstudio中调用R时可以找到，不知为何，shell中的环境变量没有带到Rstudio中。注意，$R\_HOME/etc下也有一个Renviron文件，不是改那个文件。

# cd/home/R-3.2.3/lib64/R/etc/

# vi Renviron

ORACLE\_HOME=’/usr/lib/oracle/12.1/client64’

TNS\_ADMIN=’/usr/lib/oracle/12.1/client64/network/admin’

NLS\_LANG = ’SIMPLIFIED CHINESE\_CHINA.UTF-8’

1. 管理

如果使用安装包来安装的RStudio，那么会自动将RStudio 注册到系统了，在系统启动会自动启动。手工命令如下：

# cd /usr/sbin

# ./rstudio-server –help

# ./rstudio-server verify-installation

# ./rstudio-server start&

# ./rstudio-server stop&

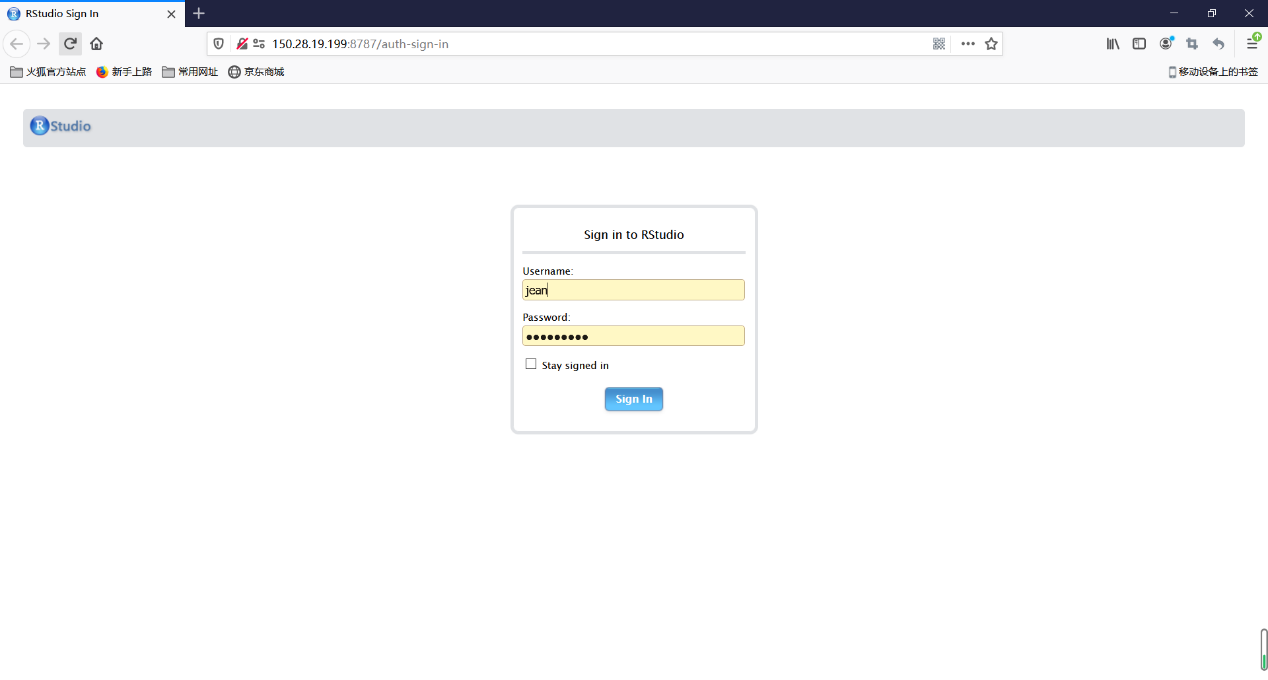
1. 访问

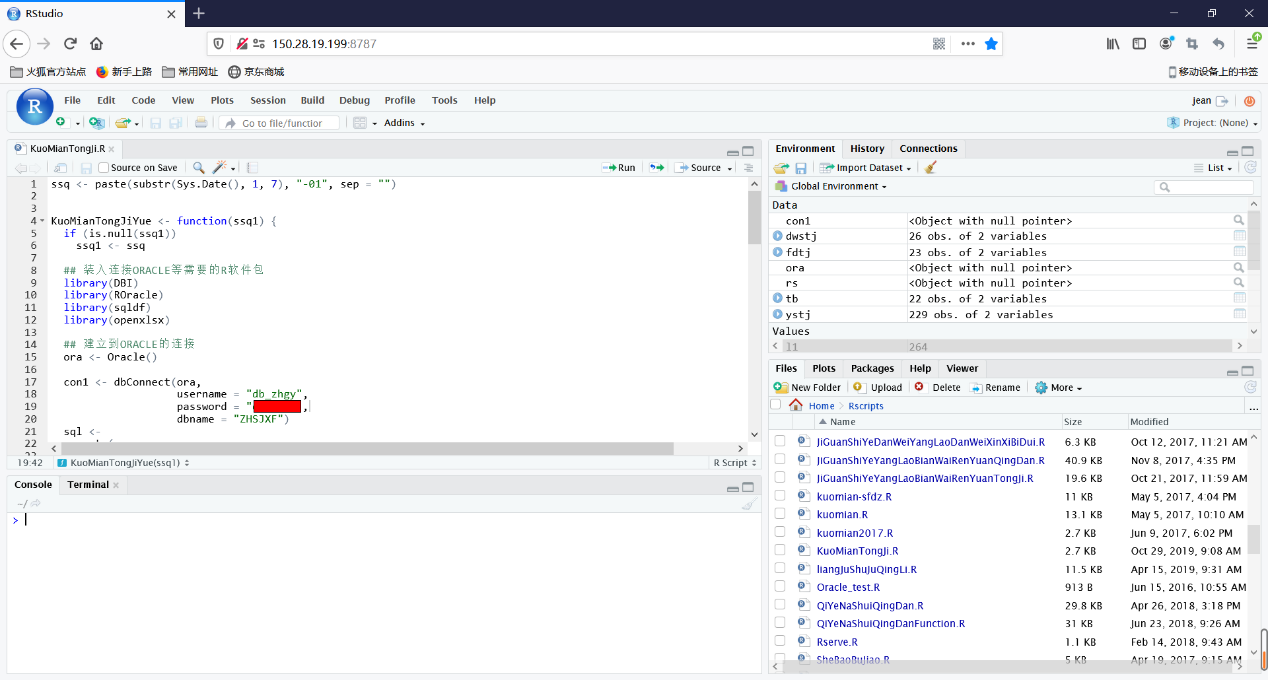
Rstudio server默认在8787端口服务，要改防火墙设置打开该端口：

#firewall-cmd --permanent --add-port=8787/tcp

#firewall-cmd –reload

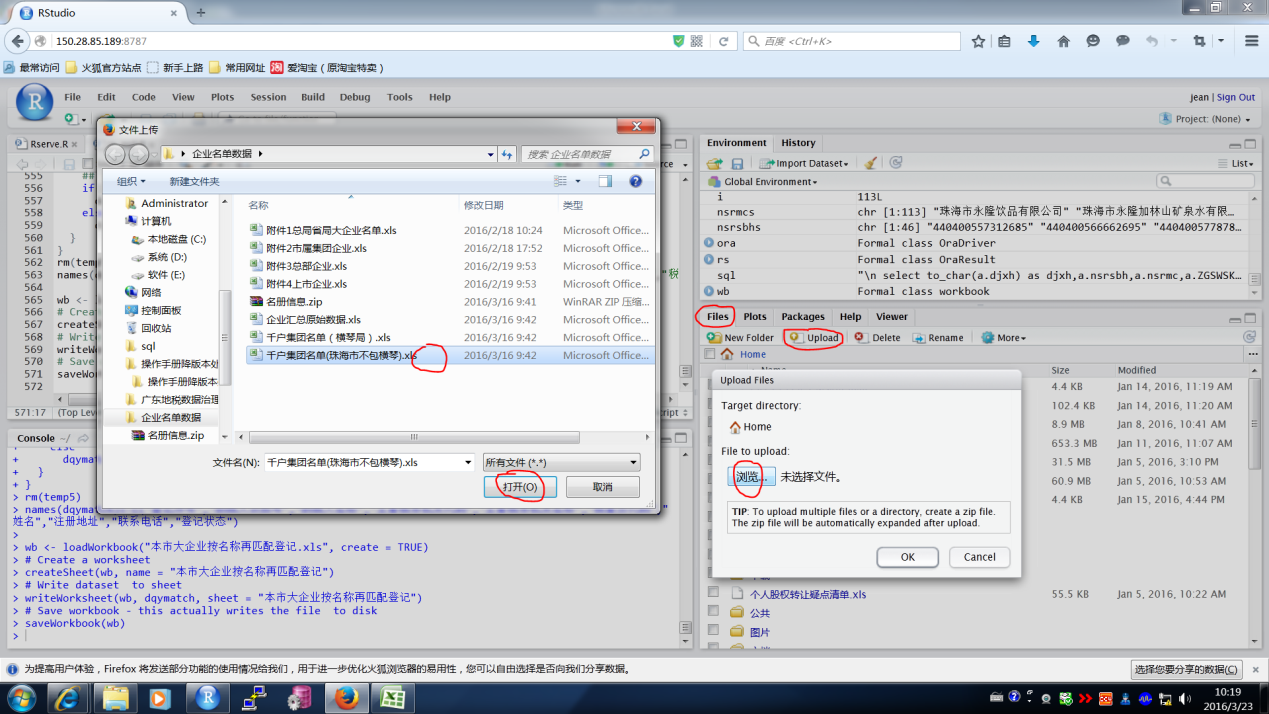
然后用IE10以上或Firefox、Chrome等浏览器访问，用户名密码是Linux系统的用户名和密码。



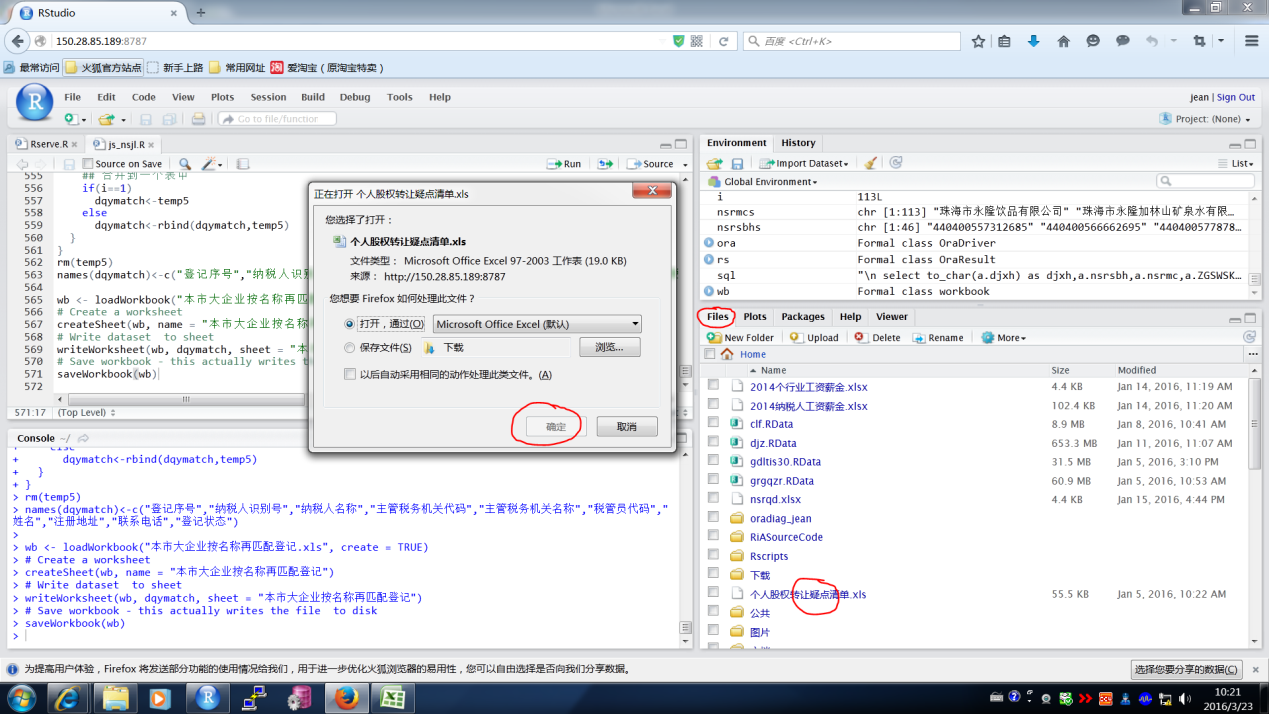


1. Rstudio server中上传输入数据文件，下载输出数据文件

可以向Rstudio server上传数据文件，用作R脚本的输入，上传后放在用户的home目录下，在程序中引用时，加上home目录作为前导路径，比如"/home/jean/mydata.xls"，具体按顺序点击图中红圈标记菜单按钮选择即可。



程序中输出的结果数据文件，一般也放在用户home目录下，比如"/home/jean"，在Rstudio server浏览器中直接点击文件链接下载保存到本地即可，如图所示。



(五)R软件包安装

注意：本节是RHEL6.7上的安装文档，RHEL7.2上可能要作一些调整。

1、 rJava，XLConnectJars，XLConnect，Rjdbc

1）、配置java，

A.先安装 java-1.7.0-openjdk

[root@gdzksjfx01 Packages]# cd /mnt/Packages

[root@gdzksjfx01 Packages]# ls |grep java-1.7.0

java-1.7.0-openjdk-1.7.0.79-2.5.5.4.el6.x86\_64.rpm

java-1.7.0-openjdk-devel-1.7.0.79-2.5.5.4.el6.x86\_64.rpm

用yum查看，确认 devel 包已安装，编译rJava需要它的jni.h等头文件，如果编译时还是报告找不到 jni.h，则把/usr/lib/jvm/java-1.7.0-openjdk-1.7.0.79.x86\_64/include目录下的头文件拷贝到 /usr/include目录下。

另外，编辑 /etc/profile, 输出 JAVA\_HOME环境变量， source /etc/profile使它生效。

# vi /etc/profile

# Added by Jean for Java

export JAVA\_HOME=/usr/lib/jvm/java-1.7.0-openjdk-1.7.0.79.x86\_64

B. 重新配置R使它支持java

# cd /usr/lib64/MRO-for-MRS-8.0.0/R-3.2.2/lib64/R/bin

# R CMD javareconf

C.增加libiuuc等要用到的库链接

# cd /usr/lib64

# ln -ns libicuuc.so.42.1 libicuuc.so

# ln -ns libicui18n.so.42.1 libicui18n.so

2）安装rJava，XLConnectJars，XLConnect，Rjdbc

#R

> .libPaths()

[1] "/usr/lib64/MRO-for-MRS-8.0.0/R-3.2.2/lib64/R/library"

>install.packages("/root/Downloads/R Linux/rJava\_0.9-7.tar.gz",repos=NULL,type="source")

>install.packages("/root/Downloads/R Linux/XLConnectJars\_0.2-9.tar.gz",repos=NULL,type="source")

>install.packages("/root/Downloads/R Linux/XLConnect\_0.2-11.tar.gz",repos=NULL,type="source")

>install.packages("/root/Downloads/R Linux/RJDBC\_0.2-5.tar.gz",repos=NULL,type="source")

D. 上传Sybase JDBC驱动程序jconn2.jar到 /usr/lib/java-ext目录以便sybase jdbc引用。

2、安装proto，gsubfn，RSQLite、chron，sqldf，以使用sqldf。

sqldf包提供了R中通过sql语法操作数据的工具，它依赖于前面的几个包，其中RSQLite是个轻量级的SQL数据库，sqldf默认使用它，所以sqldf中的sql语法是SQLite3的语法，区别于Oracle PL/ SQL的语法，使用时要注意。安装方法同上，按顺序安装即可。

因为上面安装的R软件都是64位的，有些软件包报错不支持 i386或x64模式时，可以增加选项，只编译一种模式。

install.packages("path to file",repos=NULL,type="source",INSTALL\_opts="--no-multiarch")

有些软件包安装编译时要下载资源文件，没有互联网连接装不了，只能下载二进制版本安装。有些软件包需要更改源码中的配置文件选项安装，这时可以解压到一个临时文件夹，改好后用下面的命令装：

install.packages("path to temp dir",repos=NULL,type="source")

对于每一个R软件包，都要在它依赖的软件包装好后，才可能安装成功。可以访问R CRAN的镜像站点，比如http://mirrors.xmu.edu.cn/CRAN，Packages->Table of available packages, sorted by name，找到要安装的软件包，比如ROracle，会列出depends及imports的包，按文档的顺序点击链接可逐层深入下载安装所有depends及imports的包。

(六)Rserve安装

Rserve文档：<http://rforge.net/Rserve/doc.html>

install.packages(“/home/ftp/priv/Rserve\_1.8-5.tar.gz”,repos=NULL,type=”source”)

Linux上安装Rserve，R编译时要加上 --enable-R-shlib选项。每个连接在 /tmp/Rserve下以随机数建立临时文件夹存放该连接的文件，要定期清理。

Windows上，Rserve不支持并发连接，调用代码需要特别仔细编写，以使用一个共享的连接，详见文档：

Although **Rserve** works on Windows it is not recommended to use it on that platform. Windows lacks important features that make the separation of namespaces possible, therefore **Rserve** for Windows works in cooperative mode only, that is only one connection at a time is allowed and all subsequent connections share the same namespace. Briefly: don't use Windows unless you really have to. There is a DCOM interface available for **R**, you may be better off with that solution.

Windows上启动Rserve：

>library(Rserve)

## --RS-encoding utf8, only utf-8 is supported for java client

Rserve(args="--RS-encoding utf8 --RS-enable-remote --RS-enable-control --no-save")

Linux上启动Rserve：

# cd /home/R-3.2.3/bin

# ./R CMD Rserve --RS-encoding utf8 --RS-enable-remote --RS-enable-control --no-save&

（七）、Tomcat安装

1. 修改/etc/profile，输出必要的环境变量

# added for tomcat

export JAVA\_HOME=/lib/jvm/java-1.8.0-openjdk

export CATALINA\_HOME=/usr/share/tomcat

export CATALINA\_BASE=$CATALINA\_HOME

2.安装

<http://wenku.baidu.com/link?url=NLHIPSar8qX68ydEBHdEQwnCyYEgD30QtqOfzW-rIs-A0ofWenLK39JSwU4fvhMG3TtpzLTLMq7atfvwHkw8Zs7aNjfRg7textpv4gUSfc_>

<http://www.linuxidc.com/Linux/2015-01/111289.htm>

JDK在安装Redhat时已装好，用的是OpenJDK，只需下载tomcat linux版：

<http://apache.fayea.com/tomcat/tomcat-7/v7.0.68/bin/apache-tomcat-7.0.68.tar.gz>

# tar–xzvfapache-tomcat-7.0.68.tar.gz

Redhat 7.2上tomcat装在/usr/share/tomcat目录下。

防火墙打开 8080端口：

#firewall-cmd --permanent --add-port=8080/tcp

#firewall-cmd –reload

3.配置管理账户

<http://jingyan.baidu.com/album/358570f66494a5ce4724fce9.html>

# cd /usr/share/tomcat/conf

# vi tomcat-users.xml

在<tomcat-users></tomcat-users>之间添加下面语句

<role rolename="admin-gui"/>

<role rolename="manager-gui"/>

<user username="tomcat" password="tomcat" roles="admin-gui"/>

<user username="admin" password="1234" roles="manager-gui"/>

其中用户名为tomcat，密码为tomcat的是用来登录tomcat的Host Manager的，而用户名为admin，密码为1234是用来登录tomcat的App Manager的。

4.管理

Usage: /sbin/tomcat {start|start-security|stop|version}

#cd /usr/sbin

# ./tomcat start&

#./tomcat stop&

5.发布webapp到tomcat

1）访问tomcat管理页面，http://150.28.85.189:8080

->点击 "manager app"

输入前面配置的管理员用户名/口令

2）找到WAR file to deploy，点击"浏览"，选中在本机eclipse中打包生成的web应用WAR文件，按“deploy”按钮上传发布。

3）然后在上面 Applications 列表中可以看到发布的应用状态，path一列给出了访问的URL，如/helloR，在浏览器输入即可访问（直接点击链接也可以），如：

http://150.28.85.189:8080/helloR

（八）、为Rstudio server增加用户

增加linux系统用户及vsftp用户即可：

1、增加linux系统用户，假设主组是rusers，ftp用户组为vftpuser

#useradd username -g rusers -G vftpuser

#passwd

2、增加vsftp用户，编辑vftpuser.txt增加用户名及密码，重新生成密码文件。

# vi /etc/vsftpd/vftpuser.txt

# db\_load -T -t hash -f /etc/vsftpd/vftpuser.txt /etc/vsftpd/vftpuser.db

//查看密码数据文件

# file /etc/vsftpd/vftpuser.db

/etc/vsftpd/vftpuser.db: Berkeley DB (Hash, version9, native byte-order)

（九）设置Rstudio server访问权限

[root@gdzksjfx01 sysconfig]# cd /etc/rstudio

[root@gdzksjfx01 rstudio]# ls

rserver.conf rsession.conf

[root@gdzksjfx01 rstudio]# cat rserver.conf

# Server Configuration File

rsession-which-r=/usr/lib64/MRO-for-MRS-8.0.0/R-3.2.2/lib64/R/bin/R

rsession-ld-library-path=/usr/local/lib64:/usr/local/lib:/usr/lib/jvm/java-1.7.0-openjdk-1.7.0.79.x86\_64/jre/lib/amd64/server:/usr/local/lib64/R/lib:/usr/lib/oracle/12.1/client64/lib

# Only users in group rusers can log into RStudio Server

auth-required-user-group=rusers

这样设置后只有rusers组的用户才能登录 rstudio server。

（十）设置开机自动启动Rserve及Tomcat

修改/etc/rc.d/rc.local,加入自启动服务需要使用的环境变量，/etc/profile中的环境变量要登录后才生效，自启动服务没有登录。然后让Rserve开机时作为自启动进程启动。Rstudio Server开机自启动脚本在/etc/init/rstudio-server.conf中，同样需要加入下述Oracle与Java环境变量。

# Added by Jean for Oracle

# Add environment varables for self-starting services

export ORACLE\_HOME=/usr/lib/oracle/12.1/client64

export LD\_LIBRARY\_PATH=/usr/lib/oracle/12.1/client64/lib:/usr/local/lib:$LD\_LIBRARY\_PATH

export TNS\_ADMIN=$ORACLE\_HOME/network/admin

export PATH=$PATH:$ORACLE\_HOME/bin

export NLS\_LANG="SIMPLIFIED CHINESE\_CHINA.AL32UTF8"

export LC\_ALL=zh\_CN.UTF-8

export LANG=zh\_CN.UTF-8

# Added by Jean for JAVA

export JAVA\_HOME=/usr/lib/jvm/java-1.7.0-openjdk-1.7.0.79.x86\_64

export CATALINA\_HOME=/usr/share/apache-tomcat-7.0.84

export CATALINA\_BASE=$CATALINA\_HOME

export RPICDIR=/tmp/Rpics

#Startup Tomcat

/usr/share/apache-tomcat-7.0.84/bin/startup.sh

#Startup Rserve

# change to R home to start Rserve, or Rserve can't find ROracle

cd /usr/lib64/MRO-for-MRS-8.0.0/R-3.2.2/lib64/R/bin

./R CMD Rserve --RS-encoding utf8 --RS-enable-remote --RS-enable-control --no-save

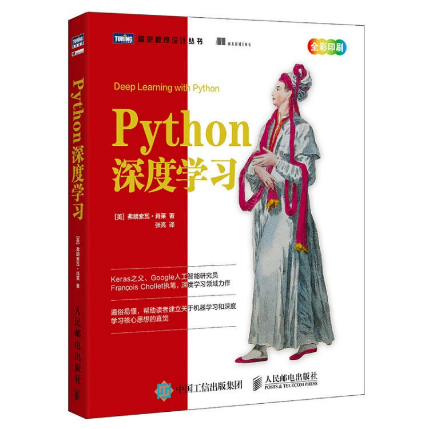
五、Keras与TensorFlow安装

可能要用到的大数据分析技术中剩下神经网络与深度学习这个板块了，趁现在闲着了解一下。行到水穷处，坐看云起时，读书自学的外溢影响应该是最小的，学习自娱总可以吧。之前用Hanlp实现发票货物劳务名称识别时，Hanlp2.0用的是Tensorflow，当时自费升级了工作用的台式机，安装了Nvidia的GeForce GTX 1060GPU（参阅第一章《GPU加速》一节，[Hanlp2.1](https://www.hanlp.com/)增加了PyTorch框架，因Tensorflow的预训练模型体量较大，不便于发行），所以现在还是用Win10 + Tensorflow + GTX 1060。这个Google的开源的框架下，Keras在Tensorflow之上提供了更高级的API，可以简化相关的开发，并且支持CNTK、Theano等不同于Tensorflow的后端。另外，Rstudio的创始人J.J.Allaire也为Keras和Tensorflow开发了R语言的接口包，所以Python与R都可以使用。虽然R语言也有Apache MXNet等其它深度学习框架，但用同一个框架，双语全栈适应性更广，更具优势，所以选择了这个学习方向。具体生产平台用什么技术框架，那是没有什么影响的，我只是学习了解一下相关的技术，看看有什么落地应用的可能性而已。

没有必要在Python与R之间纠结争论，哪个用着顺手就行了，双语全栈，没问题。应该把精力放在具体落地应用场景的发现与开发上，同一个模型翻译成另一种语言并不困难。

（一）、教材

1、《Python深度学习》。这是Keras作者François Chollet写的书，他也是Tensorflow的重要贡献者之一，所以这本书是比较经典的。



2、《R语言深度学习》，这本书是前者的R语言版本，由J.J.Allaire与François Chollet合作完成。R语言在深度学习方面的书不多，这本书年初刚翻译出版，是比较及时的。



（二）、安装配置

1、Python安装配置。因为Keras与Tensorflow都是Python上的，所以要先安装配置好Python的环境，win10上用Anaconda方便一点。

1）Anaconda3离线安装，可以看这篇[文章](https://www.meipian.cn/2vpc52r6?share_depth=1)。

2）Nvidia GPU驱动安装，可以看参阅第一章《GPU加速》一节。需要注意版本的匹配，我的版本组合是：Win10+Python3.7+CUDA10.1+cuDNN7.6+tensorflow\_gpu-2.3.0 。

3）因此现在只需要增加安装Python的keras包。Keras需要2.4.3版，参见这篇[文章](https://blog.csdn.net/weixin_43405513/article/details/109026440)。

pip install keras==2.4.3

可以看到所有依赖包前面都已装好了，只需下载keras-2.4.3离线安装即可。

注意Tensorflow也可以用CPU版本，我的旧笔记本只有4G内存，装的就是CPU版本，但这样深度学习的实例是无法顺畅跑起来的。

2、Rstudio安装配置。我是先在笔记本上装好的，这样就知道需要安装的包与顺序，然后下载并按顺序在台式机上离线安装就好了。注意这些都是R语言接口包，是对相应Python包的封装。

1）在笔记本Rstudio上执行：

install.packages("tensorflow")

可以看到，依次安装了config\_0.3.1.zip，reticulate\_1.20.zip，tfruns\_1.5.0.zip和tensorflow\_2.4.0.zip。

2）执行：

install.packages("keras")

安装的是keras-2.4.0.zip。

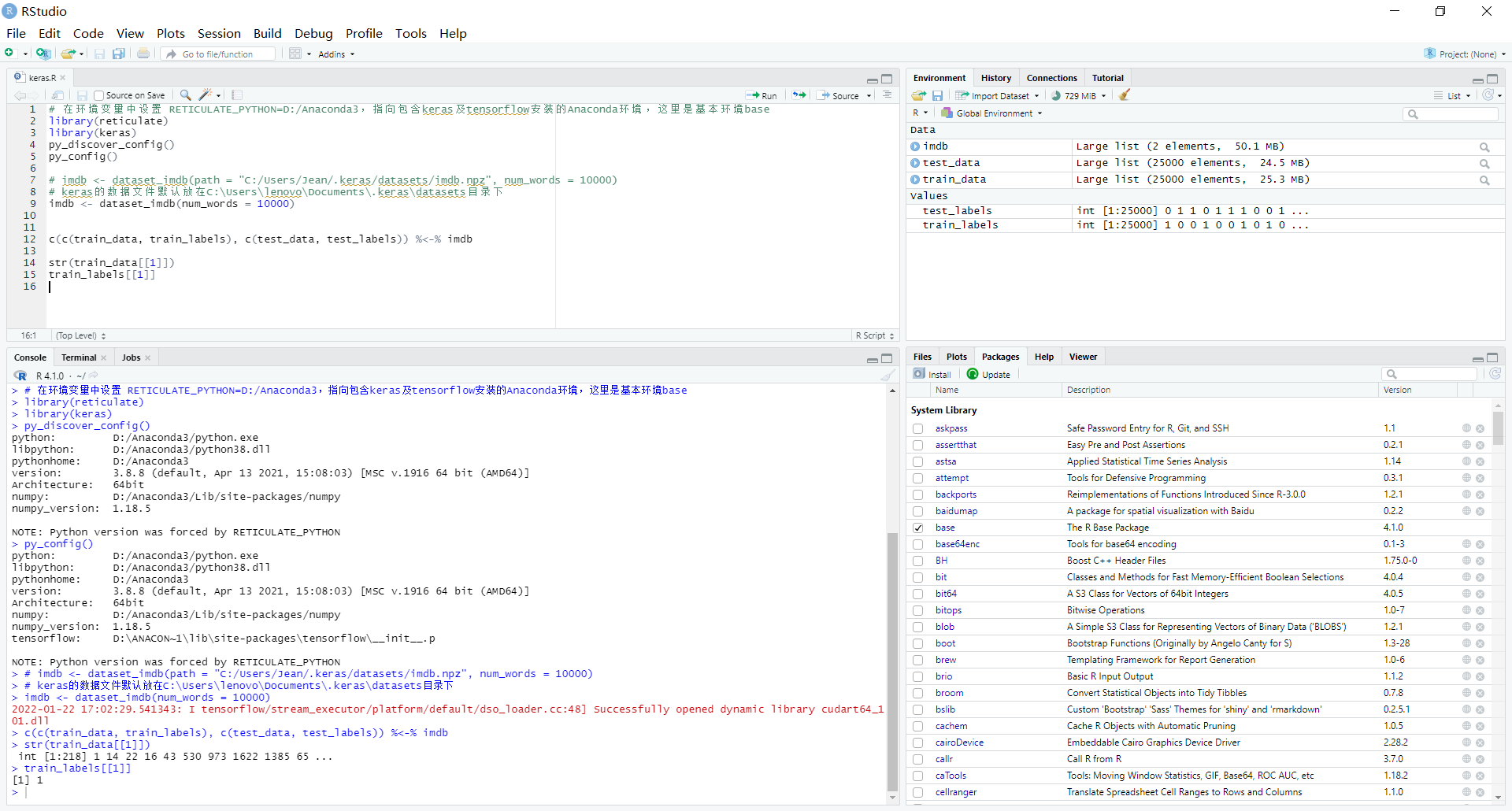
3）装好后配置系统环境变量

RETICULATE\_PYTHON=D:/Anaconda3

指向包含keras及tensorflow安装的Anaconda环境，这里是基本环境base，reticulate就是R语言中调用Python的接口包，具体参看这篇[文章](https://rstudio.github.io/reticulate/articles/versions.html)。

4）然后执行下面的命令确认R语言到Pyhton的调用正常：

library(reticulate)  
library(keras)  
py\_discover\_config()  
py\_config()



（三）、Keras数据集

Keras示例运行时要加载一些数据集，内网不能在线加载，可以先下载好，然后在程序中指出加载路径，见上图加载imdb数据集，加载成功的话表示R语言的环境装好了。

imdb等keras数据集[下载的地址](https://pan.baidu.com/s/1aZRp0uMkNj2QEWYstaNsKQ" \l "list/path=%2F)，提取码: 3a2u ，百度一下有很多。Keras在线下载的数据win10下在~/Documents/.keras/datasets目录下，把数据拷到该目录，则Rstudio中加载数据时可以不指定目录，Python则仍需指定。

（四）、书的代码

1、《R语言深度学习》源码，[下载地址](https://www.manning.com/books/deep-learning-with-r)。代码是jupyter notebook格式的，需要启动Anaconda jupyter notebook来看。参考这篇[文章](https://blog.csdn.net/fanlily913/article/details/106171791)把jupyter notebook启动后的根目录改到源码目录即可。

2、《Python深度学习》源码，[下载地址](https://github.com/fchollet/deep-learning-with-python-notebooks)，同样是jupyter notebook格式。

（五）、运行实例

这里有篇英文的[Python与R语言的同实例对比](https://www.analyticsvidhya.com/blog/2017/06/getting-started-with-deep-learning-using-keras-in-r/)，同一模型不同语言的实现，也说明了不必在Python与R之间纠结。不过我跑的是CSDN上的实例。这些都是识别keras自带mnist数据集手写体数字的经典实例，跑通了就可以验证整个框架安装配置完成了。

1、先跑通[Python上的实例](https://blog.csdn.net/umbrellalalalala/article/details/86309633)。

# -\*- coding: utf-8 -\*-

# 导入这两个是为了获取mnist数据集

import tensorflow as tf

mnist=tf.keras.datasets.mnist

# 下面三行是常规导入

from tensorflow import keras

from tensorflow.keras import layers

from tensorflow.keras import models

#导入mnist数据集，需要保持网络畅通

(X\_train, Y\_train), (X\_test, Y\_test) = mnist.load\_data(r"C:\Users\Jean\.keras\datasets\mnist.npz")

img\_rows, img\_cols = 28, 28 # 图像的尺寸

# 训练数据;X\_train是60000张28\*28的数据，所以尺寸是60000\*28\*28，Y\_train是对应的数字，尺寸是60000\*1，X\_test和Y\_test同理

X\_train, X\_test = X\_train / 255.0, X\_test / 255.0 # 将图像像素转化为0-1的实数

# 将标准答案通过one-hot编码转化为需要的格式（这两个函数不懂见下个模块的介绍）

Y\_train = keras.utils.to\_categorical(Y\_train, num\_classes=10)

Y\_test = keras.utils.to\_categorical(Y\_test, num\_classes=10)

# 将训练所用的图像调整尺寸，由于图像是黑白图像，所以最后一维的值是1

X\_train = X\_train.reshape(X\_train.shape[0], img\_rows, img\_cols, 1)

X\_test = X\_test.reshape(X\_test.shape[0], img\_rows, img\_cols, 1)

###############################

### 使用keras API开始定义模型 ###

###############################

model = models.Sequential()

# 向模型中添加层

model.add(layers.Conv2D(32, kernel\_size=(5,5), # 添加卷积层，深度32，过滤器大小5\*5

activation='relu', # 使用relu激活函数

input\_shape=(img\_rows, img\_cols, 1))) # 输入的尺寸就是一张图片的尺寸(28,28,1)

model.add(layers.MaxPooling2D(pool\_size=(2, 2))) # 添加池化层，过滤器大小是2\*2

model.add(layers.Conv2D(64, (5,5), activation='relu')) # 添加卷积层，简单写法

model.add(layers.MaxPooling2D(pool\_size=(2, 2))) # 添加池化层

model.add(layers.Flatten()) # 将池化层的输出拉直，然后作为全连接层的输入

model.add(layers.Dense(500, activation='relu')) # 添加有500个结点的全连接层，激活函数用relu

model.add(layers.Dense(10, activation='softmax')) # 输出最终结果，有10个，激活函数用softmax

# 定义损失函数、优化函数、评测方法

model.compile(loss=keras.losses.categorical\_crossentropy,

optimizer=keras.optimizers.SGD(),

metrics=['accuracy'])

# 自动完成模型的训练过程

model.fit(X\_train, Y\_train, # 训练集

batch\_size=128, # batchsize

epochs=20, # 训练轮数

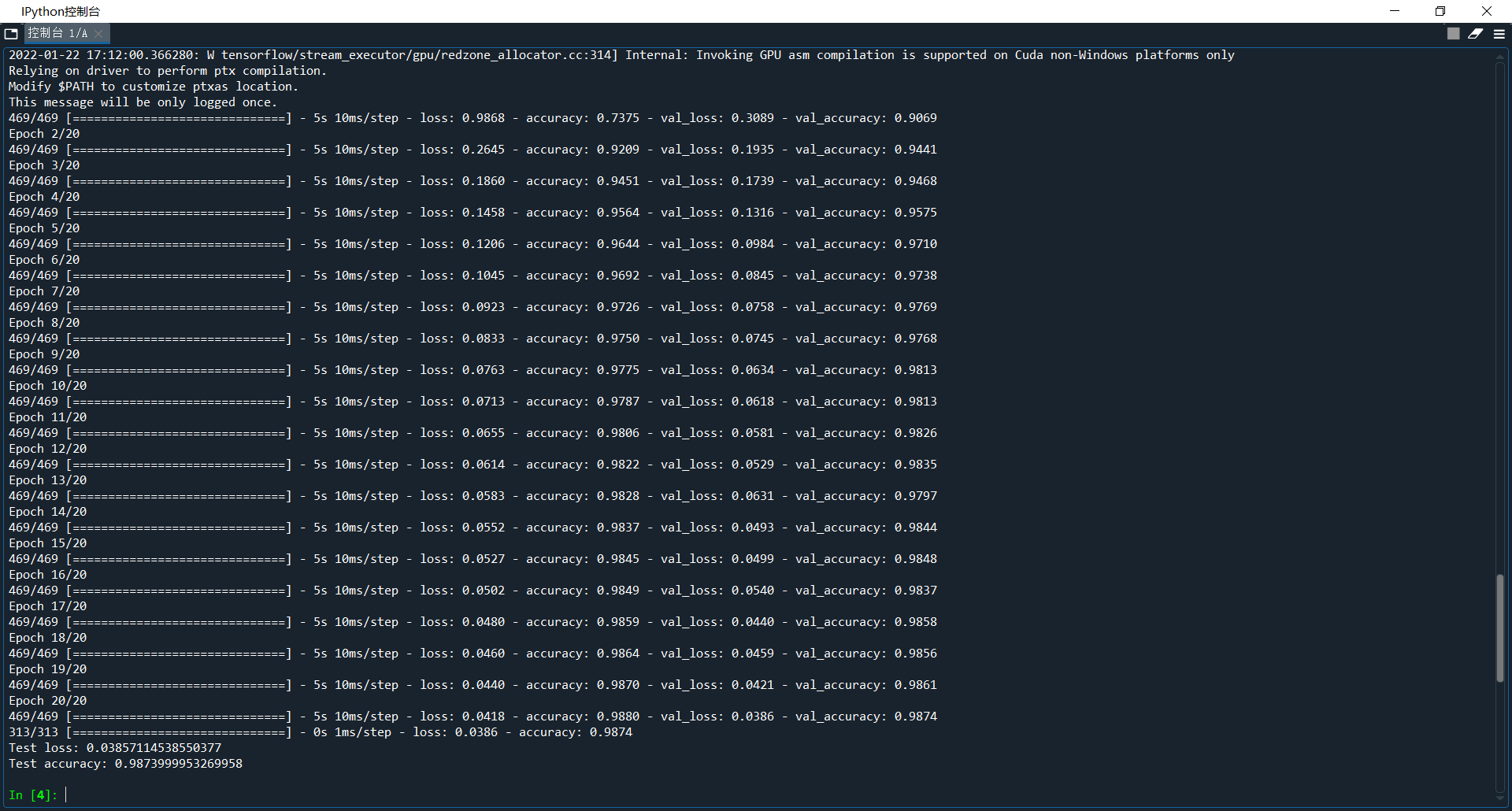
validation\_data=(X\_test, Y\_test)) # 验证集

# 打印运行结果，即损失和准确度

score = model.evaluate(X\_test, Y\_test)

print('Test loss:', score[0])

print('Test accuracy:', score[1])



2、再跑通[Rstudio上的实例](https://mp.weixin.qq.com/s/ZafgS9H0LDRZCD91SgDdfQ)。

# R包

# 在环境变量中设置 RETICULATE\_PYTHON=D:/Anaconda3，指向包含keras及tensorflow安装的Anaconda环境，这里是基本环境base

library(reticulate)

library(keras)

library(dplyr)

py\_discover\_config()

py\_config()

mnist <- dataset\_mnist(path = "C:/Users/Jean/.keras/datasets/mnist.npz")

# keras的数据文件默认放在C:\Users\lenovo\Documents\.keras\datasets目录下

#mnist <- dataset\_mnist()

str(mnist)

train\_images <- mnist$train$x

train\_label <- mnist$train$y

test\_images <- mnist$test$x

test\_label <- mnist$test$y

dim(train\_images)

dim(test\_images)

dim(train\_label)

dim(test\_label)

str(train\_images)

glimpse(train\_images)

str(test\_images)

# 设计神经网络结构,由2个完全连接层处理，一个有512个神经元，一个有10个神经元， 28\*28=784 -> 512 ->10

# 用线性堆栈层构建模型

network <- keras\_model\_sequential() %>%

layer\_dense(units = 512, activation = "relu", input\_shape = c(28 \* 28)) %>%

layer\_dense(units = 10, activation = "softmax") # 第二层没有输入格式参数，接受第一层的输出，512维向量。

# 设计算法的损失函数，优化算法和度量准则

network %>% compile(

optimizer = "rmsprop", # 优化器，选择的梯度下降算法

loss = "categorical\_crossentropy", # 损失函数

metrics = c("accuracy") # 观察指标

)

# 数据预处理， 每张图由二维数组变为1维数组，输入是1维数组（1维张量），标准化至[0,1]，完全连接层要求二维张量

train\_images <- array\_reshape(train\_images, c(60000, 28 \* 28))

train\_images <- train\_images / 255

test\_images <- array\_reshape(test\_images, c(10000, 28 \* 28))

test\_images <- test\_images / 255

# 目标变量因子化，变为类变量

train\_labels <- to\_categorical(train\_label)

test\_labels <- to\_categorical(test\_label)

dim(train\_images)

dim(test\_images)

dim(train\_labels)

dim(test\_labels)

# 训练数据集上拟合模型

network %>% fit(train\_images, train\_labels, epochs = 5, batch\_size = 128)

# 60000/128 =469，GPU版并行处理，每批128个样本，共469批，书中 60000/60000是CPU版串行训练，每次迭代要9~10s

# Epoch 1/5

# 469/469 [==============================] - 1s 3ms/step - loss: 0.2589 - accuracy: 0.9254

# Epoch 2/5

# 469/469 [==============================] - 1s 2ms/step - loss: 0.1033 - accuracy: 0.9696

# Epoch 3/5

# 469/469 [==============================] - 1s 2ms/step - loss: 0.0682 - accuracy: 0.9797

# Epoch 4/5

# 469/469 [==============================] - 1s 2ms/step - loss: 0.0493 - accuracy: 0.9852

# Epoch 5/5

# 469/469 [==============================] - 1s 2ms/step - loss: 0.0373 - accuracy: 0.9888

# 测试数据集测试模型

metrics <- network %>% evaluate(test\_images, test\_labels, batch\_size = 128)

metrics

# 10000/313=32，GPU并行处理同上。evaluate()如果没有给出 batch\_size参数，默认是32。

# 313/313 [==============================] - 0s 1ms/step - loss: 0.0728 - accuracy: 0.9783

# 测试样本的前10个样本的类别预测

network %>% predict\_classes(test\_images[1:10, ])

# [1] 7 2 1 0 4 1 4 9 5 9

# 画出这10个样本，看看识别的效果，因为test\_images经过reshape，要引用原来载入的数据。

# 第9个数字识别为5，有疑问。

opar <- par(no.readonly = TRUE)

par(mfrow=c(2,5),mai=c(0.1,0.1,0.3,0.1))

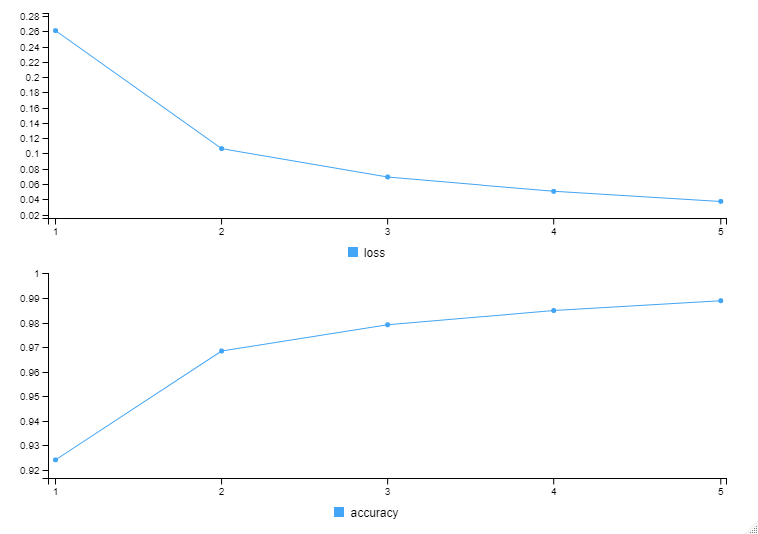
for(i in 1:10){

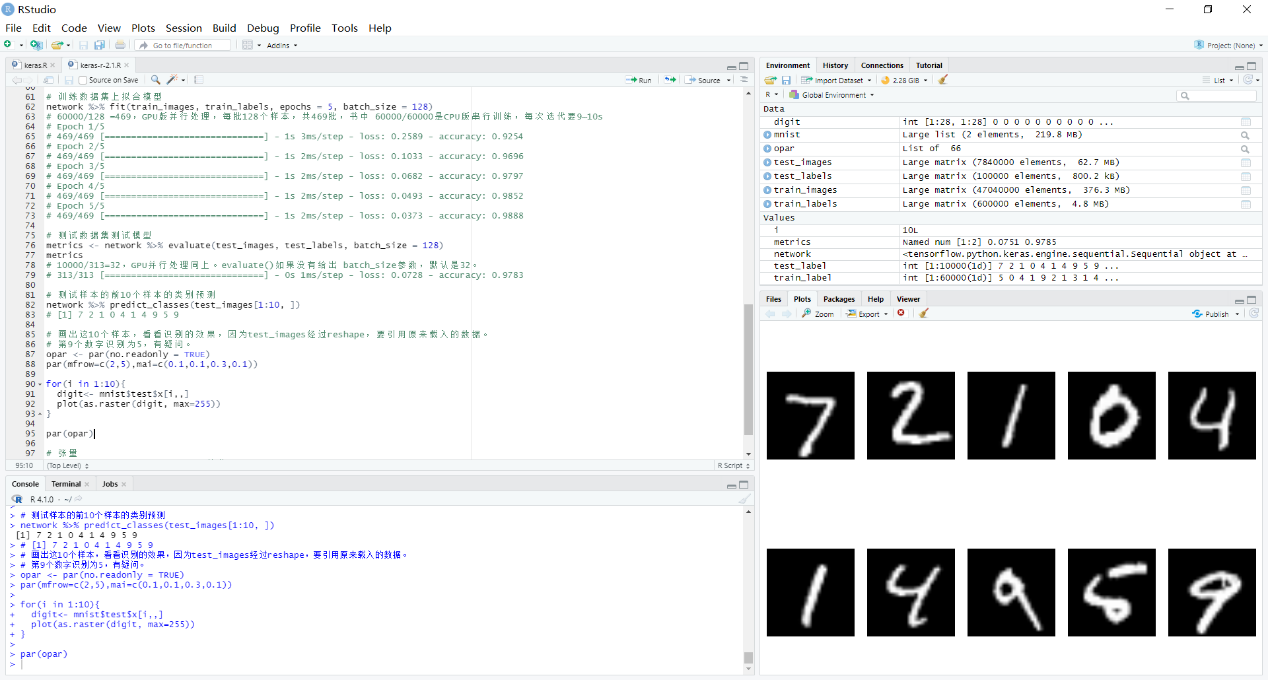
digit<- mnist$test$x[i,,]

plot(as.raster(digit, max=255))

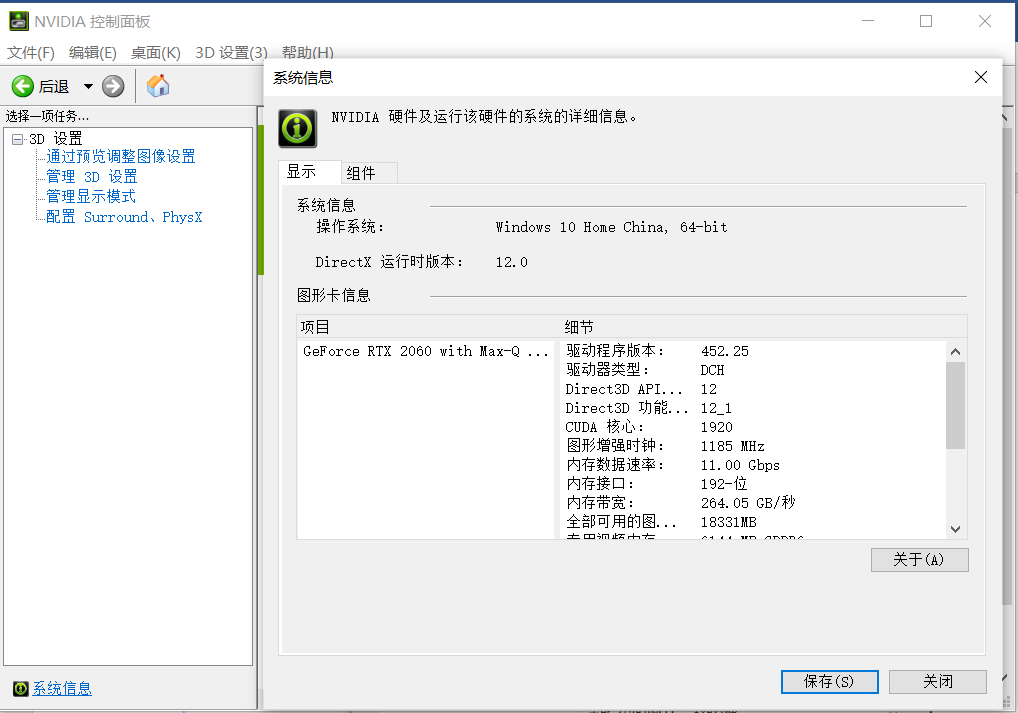
}

par(opar)





（六）、GPU型号参数



NVIDIA支持CUDA的GPU及其算力可以在[这里](https://developer.nvidia.com/zh-cn/cuda-gpus)查询。

（七）、产业概况

初步的了解，深度学习虽然要消耗海量数据，终端可能触及非常多的用户，但在平台端系统端开发端属于金字塔尖的小众应用。它的IP主要有3层：硬件、软件与容器。硬件就是GPU、FPGA等专用AI芯片，软件就是Tensorflow等软件框架，容器就是各大厂在云端提供的各种云服务。目前国内BATH等大厂主要在云平台上提供集成性质的容器服务，并提供相应的应用开发工具，框架主要是Tensorflow、PyTorch、MXNet、Caffe等开源框架，百度开发了自有知识产权的开源框架飞桨，华为、旷视等也自研深度学习框架。各大厂底层的GPU未作深入了解，目前市场上好像是NVIDIA与AMD、Intel等占据了主要份额（参考资料：[2020年GPU行业发展前景](https://www.chyxx.com/industry/202004/848005.html)）。

1、各大厂初步了解

1）阿里云。容器：DLC，开发工具PAI Studio等。支持Tensorflow、MXNet、Caffe、PyTorch。

2）华为云。容器：DLS，开发工具ModelArts等。支持Tensorflow、MXNet、Caffe、PyTorch、CNTK、Theano。

3）腾讯云。容器DI-X，开发工具TI-ONE等。支持Tensorflow、Caffe、PyTorch。

4）百度云。[飞桨开源深度学习平台](https://www.paddlepaddle.org.cn/)，自有知识产权。

2、主要深度学习框架初步了解

1）Tensorflow， Google开源。

2）MXNet, Apache软件基金会开源，Amazon AWS支持 。

3）PyTorch，Facebook开源，Torch纽约大学开源。

4）Caffe，加州大学伯克利分校开源。

5）CNTK，微软开源。

6）Theano，蒙特利尔大学开源。

3、参考资料：

1）[深度学习框架-TensorFlow, Catte, Theano, Torch等](https://blog.csdn.net/Alice__Time/article/details/109253280)

2）[11种深度学习框架影响力对比](http://m.elecfans.com/article/798648.html)

3）[从TensorFlow到Theano：横向对比七大深度学习框架](https://zhuanlan.zhihu.com/p/25285133)

4）这篇文章提供了更深入的介绍，[主流深度学习框架对比](https://blog.csdn.net/weixin_43061212/article/details/82718511?utm_term=%E4%B8%BB%E6%B5%81%E6%B7%B1%E5%BA%A6%E5%AD%A6%E4%B9%A0%E6%A1%86%E6%9E%B6%E5%AF%B9%E6%AF%94&amp;utm_medium=distribute.pc_aggpage_search_result.none-task-blog-2~all~sobaiduweb~default-0-82718511&amp;spm=3001.4430)。

5）这篇文章略微探讨了[国内深度学习框架自研之路](https://ai.51cto.com/art/202008/624938.htm)，介绍了国产[华为Mindspore](https://www.mindspore.cn/)、[旷视MegEngine](https://megengine.org.cn/)、[腾讯TNN](https://tnn.tencent.com/)、[清华Jittor](http://cg.cs.tsinghua.edu.cn/jittor/about/)等数个国产的开源框架。

从这些资料来看，大多数框架支持Python，小白从Keras开始是比较合适的。Rstudio开发了Keras与Tensorflow接口包，让R语言除了MXNet外，还可以使用Keras与Tensorflow，是非常好的扩展，其它框架以后也许陆续会有R语言的接口包出现（注：最近Rstudio发布了[R语言原生的Torch接口包](https://blogs.rstudio.com/ai/posts/2020-09-29-introducing-torch-for-r/)，以后有时间再了解一下）。

单机与云端的主要区别就是不需要容器，所以，作为小白个人学习来说，本文选择的硬件、框架与教材都是比较合适的。

一个深度学习框架或平台，不管技术上多么先进，只有解决了面临的实际问题，才是可望与可及的，比如，发票货物劳务名称识别就可以成为税务系统中一个成功的NLP落地应用，解决税收大数据应用中的瓶颈问题。我也期待其它有价值的落地应用不断涌现。

六、Shiny Server安装配置

我的Shiny Server安装在腾讯云的虚拟主机上，因为有学习研究的需要，春节后腾讯云有非常优惠的推广活动，4CPU/8G RAM/100G SSD，性能相当于过去的小型机，不到300块钱就可以用一年，然后也要避免陷入本系统两大云服务商的内卷之中，所以选择了第三方。

因为一直有用Redhat Linux，内网虚拟主机上装的是Redhat6.7,所以安装的是与Redhat兼容的CentOS7.6，这个版本比较老，后面编译安装R语言的软件包碰到了不少版本问题。CentOS8.0以上已经停止更新了，不过服务器端的软件更新比较慢，再用几年应该没有问题。所以如果重新装一次，应该用最新的版本。

（一）、安装CentOS的注意事项

1、安装最新版本的CentOS。

2、安装为devel开发模式，主要是会安装各种软件包的devel头文件包，R的软件包大部分是C语言与fortran写的，编译时需要使用这些软件包的头文件,装OS时一次一起装了就不用以后不断的编译错误一个一个的装。

3、Telnet与FTP

我使用SecureCRT在标准的SSH端口22上远程登录以及在该端口用SecureFX通过FTP上传文件，可以不安装FTP服务器，Rstudio IDE里也可以上传文件。有需要的话可以安装vsftp等FTP服务器。

4、更新gcc编译器到最新的版本,不少R软件包都要求较高版本的gcc编译器。CentOS7.6默认gcc版本为4.8，这里以升级至8.3.1版本为例，分别执行下面三条命令即可，无需手动下载源码编译。

参阅：<https://www.cnblogs.com/jixiaohua/p/11732225.html>

Gcc最新版本见： <https://github.com/gcc-mirror/gcc/tags> 。

1）安装centos-release-scl：

# yum install centos-release-scl

2） 安装devtoolset，注意，如果想安装7.\*版本的，就改成devtoolset-7-gcc\*，以此类推。

# yum install devtoolset-8-gcc\*

3） 激活对应的devtoolset，所以可以一次安装多个版本的devtoolset，需要的时候用下面这条命令切换到对应的版本

# scl enable devtoolset-8 bash

这条激活命令只对本次会话有效，重启会话后还是会变回原来的4.8.5版本。

4） 查看gcc版本

# gcc -v

显示为 gcc version 8.3.1 20190311 (Red Hat 8.3.1-3) (GCC)

5） 随意切换可按如下操作，重启后一直生效。

安装的devtoolset是在/opt/rh 目录下的，每个版本的目录下面都有个 enable 文件，如果需要永久启用某个版本，只需要执行

# source /opt/rh/devtoolset-8/enable

5、配置CentOS7中文支持，R语言的作图函数需要中文字体等支持。

参阅[参考资料1](https://blog.csdn.net/wlwlwlwl015/article/details/51482065)。

1)配置系统语言为中文。

# yum install -y system-config-language

# system-config-language

选择中文简体，确定后，会自动将系统语言设置成zh\_CN.UTF-8，重启系统,输入locale或env命令可以看到系统语言已经变成了zh\_CN.utf-8。

2)安装中文字体。

安装字体库配置工具：

# yum -y install fontconfig

查看是否有中文字体：

# fc-list :lang=zh

如果命令执行没有输出，则表示没有中文字体。

创建中文字体目录：

# mkdir /usr/share/fonts/chinese

在 windows 的 C:\Windows\Fonts目录下找到需要的字体， copy 到 /usr/share/fonts/chinese目录下，这里添加雅黑和新宋体：msyh.ttc,simsun.ttc

紧接着需要修改chinese目录的权限：

# chmod -R 755 /usr/share/fonts/Chinese

查看中文字体是否生效：

# fc-list :lang=zh

接下来需要安装ttmkfdir来搜索目录中所有的字体信息，并汇总生成fonts.scale文件，输入命令：

# yum -y install ttmkfdir

然后执行ttmkfdir命令即可：

# ttmkfdir -e /usr/share/X11/fonts/encodings/encodings.dir

最后一步就是修改字体配置文件了，首先通过编辑器打开配置文件：

# vi /etc/fonts/fonts.conf

可以看到一个Font list，即字体列表，在这里需要把我们添加的中文字体位置加进去：

<!-- Font directory list -->

<dir>/usr/share/fonts</dir>

<dir>/usr/share/X11/fonts/Type1</dir>

<dir>/usr/share/X11/fonts/TTF</dir>

<dir>/usr/local/share/fonts</dir>

<dir>/usr/share/fonts/chinese</dir>

<dir prefix="xdg">fonts</dir>

<!-- the following element will be removed in the future -->

<dir>~/.fonts</dir>

然后输入:wq保存退出，最后别忘了刷新内存中的字体缓存，这样就不用reboot重启了：

# fc-cache

这样所有的步骤就算完成了，最后再次通过fc-list看一下字体列表。

3)然后设定登录语言

# vi /etc/profile

# Added by Jean for Chinese support of R

export LANG=zh\_CN.UTF-8

export LC\_ALL=zh\_CN.UTF-8

6、配置防火墙。

腾讯云采用集中式的云防火墙，默认CentOS上的防火墙没有启动，后面安装了各种服务，要在云防火墙上打开相应的端口，如果是其它云平台，要看看是否需要启动虚拟机的防火墙。需要打开以下的TCP端口：

22：SSH 登录，默认；8787：Rstudio Linux Server

3838: Shiny Server; 8080: Tomcat

8000: Jupyterhub

腾讯云防火墙默认还打开了80(HTTP)，443(SSL)，3389(windows登录)端口，将来也许会使用。

可以安装文本浏览器links在终端命令行里测试上述各服务端口在防火墙内本机上是否正常启动，然后再在公网上测试。

# yum install links

# links http://127.0.0.1:8787

（二）、从源码编译安装R

1、安装依赖包

要安装libcurl-devel.x86\_64包，参阅：

<https://stackoverflow.com/questions/38690232/installing-r-on-linux-configure-error-libcurl-7-28-0-library-and-headers-a>

安装配置必要的依赖库：

# yum install -y --skip-broken zlib java gcc-gfortran gcc gcc-c++ readline-devel libXt-devel bzi zlib-devel java-1.8.0-openjdk-\* libcurl-devel.x86\_64

安装配置必要图片库：

# yum install -y libpng-devel libjpeg-devel libtiff-devel libcairo-devel cairo-devel

2、从源码编译安装

编译最重要一步就是检查配置，运行configure，会提示缺少的依赖包，只需要将相应的依赖包安装成功即可。R最新的版本是4.1.2。

下载：

# curl -O https://mirrors.tuna.tsinghua.edu.cn/CRAN/src/base/R-4/R-4.1.2.tar.gz

# tar -zxvf R-4.1.2.tar.gz

# cd R-4.1.2

检测配置R，用pcre1和tcltk选项，后面Radiant包有需要tcltk GUI支持的包，虽然运行在服务器上并不需要GUI界面，只通过终端界面管理。

# ./configure --prefix=/usr/lib64/R-4.1.2 --enable-R-shlib --with-pcre1 --with-tcltk

编译：

# make -j4

-jN (N为可变数字，可以加速编译，但是如果Makefile逻辑写的不好的话，容易报错，建议串行编译)

安装：

# make install

3、R可执行文件添加到PATH

#vi /etc/profile

export PATH=$PATH:/usr/lib64/R-4.1.2/bin

（三）、安装Rstudio-server

1、下载

# wget https://download2.rstudio.org/server/centos7/x86\_64/rstudio-server-rhel-2021.09.2-382-x86\_64.rpm

2、安装

# yum install rstudio-server-rhel-2021.09.2-382-x86\_64.rpm

3、配置

1）服务器配置

#vi /etc/rstudio/rserver.conf

# Server Configuration File

www-port=8787

rsession-which-r=/usr/lib64/R-4.1.2/bin/R

auth-required-user-group=rusers

2）session配置

#vi /etc/rstudio/rsession.conf

# R Session Configuration File

session-timeout-minutes=0

3）[为作图配置Rstudio(Browser Client)全局选项](https://github.com/rstudio/rstudio/issues/2214)，否则作图会出错。

Tools menu is the Global Options... command that lets you set the graphics backend (under the General section, Graphics tab). Changing the graphics device backend to "Cairo"

或者配置服务器的全局选项，参阅：[参考资料](https://github.com/rstudio/rstudio/issues/2142)。

# vi /etc/rstudio/rstudio-prefs.json

#"enum": ["default", "cairo", "cairo-png", "quartz", "windows", "ragg"],

#options(RStudioGD.backend = "ragg")

#set it in (say) your R installation's Rprofile.site

{

graphics\_backend: "cairo"

}

Rstudio Server用户的配置和管理可以参阅前面《附录一 软件和数据》之《四、Rstudio Linux Server 安装配置》。

#systemctl enable rstudio-server.service

#systemctl start rstudio-server.service

#systemctl stop rstudio-server.service

#systemctl status rstudio-server

（四）、安装Shiny包

在SecureCRT终端启动R命令行从源码安装，会自动下载依赖的包并编译安装。Linux上R的软件包基本上都要从源码编译安装。

# R

>install.packages(“shiny”)

（五）、安装常用的软件包

[离线安装 stringi](https://github.com/gagolews/stringi/blob/master/INSTALL)：

1. [下载源码](https://cran.r-project.org/src/contrib/stringi_1.7.6.tar.gz)
2. [下载数据](https://github.com/unicode-org/icu/releases/download/release-69-1/icu4c-69_1-data-bin-l.zip)
3. 上传到服务器，[离线安装](https://github.com/gagolews/stringi/blob/master/INSTALL)，其中/root/R是ICUDT数据所在目录。

install.packages("/root/R/stringi\_1.7.6.tar.gz", configure.vars="ICUDT\_DIR=/root/R/")

安装 Cairo包等， Shiny要用到，用install.packages(“包名”)安装。cairo, devtools, ggplot2, sqldf, dplyr, openxlsx, Rserve, igraph, reticulate等。

（六）、安装shiny server

[参考资料](https://www.rstudio.com/products/shiny/download-server/redhat-centos/)。

下载：

# wget https://download3.rstudio.org/centos7/x86\_64/shiny-server-1.5.17.973-x86\_64.rpm

安装：

# yum install --nogpgcheck shiny-server-1.5.17.973-x86\_64.rpm

This will install Shiny Server into /opt/shiny-server/, with the main executable in /opt/shiny-server/bin/shiny-server, and also create a new shiny user.

[管理：](https://docs.rstudio.com/shiny-server/)

启动停止：

#systemctl start shiny-server

#systemctl stop shiny-server

#systemctl restart shiny-server

#systemctl status shiny-server

作为系统服务运行：

#systemctl enable shiny-server

#systemctl disable shiny-server

访问：

<http://124.223.110.20:3838/sample-apps/hello/>.

（七）、发布APP到Shiny Server

1. [配置Shiny Server为用户发布模式](https://docs.rstudio.com/shiny-server/#host-per-user-application-directories)，允许用户自行发布App，这样用户只要把APP拷贝到Ta的home目录下的~/ShinyApps目录下即可，比如jean把/home/jean/ShinyDev/weibo拷贝成/home/jean/ShinyApps/weibo就发布了APP。具体可参阅[Shiny Server Administrator’s Guide](https://docs.rstudio.com/shiny-server/)。

配置：

# vi /etc/shiny-server/shiny-server.conf

# Define a top-level server which will listen on a port

server {

# Instruct this server to listen on port 3838

listen 3838;

# Define the location available at the base URL

location /{

# Define the user we should use when spawning R Shiny processes

run\_as shiny;

# Run this location in 'site\_dir' mode, which hosts the entire directory

# tree at '/srv/shiny-server'

site\_dir /srv/shiny-server;

# Define where we should put the log files for this location

log\_dir /var/log/shiny-server;

# Should we list the contents of a (non-Shiny-App) directory when the user

# visits the corresponding URL?

directory\_index on;

}

# Should make a directory named ShinyApps at user home directory

location /users {

run\_as :HOME\_USER:;

user\_dirs;

}

}

1. 在用户home目录下建立ShinyApps目录，测试好的App拷贝到该目录下，每个App一个子目录。
2. [单文件发布](http://rstudio-pubs-static.s3.amazonaws.com/28352_f5bdc2b1f93d4f9993de55a049aa7e1a.html)，在子目录下命名为app.R，DESCRIPTION文件可选。
3. 双文件发布，在子目录下命名为ui.R与server.R，global.R可选。
4. 通过 <http://ip:port/users/UserName/AppDirectory>访问App。比如：

http://124.223.110.20:3838/jean/weibo/

（八）、连接阿里云

阿里云支持R语言连接，先连接阿里云，前面编译R时已经安装了JDK，如果有需要，要先配置好JAVA\_HOME等环境变量。

1. 上传[odpscmd](https://github.com/aliyun/aliyun-odps-console)客户端，它是个java应用程序，需要JDK运行。

配置好 /usr/local/odpscmd/conf/odps\_config.ini

1. 安装[RODPS](https://github.com/aliyun/aliyun-odps-r-plugin)

install.packages('https://github.com/aliyun/aliyun-odps-r-plugin/releases/download/v2.0.3/RODPS\_2.0.3.tar.gz', type="source", repos=NULL)

1. 设置环境变量

# vi /etc/profile

# Added by Jean 2022/03/02 for Java & Tomcat

JAVA\_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.322.b06-1.el7\_9.x86\_64

JRE\_HOME=$JAVA\_HOME/jre

CLASS\_PATH=.:$JAVA\_HOME/lib/dt.jar:$JAVA\_HOME/lib/tools.jar:$JRE\_HOME/lib

PATH=$PATH:$JAVA\_HOME/bin:$JRE\_HOME/bin

CATALINA\_BASE=/usr/share/tomcat

CATALINA\_HOME=/usr/share/tomcat

export JAVA\_HOME PATH CLASSPATH CATALINA\_BASE CATALINA\_HOME

# Added by Jean for rJava

export LD\_LIBRARY\_PATH=$JAVA\_HOME/jre/lib/amd64/server:$LD\_LIBRARY\_PATH

# Added by Jean for ODPS

export RODPS\_CONFIG=/usr/local/odpscmd/conf/odps\_config.ini

（九）、安装Shiny机器学习教学App Radiant

Radiant会自动安装很多依赖或可能引用的包，其中一些包会在编译时有问题，需要先单独安装，否则不能继续。另外有大约40个包因缺少相关包或头文件等原因安装不成功，需要事后一个个用yum install补装相关包的头文件，然后一个个安装。最后剩下7个依赖tck/tk GUI的包，不装也没有影响。

1. 安装nlopt，R包nloptr需要它。

在线安装：

# yum install nlopt nlopt-devel

离线安装：

下载：

NLopt-2.4.2-2.el7.x86\_64.rpm- <https://pkgs.org/centos-7/epel-x86_64/NLopt-2.4.2-2.el7.x86_64.rpm.html>

NLopt-devel-2.4.2-2.el7.x86\_64.rpm - <https://pkgs.org/centos-7/epel-x86_64/NLopt-devel-2.4.2-2.el7.x86_64.rpm.htm>

安装：

# rpm -Uvh NLopt-2.4.2-2.el7.x86\_64.rpm

# rpm -Uvh NLopt-devel-2.4.2-2.el7.x86\_64.rpm

建立link，编译nloptr包时需要找到libnlopt.so:

参阅： <https://blog.csdn.net/a922922737/article/details/109697291>

#link /usr/lib64/libnlopt\_cxx.so.0.8.2 /usr/lib64/libnlopt.so

安装nloptr:

# R

>Install.packages(“nloptr”)

1. 升级gcc，以便编译xgboost，需要C++14，前面已讲。
2. 安装R包xgboost。

参阅： <https://blog.csdn.net/Damonyugg/article/details/112481346>

在终端root的home目录/root下运行以下命令

# [mkdir](https://so.csdn.net/so/search?q=mkdir&spm=1001.2101.3001.7020) ~/.R

# vim ~/.R/Makevars

CXX14FLAGS=-O3 -march=[native](https://so.csdn.net/so/search?q=native&spm=1001.2101.3001.7020) -mtune=native -fPIC

CXX14=g++

PATH和LD\_LIBRARY\_PATH的设置上面的切换gcc版本命令已经设置好了，直接进入R命令行安装。

# R

>Install.packages(“xgboost”)

1. 安装radiant

参阅： <https://radiant-rstats.github.io/docs/install.html> 。

# R

>Install.packages(“radiant”)

1. 在Rstudio中启动

>radiant::radiant()

1. 安装更新包并更新，注意更新可能需要比较长的时间，很多包会更新。

>install.packages("remotes")

>remotes::install\_github("radiant-rstats/radiant.update", upgrade = "never")

>radiant.update::radiant.update()

1. 在shiny server上运行radiant

参考：<https://github.com/radiant-rstats/radiant>

1. Update log，有些包未有安装，这些是GUI相关的包，需要GUI环境tcl/tk的支持，但R-4.1.2编译时没有用 –with-tcltk选项打开，因为是服务器端运行Rstudio Linux Server，是web界面，没有GUI界面，用 –with-pcre1编译的选项。不影响使用，可以忽略。

Warning messages:

1: In install.packages(c("gitgadget", "fda", "fds", "gganimate", "hdrcde", :

installation of package ‘misc3d’ had non-zero exit status

2: In install.packages(c("gitgadget", "fda", "fds", "gganimate", "hdrcde", :

installation of package ‘plot3D’ had non-zero exit status

3: In install.packages(c("gitgadget", "fda", "fds", "gganimate", "hdrcde", :

installation of package ‘ks’ had non-zero exit status

4: In install.packages(c("gitgadget", "fda", "fds", "gganimate", "hdrcde", :

installation of package ‘hdrcde’ had non-zero exit status

5: In install.packages(c("gitgadget", "fda", "fds", "gganimate", "hdrcde", :

installation of package ‘rainbow’ had non-zero exit status

6: In install.packages(c("gitgadget", "fda", "fds", "gganimate", "hdrcde", :

installation of package ‘fds’ had non-zero exit status

7: In install.packages(c("gitgadget", "fda", "fds", "gganimate", "hdrcde", :

installation of package ‘fda’ had non-zero exit status

1. Install GDAL 2.4.2 on CentOS 7

<https://gist.github.com/alanorth/9681766ed4c737adfb48a4ef549a8503>

1. 从github安装sf包

<https://stackoverflow.com/questions/71163200/sf-install-on-ubuntu-fails-no-member-named-setaxismappingstrategy>

（十)、安装anaconda3

参阅：

<https://blog.csdn.net/lwgkzl/article/details/89329383>

<https://linux265.com/news/3530.html>

选择执行 conda init会修改/root/.bashrc，加入激活conda环境的脚本，它在/etc/profile之后执行，会改变PATH的搜索顺序，后面升级node.js安装Jupyterhub时会有影响，需要修改/root/.bashrc更正PATH。

done

installation finished.

Do you wish the installer to initialize Anaconda3

by running conda init? [yes|no]

[no] >>> yes

no change /usr/lib64/anaconda3/condabin/conda

no change /usr/lib64/anaconda3/bin/conda

no change /usr/lib64/anaconda3/bin/conda-env

no change /usr/lib64/anaconda3/bin/activate

no change /usr/lib64/anaconda3/bin/deactivate

no change /usr/lib64/anaconda3/etc/profile.d/conda.sh

no change /usr/lib64/anaconda3/etc/fish/conf.d/conda.fish

no change /usr/lib64/anaconda3/shell/condabin/Conda.psm1

no change /usr/lib64/anaconda3/shell/condabin/conda-hook.ps1

no change /usr/lib64/anaconda3/lib/python3.9/site-packages/xontrib/conda.xsh

no change /usr/lib64/anaconda3/etc/profile.d/conda.csh

modified /root/.bashrc

==> For changes to take effect, close and re-open your current shell. <==

If you'd prefer that conda's base environment not be activated on startup,

set the auto\_activate\_base parameter to false:

conda config --set auto\_activate\_base false

Thank you for installing Anaconda3!

（十一）、用Rstudio运行Python

Rstudio通过reticulate包连接Python。参阅：

<https://community.rstudio.com/t/start-rstudio-server-session-in-conda-environment/12516>

<https://rstudio.github.io/reticulate/articles/versions.html>

1、从GitHub安装reticulate，否则有bug，library(reticulate)装入时会报错。也可以下载最新源码包离线安装，参阅：

<https://github.com/rstudio/reticulate/issues/1155>

2、设置RETICULATE\_PYTHON环境变量，所有R进程都使用Anaconda3的Python。

# vi /usr/lib64/R-4.1.2/lib64/R/etc/Renviron

# Added for python by Jean 2022/02/27

RETICULATE\_PYTHON='/usr/lib64/anaconda3/bin/python'

3、测试import语句，比如matplotlib，会报错，因为CentOS7的一些动态库版本太低，在终端命令行状态下运行python import就没有问题。所以这些是动态库加载时搜索路径的问题，更新动态库的连接或搜索路径顺序。参阅：

<https://stackoverflow.com/questions/49875588/importerror-lib64-libstdc-so-6-version-cxxabi-1-3-9-not-found>

<https://www.cnblogs.com/jimlau/p/13263052.html>

<https://blog.csdn.net/zhuiqiuzhuoyue583/article/details/80150207>

<https://docs.oracle.com/cd/E19205-01/820-1204/aeucy/index.html>

<https://docs.oracle.com/cd/E19205-01/820-1204/aeucy/index.html>

<https://blog.51cto.com/lsscto/904078>

# vi /etc/profile

# Added for matplotlib

export LD\_LIBRARY\_PATH=/usr/lib64/anaconda3/lib:$LD\_LIBRARY\_PATH

4、配置Conda与pip国内镜像

参阅：

<https://blog.csdn.net/weixin_44064649/article/details/102868067>

<https://blog.csdn.net/jiankunking/article/details/86627207>

5、安装阿里云ODPS python 接口包并在Rstudio中运行测试程序连接。

# pip install odps

6、安装华为云Data Lake Insight python接口包并在Rstudio中运行程序测试连接。

OBS python 驱动：

# pip install esdk-obs-python

DLI python 驱动：

<https://uquery-sdk.obs-website.cn-north-1.myhwclouds.com/>

# unzip dli-sdk-python-1.0.8.zip

# cd dli-sdk-python-1.0.8

# python setup.py install

7、Obs client在Rstudio中由reticulate调用时可能会报错：

UnicodeEncodeError: ‘latin-1‘ codec can't encode characters

这是'zh\_CN.UTF-8' 的 local 令datetime.strftime(GMT\_DATE\_FORMAT)输出汉字，obsClient验证会出错，在程序中明确设定en\_US.UTF-8解决。

# import locale

# locale.setlocale(locale.LC\_ALL, 'en\_US.UTF-8')

8、为Python设置环境变量

#vi /etc/profile

# Added by Jean for Python

export PYTHONIOENCODING=utf-8

export RETICULATE\_PYTHON='/usr/lib64/anaconda3/bin/python'

export PATH=/usr/lib64/anaconda3/bin:$PATH

# vi /usr/lib64/R-4.1.2/lib64/R/etc/Renviron

# Added for python by Jean 2022/02/27

RETICULATE\_PYTHON='/usr/lib64/anaconda3/bin/python'

（十二)、安装Tomcat

安装参阅：

<https://www.cnblogs.com/nicknailo/p/8571004.html>

yum运行报错：/usr/lib64/python2.7/site-packages/pycurl.so: undefined symbol: CRYPTO\_num\_locks，参阅：

<https://www.cnblogs.com/relax-zw/p/11328504.html>

是因为yum动态库使用了新安装libcurl库导致的。使用ldd查看动态库依赖关系：

# ldd /usr/lib64/python2.7/site-packages/pycurl.so

连接到正确的版本即可。

可能会报错：/lib/liblzma.so.5: version `XZ\_5.1.2alpha‘ not found (required by /lib64/librpmio.so.3)，是因为anaconda3安装了更高版本的liblzma.so.5.2.5，安装Tomcat用的xz工具需要它（XZ\_5.2），而yum需要自带低版本支持XZ\_5.1.2alpha的liblzma.so.5.2.2，把搜索路径上的相应版本拷贝连接到liblzma.so.5.2.2即可。

查看某动态库支持的软件版本：

# strings -a liblzma.so.5.2.2|grep XZ\_5

找到所有版本的同一个文件：

# Type -a xz

更新动态库路径时用 ldconfig -X，以排除生成符号连接连接到不正确的版本。

输出环境变量：

# Added by Jean 2022/03/02 for Tomcat

JAVA\_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.322.b06-1.el7\_9.x86\_64

JRE\_HOME=$JAVA\_HOME/jre

CLASS\_PATH=.:$JAVA\_HOME/lib/dt.jar:$JAVA\_HOME/lib/tools.jar:$JRE\_HOME/lib

PATH=$PATH:$JAVA\_HOME/bin:$JRE\_HOME/bin

CATALINA\_BASE=/usr/share/tomcat

CATALINA\_HOME=/usr/share/tomcat

export JAVA\_HOME PATH CLASSPATH CATALINA\_BASE CATALINA\_HOME

配置管理账户，参阅：<http://jingyan.baidu.com/album/358570f66494a5ce4724fce9.html>

# cd /usr/share/tomcat/conf

# vi tomcat-users.xml

在<tomcat-users></tomcat-users>之间添加下面语句

<role rolename="admin-gui"/>

<role rolename="manager-gui"/>

<user username="tomcat" password="tomcat" roles="admin-gui"/>

<user username="admin" password="1234" roles="manager-gui"/>

其中用户名为tomcat，密码为tomcat的是用来登录tomcat的Host Manager的，而用户名为admin，密码为1234是用来登录tomcat的App Manager的。

管理：

#/sbin/tomcat {start|start-security|stop|version}

#cd /usr/sbin

# ./tomcat start&

#./tomcat stop&

处理Apache Tomcat 文件包含漏洞（CVE-2020-1938），参阅：

<https://www.cnblogs.com/miracle-luna/p/12359264.html>

直接关闭AJP Connector：

# cd /usr/share/tomcat/conf

# vi server.xml

<!-- <Connector port="8009" protocol="AJP/1.3" redirectPort="8443" /> -->

工具 xz依赖于最终版liblzma.so.5.2.2，如果要运行，则切换到该版本。

#cd /usr/local/lib

# cp liblzma.so.5.2.2 liblzma.so.5.2.2.saved

# strings -a liblzma.so.5.2.2|grep XZ\_5

# cp liblzma.so.5.2.2.new liblzma.so.5.2.2

# strings -a liblzma.so.5.2.2|grep XZ\_5

用过之后，切换回带alpha的5.2.2版， yum工具用旧版。

# cp liblzma.so.5.2.2.saved liblzma.so.5.2.2

（十三）、安装Jupyterhub2.0.0

Jupyterhub提供了Linux上多用户使用Jupyterlab,在WEB界面上开发Python程序的途径，注意它使用的是Jupyter Notebook的.ipynb后缀而不是Spyder的.py后缀。Rstudio可以调试R程序，也可以运行Python程序，但它不能调试Python程序，所以还是需要有调试功能的Jupyterlab。Jupyterlab是单用户的，Jupyterhub则通过configurable-http-proxy提供了多用户对Jupyterlab的访问，它提供了类似Rstudio Linux Server的功能。

参考这两篇文章安装：

<https://github.com/jupyterhub/jupyterhub-the-hard-way/blob/HEAD/docs/installation-guide-hard.md>

<https://jupyterhub.readthedocs.io/en/stable/quickstart.html>

参考这两篇文章配置：<https://xujinzh.github.io/2021/02/20/jupyterhub/>

<https://xujinzh.github.io/2021/12/21/jupyterhub-2-multiuser/>

需要先升级 node.js与npm，参考这篇文章:

<https://segmentfault.com/a/1190000015302680>

1. 安装anaconda

见上面第（十）节。把anaconda3/bin加入用户的PATH环境变量。

1. CentOS7安装并升级node.js,npm

Jupyterhub需要较新版本的node.js及npm，按这篇文章的介绍安装并升级：

<https://segmentfault.com/a/1190000015302680>

先安装旧版的nodejs与npm，此时yum需要使用旧版的liblzma.so.5.2.2，参见上面第(十二)节末尾的说明替换:

# yum install nodejs npm

# npm version

npm 3.10.10 node 6.11.2

升级需要使用新版的liblzma.so.5.2.2，按上面第(十二)节末尾的说明替换，升级后再替换回去。

#cd /usr/local/lib

# cp liblzma.so.5.2.2.new liblzma.so.5.2.2

用过之后，切换回带alpha的5.2.2版， yum工具用旧版。

# cp liblzma.so.5.2.2.saved liblzma.so.5.2.2

1. 安装n

n是node.js管理工具，是TJ写的，Github: [https://github.com/tj/n](https://link.segmentfault.com/?enc=UKeUuNsQljlxPUD31sUPUg%3D%3D.nNrl98T0NksbKBRraiyxwMeVu1ySgI1CPeemfUh6oUw%3D)

# npm install -g n

1. 升级node.js及npm

# n latest

1. 切换 node

查看 node 当前安装路径:

#which node

最新版node安装在/usr/local/n/versions/node/17.7.1，因为anaconda3的init命令会更改/etc/profile之后再执行的.bashrc，设置路径指向/usr/lib64/anaconda3/bin，把低版本的node排在前面，所以还要把这两个变量在.bashrc的最后输出。

# vi /root/.bashrc

# vi /etc/profile

# for node.js, change to the current version, for jupyterhub & jypyterlab

export N\_PREFIX=/usr/local/n/versions/node/17.7.1

export PATH=$N\_PREFIX/bin:$PATH

# source /etc/profile

# node -v

v17.7.1

1. 安装configurable-http-proxy

# npm install -g configurable-http-proxy

# configurable-http-proxy -h

# conda list configurable-http-proxy

4.0.1 build node6\_0

1. 安装jupyterhub, jupyterlab, jupyter notebook

# conda install -c conda-forge jupyterhub

# conda install jupyterlab notebook

# jupyterhub -h

1. 生成配置文件

#jupyterhub --generate-config

#mkdir -p /etc/jupyterhub

#mv jupyterhub\_config.py /etc/jupyterhub/config.py

## 使得rusers组用户能够在该目录创建文件

#chown -R root:rusers /etc/jupyterhub/

#chmod 770 -R /etc/jupyterhub/

1. 设置管理员账户及 default url

# vi /etc/jupyterhub/config.py

# Added by Jean 2022/03/19

c.Spawner.default\_url = '/lab'

c.Authenticator.admin\_users = set(['jean'])

c.JupyterHub.admin\_access = True

c.JupyterHub.extra\_log\_file = '/var/log/jupyterhub.log'

1. 启动jupyterhub

先用默认配置启动测试，nohup是不要挂起后台服务进程。

# nohup jupyterhub &

# nohup jupyterhub >> /var/log/jupyterhub.log &

测试通过后可以用配置文件再测试一下，新手很容易写出不正确的配置，导致启动不了或访问不了。

# nohup jupyterhub -f /etc/jupyterhub/config.py &

1. 打开防火墙端口8000
2. 配置成系统服务。参阅：

<https://stackoverflow.com/questions/35439123/systemd-environment-directive-to-set-path>

<https://github.com/jupyterhub/jupyterhub-the-hard-way/blob/HEAD/docs/installation-guide-hard.md>

# vi /etc/systemd/system/jupyterhub.service

[Unit]

Description=Jupyterhub service

After=syslog.target network.target

[Service]

User=root

Environment="PATH=/usr/local/n/versions/node/17.7.1/bin:/usr/lib64/anaconda3/bin:/usr/lib64/anaconda3/condabin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/root/bin"

ExecStart=/usr/lib64/anaconda3/bin/jupyterhub -f /etc/jupyterhub/config.py

[Install]

WantedBy=multi-user.target

1. 管理服务

# root 用户使用，普通用户请使用 sudo

#systemctl status jupyterhub.service

#systemctl start jupyterhub.service

#systemctl stop jupyterhub.service

#systemctl enable jupyterhub.service

查看服务输出日志

#journalctl -u jupyterhub.service -f

也可以把上面配置的日志文件/var/log/jupyterhub.log下到客户端看。

（十四）、为Rstudio Linux Server与Jupyterhub等配置SSL连接

这两个服务默认都是没有加密的连接，用户验证等操作时用户名口令传送的都是明文，在公网上不够安全，可以用自签的数字证书配置这些服务器，为它们加上SSL连接。Rstudio与Jupyterlab里面都带有Linux终端功能，可以打开终端窗口远程管理服务器，比如su到root用户修改配置和安装新的软件包，这是http协议上的隧道连接，可以穿透一些防火墙的拦截，比较方便。

比如单位的外网WIFI就屏蔽了SSH端口、github和stackoverflow，我不知道为什么要屏蔽它们，只有搞IT的人需要它们，搞IT的人也离不开它们。我平常用这些网站时，要么用手机热点，要么翻墙，就不太方便。配个SSL用http隧道就比较方便了。

1、创建服务器自签证书

参考资料：[OpenSSL创建带SAN扩展的证书并进行CA自签](https://blog.csdn.net/dotalee/article/details/78041691)

[自建CA服务器&申请和颁发证书](http://blog.wangshuai.tech/%E8%87%AA%E5%BB%BACA%E6%9C%8D%E5%8A%A1%E5%99%A8&%E7%94%B3%E8%AF%B7%E5%92%8C%E9%A2%81%E5%8F%91%E8%AF%81%E4%B9%A6.html)

[How to generate a self-signed SSL certificate for an IP address](https://medium.com/@antelle/how-to-generate-a-self-signed-ssl-certificate-for-an-ip-address-f0dd8dddf754)

这里有两个要注意的地方：

1. Chrome浏览器需要自签证书指定SAN(Subject Alternative Name)扩展。
2. 自签证书的SAN绑定到IP，因为我的站点直接通过IP访问，还没有域名。

（1）、生成CA密钥，这里没有加密码保护，因为在根目录。

# cd /root

# mkdir cert

# cd cert

# openssl genrsa -out ca.key 2048

（2）、生成CA根证书

# openssl req -sha256 -new -x509 -days 3650 -key ca.key -out ca.crt \

-subj "/C=CN/ST=GD/L=ZhuHai/O=Jean/OU=Study/CN=RootCA"

（3）、创建自建CA所需要的文件

生成证书索引数据库文件

# touch /etc/pki/CA/index.txt

指定第一个颁发证书的序列号

# echo 01 > /etc/pki/CA/serial

指定第一个吊销证书的编号

# echo 01 > /etc/pki/CA/crlnumber

（4）、生成服务器密钥

# openssl genrsa -out server.key 2048

（5）、生成服务器证书请求文件

# openssl req -new \

-sha256 \

-key server.key \

-subj "/C=CN/ST=GD/L=ZhuHai/O=Jean/OU=Study/CN=124.223.110.20" \

-reqexts SAN \

-config <(cat /etc/pki/tls/openssl.cnf \

<(printf "[SAN]\nsubjectAltName=IP.1:124.223.110.20")) \

-out server.csr

（6）、CA签署服务器证书

# openssl ca -in server.csr \

-md sha256 \

-keyfile ca.key \

-cert ca.crt \

-extensions SAN \

-config <(cat /etc/pki/tls/openssl.cnf \

<(printf "[SAN]\nsubjectAltName=IP.1:124.223.110.20")) \

-out server.crt

（7）、吊销证书

# openssl ca -revoke /etc/pki/CA/newcerts/01.pem \

-keyfile ca.key -cert ca.crt -config /etc/pki/tls/openssl.cnf

（8）、更新证书吊销列表

# openssl ca -gencrl -out /etc/pki/CA/crl/crl.pem \

-keyfile ca.key -cert ca.crt -config /etc/pki/tls/openssl.cnf

2、配置Jupyterhub，参阅[Jupyterhub security settings](https://jupyterhub.readthedocs.io/en/stable/getting-started/security-basics.html)，重启服务。

# vi /etc/jupyterhub/config.py

# Added by Jean for SSL 2022/03/19

c.JupyterHub.ssl\_key = '/root/cert/server.key'

c.JupyterHub.ssl\_cert = '/root/cert/server.crt'

3、下载自签证书的CA根证书导入浏览器，存入“受信任根证书颁发机构”。

4、访问Jupyterhub

https://124.223.110.20:8000

5、配置SSL访问Rstudio Server。

Rstudio Server与Shiny Server开源的社区版都不支持SSL，只有商用版支持。因此需要配置一级前置的代理服务器，由代理服务器实现SSL的支持，[参阅资料](https://community.rstudio.com/t/rstudio-server-and-shiny-server-open-source-versions-both-were-taking-unsecured-path-http-path-even-when-the-parent-domain-is-ssl-certified/35978)。看起来Nginx的配置比Apache要简单一点，虽然以前用Apache。

参阅：配置[Proxy to Rstudio Server](https://support.rstudio.com/hc/en-us/articles/200552326-Running-RStudio-Server-with-a-Proxy)。

（1）安装配置Nginx

A、安装：

# yum install nginx

B、管理：

#systemctl enable nginx.service

#systemctl start nginx

#systemctl stop nginx

#systemctl status nginx

#systemctl restart nginx

C、配置SSL代理Rstduio：

Nginx的服务器证书要把自签CA的证书打包在一起形成完整的证书链，否则手机上访问SSL会有问题。

# cd /root/cert

# cat ca.crt >> server.crt

# vi /etc/nginx/nginx.conf

# For more information on configuration, see:

# \* Official English Documentation: http://nginx.org/en/docs/

# \* Official Russian Documentation: http://nginx.org/ru/docs/

user nginx;

worker\_processes auto;

error\_log /var/log/nginx/error.log;

pid /run/nginx.pid;

# Load dynamic modules. See /usr/share/doc/nginx/README.dynamic.

include /usr/share/nginx/modules/\*.conf;

events {

worker\_connections 1024;

}

http {

log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" '

'$status $body\_bytes\_sent "$http\_referer" '

'"$http\_user\_agent" "$http\_x\_forwarded\_for"';

access\_log /var/log/nginx/access.log main;

sendfile on;

tcp\_nopush on;

tcp\_nodelay on;

keepalive\_timeout 65;

types\_hash\_max\_size 4096;

include /etc/nginx/mime.types;

default\_type application/octet-stream;

# Load modular configuration files from the /etc/nginx/conf.d directory.

# See http://nginx.org/en/docs/ngx\_core\_module.html#include

# for more information.

include /etc/nginx/conf.d/\*.conf;

# Support proxying of web-socket connections

map $http\_upgrade $connection\_upgrade {

default upgrade;

'' close;

}

# Server with SSL enabled

server {

listen 443 ssl http2;

listen [::]:443 ssl http2;

server\_name 124.223.110.20;

# Change to point to your certs

ssl\_certificate /root/cert/server.crt;

ssl\_certificate\_key /root/cert/server.key;

ssl\_protocols TLSv1.2 TLSv1.1 TLSv1;

rewrite ^/rstudio$ $scheme://$http\_host/rstudio/ permanent;

location /rstudio/ {

# Needed only for prefix of /rstudio

rewrite ^/rstudio/(.\*)$ /$1 break;

# Use http here when ssl-enabled=0 is set in rserver.conf

proxy\_pass http://localhost:8787;

proxy\_redirect http://localhost:8787/ $scheme://$http\_host/rstudio/;

proxy\_http\_version 1.1;

proxy\_set\_header Upgrade $http\_upgrade;

proxy\_set\_header Connection $connection\_upgrade;

proxy\_read\_timeout 20d;

# In some cases, such as when streaming job statuses from the launcher, the default response buffering in nginx

# can be too slow for delivering real-time updates, especially when configured to use SSL.

# disabling response buffering

proxy\_buffering off;

}

}

}

（2）、打开防火墙的443端口（腾讯云默认）

（3）、访问。 <https://124.223.110.20/rstudio>

为了从同一个https端口分别反向代理Rstudio与Shiny Server，Nginx给它们加上了不同的前缀，Rstudio加的是/rstudio。可以关闭防火墙的8787端口，或在Rstudio Server的配置中限制8787端口只能从本机访问。为了让以前发布过的例子及URL继续可用，这里不作处理。

# vi /etc/rstudio/rserver.conf

www-address=127.0.0.1

6、配置SSL访问Shiny Server

参阅：配置[Proxy to Shiny Server](https://support.rstudio.com/hc/en-us/articles/213733868-Running-Shiny-Server-with-a-Proxy)。

（1）、配置

在上面的配置文件中增加对Shiny Server的反向代理：

……

# Server with SSL enabled

server {

……

# Reverse proxy for Shiny server

rewrite ^/shiny$ $scheme://$http\_host/shiny/ permanent;

location /shiny/ {

rewrite ^/shiny/(.\*)$ /$1 break;

proxy\_pass http://localhost:3838;

proxy\_redirect / $scheme://$http\_host/shiny/;

proxy\_http\_version 1.1;

proxy\_set\_header Upgrade $http\_upgrade;

proxy\_set\_header Connection $connection\_upgrade;

proxy\_read\_timeout 20d;

proxy\_buffering off;

}

……

}

……

（2）、访问，https在所有的URL上都加上前缀shiny，如：

https://124.223.110.20/shiny

https://124.223.110.20/shiny/radiant/

https://124.223.110.20/shiny/users/jean/weibo/

注意SSL与Admin Dashboard是商业版Shiny Server Pro才有的功能，这里也是通过前置的反向代理Nginx去实现SSL连接。

7、配置SSL访问Tomcat7

参阅：[给tomcat7配置SSL证书](https://cloud.tencent.com/developer/article/1120227)

（1）、生成p12文件:

#cd /root/cert

# openssl pkcs12 -export -inkey server.key -in server.crt -chain -CAfile \

ca.crt -out server.p12 -name 124.223.110.20 -passout pass:123456

（2）、转化为jks的keystore

# keytool -importkeystore -v -srckeystore server.p12 -srcstoretype \

pkcs12 -srcstorepass 123456 -destkeystore server.jks \

-deststoretype jks -deststorepass 123456

（3）、拷贝到Tomcat有权限的目录下

# cp server.jks /etc/tomcat/server.jks

# chown root:tomcat server.jks

（4）、配置Tomcat SSL Connector

# vi /etc/tomcat/server.xml

<Connector port="8443"

protocol="org.apache.coyote.http11.Http11Protocol"

maxThreads="150"

SSLEnabled="true"

scheme="https"

secure="true"

clientAuth="false"

sslProtocol="TLS"

keystoreFile="/etc/tomcat/server.jks"

keystorePass="123456"

/>

（5）、防火墙打开8443端口

（6）、访问。<https://124.223.110.20:8443> 。

（十五）、安装Neo4j中文社区版并配置SSL

1、升级JDK。

Neo4j4.x需要JDK11，CentOS7.6自带的是Open JDK 8，[参考资料](https://blog.csdn.net/penriver/article/details/120137308)。

# yum search java-11

安装jdk, 同时安装了依赖 java-11-openjdk-headless

# yum install java-11-openjdk.x86\_64

安装开发工具

# yum install java-11-openjdk-devel

更新设置JAVA\_HOME等：

#vi /etc/profile

# JDK11

JAVA\_HOME=/usr/lib/jvm/java-11

JRE\_HOME=/usr/lib/jvm/jre-11

PATH=$JAVA\_HOME/bin:$PATH

CLASS\_PATH=$JAVA\_HOME/lib:$CLASS\_PATH

2、下载

下载地址：<https://we-yun.com/down/neo4j-4.2.6-133.html>

# wget --no-check-certificate https://we-yun.com/doc/neo4j-chs/4.2.6/neo4j-chs-community-4.2.6-unix.tar.gz

3、安装，参阅[简体中文版指南](https://we-yun.com/doc/neo4j-chs-doc/)。

# tar -xf neo4j-chs-community-4.2.6-unix.tar.gz

# mv neo4j-chs-community-4.2.6-unix /opt/neo4j-chs-community-4.2.6-unix

4、首次启动Neo4j

# cd /opt/neo4j-chs-community-4.2.6-unix/bin

# ./neo4j console

该命令会建立各个初始目录

(base) [root@VM-4-12-centos bin]# ./neo4j console

Directories in use:

home: /opt/neo4j-chs-community-4.2.6-unix

config: /opt/neo4j-chs-community-4.2.6-unix/conf

logs: /opt/neo4j-chs-community-4.2.6-unix/logs

plugins: /opt/neo4j-chs-community-4.2.6-unix/plugins

import: /opt/neo4j-chs-community-4.2.6-unix/import

data: /opt/neo4j-chs-community-4.2.6-unix/data

certificates: /opt/neo4j-chs-community-4.2.6-unix/certificates

run: /opt/neo4j-chs-community-4.2.6-unix/run

Starting Neo4j.

2022-03-21 07:14:53.053+0000 INFO Starting...

2022-03-21 07:14:54.607+0000 INFO ======== Neo4j 4.2.6 ========

2022-03-21 07:14:56.755+0000 INFO Initializing system graph model for component 'security-users' with version -1 and status UNINITIALIZED

2022-03-21 07:14:56.762+0000 INFO Setting up initial user from defaults: neo4j

2022-03-21 07:14:56.762+0000 INFO Creating new user 'neo4j' (passwordChangeRequired=true, suspended=false)

2022-03-21 07:14:56.773+0000 INFO Setting version for 'security-users' to 2

2022-03-21 07:14:56.778+0000 INFO After initialization of system graph model component 'security-users' have version 2 and status CURRENT

2022-03-21 07:14:56.782+0000 INFO Performing postInitialization step for component 'security-users' with version 2 and status CURRENT

2022-03-21 07:14:57.065+0000 INFO Bolt enabled on localhost:7687.

2022-03-21 07:14:57.870+0000 INFO Remote interface available at http://localhost:7474/

2022-03-21 07:14:57.870+0000 INFO Started.

^C2022-03-21 07:17:48.156+0000 INFO Neo4j Server shutdown initiated by request

2022-03-21 07:17:48.156+0000 INFO Stopping...

2022-03-21 07:17:53.394+0000 INFO Stopped.

(base) [root@VM-4-12-centos bin]#

5、拷贝空白数据库neo4j备份，社区版不支持多个并发数据库，以后新建数据库时拷贝该备份并重新命名，然后切换至新的数据库即可。

# cd /opt/neo4j-chs-community-4.2.6-unix/data

# cp -R neo4j blank

6、修改配置，打开网络访问地址等，默认只侦听loopback地址 127.0.0.1：

# cd /opt/neo4j-chs-community-4.2.6-unix/conf

# vi neo4j.conf

# 打开网络访问地址

dbms.default\_listen\_address=0.0.0.0

# 启动时打开的数据库，社区版只能在线打开一个数据库，通过改变下面的名字切换。

# Change to the database you want

dbms.default\_database=neo4j

# 切换时重建事务日志，否则不能启动，因为数据库变了，不匹配。

# Create a new transaction log when change to a new database

dbms.recovery.fail\_on\_missing\_files=false

# 配置允许CSV数据文件导入，Linux上必须配置，

# 限制从其它目录导入数据，以堵塞安全漏洞。

# This setting constrains all `LOAD CSV` import files to be under the `import` directory.

dbms.directories.import=/root/R\_Linux

# Determines if Cypher will allow using file URLs when loading data using

# `LOAD CSV`. Setting this value to `false` will cause Neo4j to fail `LOAD CSV`

# clauses that load data from the file system.

dbms.security.allow\_csv\_import\_from\_file\_urls=true

#取消对APOC及GDS过程的安全限制。

# A comma separated list of procedures and user defined functions that are allowed

# full access to the database through unsupported/insecure internal APIs.

#dbms.security.procedures.unrestricted=my.extensions.example,my.procedures.\*

dbms.security.procedures.unrestricted=jwt.security.\*,gds.\*,apoc.\*

#打开用户验证，注释掉后默认是打开的。

# Whether requests to Neo4j are authenticated.

# To disable authentication, uncomment this line

#dbms.security.auth\_enabled=false

# 网络协议设置，默认在7474打开http，7687打开bolt，

# 这里关闭http，在默认的7473打开https，bolt打开SSL。

# Bolt connector

dbms.connector.bolt.enabled=true

dbms.connector.bolt.tls\_level=REQUIRED

#dbms.connector.bolt.listen\_address=:7687

#dbms.connector.bolt.advertised\_address=:7687

# HTTP Connector. There can be zero or one HTTP connectors.

dbms.connector.http.enabled=false

# HTTPS Connector. There can be zero or one HTTPS connectors.

dbms.connector.https.enabled=true

#dbms.connector.https.listen\_address=:7473

#dbms.connector.https.advertised\_address=:7473

# Bolt SSL configuration

dbms.ssl.policy.bolt.enabled=true

dbms.ssl.policy.bolt.base\_directory=/root/cert

dbms.ssl.policy.bolt.private\_key=server.key

dbms.ssl.policy.bolt.public\_certificate=server2.crt

dbms.ssl.policy.bolt.client\_auth=NONE

dbms.ssl.policy.https.enabled=true

dbms.ssl.policy.https.base\_directory=/root/cert

dbms.ssl.policy.https.private\_key=server.key

dbms.ssl.policy.https.public\_certificate=server2.crt

dbms.ssl.policy.https.client\_auth=NONE

7、生成服务器自签证书链。

这里Neo4j HTTPS仍然使用前面Shiny Server等服务器使用的同一个自签服务器数字证书，不过bolt+s协议需要使用把服务器证书与自建CA证书拼接到一起包含完整链的证书，否则Chrome浏览器不能连接bolt+s协议，具体可参阅：[参考资料1](https://stackoverflow.com/questions/54066990/neo4j-service-unavailable-websocket-connection-failure)，[参考资料2](https://stackoverflow.com/questions/61541942/how-to-enable-tls-on-bolt-protocol-on-noe4j)，[参考资料3](https://neo4j.com/docs/operations-manual/current/security/ssl-framework/)。

# cd /root/cert

# cp server.crt server2.crt

# cat ca.crt >> server2.crt

8、配置Neo4j环境变量

# vi /etc/profile

# Added for Neo4j

export NEO4J\_HOME=/opt/neo4j-chs-community-4.2.6-unix

export NEO4J\_CONF=$NEO4J\_HOME/conf

9、安装APOC、GDSL等各种plug-in

安装APOC4.2.0、GDS1.6.1等plug-in,拷贝到plugin目录,注意要和Neo4j的版本匹配，具体参阅[APOC主页](https://github.com/neo4j-contrib/neo4j-apoc-procedures)与[GDSL主页](https://neo4j.com/docs/graph-data-science/current/installation/supported-neo4j-versions/)。GDSL的最新版本是1.8，但我开发的最小树形图算算法等还没有更新到1.8，而是依赖于1.6.1，所以安装GDSL1.6.1。

# cd /opt/neo4j-chs-community-4.2.6-unix/plugins

# wget https://github.com/neo4j-contrib/neo4j-apoc-procedures/releases/download/4.2.0.9/apoc-4.2.0.9-all.jar

# wget https://github.com/neo4j/graph-data-science/releases/download/1.6.5/neo4j-graph-data-science-1.6.5.jar

重启Neo4j，测试：

//验证GDS安装

return gds.version()

call gds.list

//验证APOC安装

return apoc.version()

return apoc.math.round(1.783,0,"HALF\_UP") AS OUTPUT

这里也上传我开发的最小树形图算法proc-arborescence-0.0.1-SNAPSHOT.jar。

//测试最小树形图

//清空目前在用的图

match(n) detach delete n;

//创建一个测试图

CREATE(a:Node {name: 'a'}),

(b:Node {name: 'b'}),

(c:Node {name: 'c'}),

(d:Node {name: 'd'}),

(e:Node {name: 'e'}),

(b)-[:TYPE {cost:17}]->(a), (c)-[:TYPE {cost:16}]->(a), (d)-[:TYPE {cost:19}]->(a), (e)-[:TYPE {cost:16}]->(a),

(c)-[:TYPE {cost:3}]->(b), (d)-[:TYPE {cost:3}]->(b), (e)-[:TYPE {cost:11}]->(b),

(b)-[:TYPE {cost:3}]->(c), (d)-[:TYPE {cost:4}]->(c), (e)-[:TYPE {cost:8}]->(c),

(b)-[:TYPE {cost:3}]->(d),(c)-[:TYPE {cost:4}]->(d), (e)-[:TYPE {cost:12}]->(d),

(b)-[:TYPE {cost:11}]->(e), (c)-[:TYPE {cost:8}]->(e), (d)-[:TYPE {cost:12}]->(e);

//调用最小树形图算法

MATCH (n:Node{name: 'a'})

CALL gds.alpha.spanningArborescence.minimum.write({

nodeProjection: 'Node',

relationshipProjection: {

TYPE: {

type: 'TYPE',

properties: 'cost',

orientation: 'NATURAL'

} },

startNodeId: id(n),

relationshipWeightProperty: 'cost',

writeProperty: 'MINSA',

weightWriteProperty: 'writeCost'

})

YIELD createMillis, computeMillis, writeMillis, effectiveNodeCount

RETURN createMillis,computeMillis, writeMillis, effectiveNodeCount;

//输出最小树形图

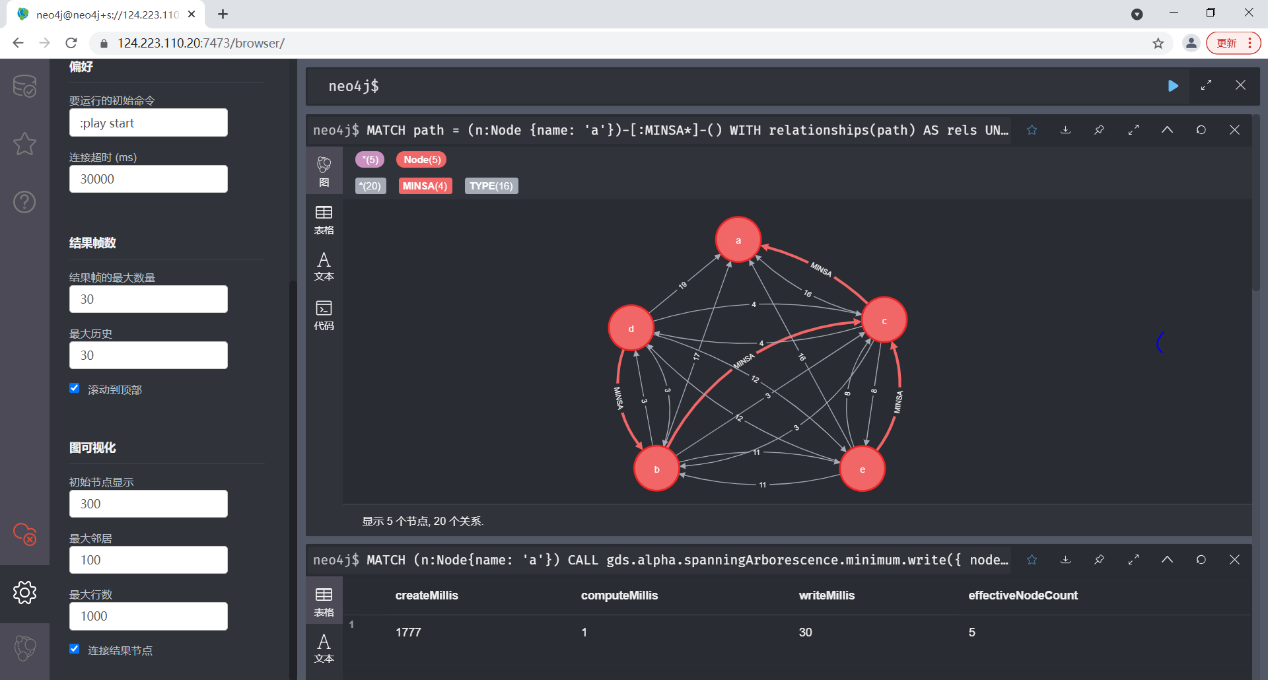
MATCH path = (n:Node {name: 'a'})-[:MINSA\*]-()

WITH relationships(path) AS rels

UNWIND rels AS rel

WITH DISTINCT rel AS rel

RETURN startNode(rel) as source, endNode(rel) AS destination, rel;



10、配置系统重启后自动启动Neo4j、Tomcat

修改/etc/rc.d/rc.local开机自动启动neo4j:

# vi /etc/rc.d/rc.local

# JDK11

JAVA\_HOME=/usr/lib/jvm/java-11

JRE\_HOME=/usr/lib/jvm/jre-11

PATH=$JAVA\_HOME/bin:$PATH

CLASS\_PATH=$JAVA\_HOME/lib:$CLASS\_PATH

CATALINA\_BASE=/usr/share/tomcat

CATALINA\_HOME=/usr/share/tomcat

export JAVA\_HOME PATH CLASSPATH CATALINA\_BASE CATALINA\_HOME

# Added for rJava

export LD\_LIBRARY\_PATH=$JAVA\_HOME/lib/server:$LD\_LIBRARY\_PATH

# Startup Tomcat

cd /usr/sbin

./tomcat start

# Added for Neo4j

export NEO4J\_HOME=/opt/neo4j-chs-community-4.2.6-unix

export NEO4J\_CONF=$NEO4J\_HOME/conf

# Startup Neo4j

cd /opt/neo4j-chs-community-4.2.6-unix/bin

./neo4j start

（十六）、安装[Rserve](https://github.com/s-u/Rserve)并配置SSL

参考： [R语言服务器程序 Rserve详解](http://r1.fens.me/%E7%AC%AC%E4%BA%94%E7%AB%A0/R%E8%AF%AD%E8%A8%80%E6%9C%8D%E5%8A%A1%E5%99%A8%E7%A8%8B%E5%BA%8FRserve%E8%AF%A6%E8%A7%A3.html)，配置用户验证。

[How to enable T.L.S 1.2 in R-Serve](https://dev.to/sirius93/how-to-enable-t-l-s-1-2-in-r-serve-4ope)，配置SSL。

[Rserve TLS SSL Support](https://github.com/s-u/Rserve/wiki/TLS-SSL-support), 选择使用的协议与参数。

[Configuration directives supported by latest Rserve version](https://github.com/s-u/Rserve/wiki/rserve.conf)。

1、安装，参阅[Rserve Github主页](https://github.com/s-u/Rserve)：

# R

>install.packages("Rserve")

2、配置：

#vi /etc/Rserv.conf

//允许远程登录

remote enable

//连接需要用户验证

auth required

//验证密码不允许明文

plaintext disable

//使用utf-8编码

encoding utf8

//允许远程控制后台R进程

control enable

//使用qap+tls协议

qap.tls.port 6311

//Rserve服务器密钥

tls.key /root/cert/server.key

//Rserve服务器证书，这里是自签证书

tls.cert /root/cert/server.crt

//可选，服务器CA证书

tls.ca /root/cert/ca.crt

//禁用非加密的qap协议

qap disable

3、配置开机自动启动

# vi /etc/rc.d/rc.local

# Startup Rserve with /etc/Rserv.conf

cd /usr/lib64/R-4.1.2/bin

./R CMD Rserve --no-save

4、本机测试SSL连接，可以与配好的Nginx SSL端口比较一下以确认。

#openssl s\_client -connect localhost:6311 -tls1\_2

#openssl s\_client -connect localhost:443 -tls1\_2

5、本机测试R客户端连接，先用localhost再用外网IP测试。

#R

>install.packages(“RSclient”)

>library(RSclient)

>conn<-RS.connect(host="124.223.110.20",tls=TRUE, verify=FALSE)

>RS.login(conn,"user","password",authkey=RS.authkey(conn))

>RS.eval(conn,rnorm(5))

6、客户端Rstudio测试R客户端连接，同上。

7、增加一个专用Linux系统用户以用于远程访问Rserve，以增加安全性。

#adduser rserve

#passwd rserve

8、Tomcat及J2EE客户端程序连接qap+tls协议Rserve的配置。

Rserve Java客户端类库[REngin](https://github.com/s-u/REngine)的老版本并不支持SSL连接，最新的源码中已经加入了支持，要下载最新的源码重新编译打包。在Github主页上按Code->Download ZIP下载源码后解压，打开命令行窗口,转到在解压目录下：

make

即可编译打包（用的是make工具！）。为了保持兼容性，用JDK8来编译（Neo4j要JDK11）。然后参考第五章《社保明细申报数据分析》第四节《扩面分析：个税社保比对模型》第九小节《Rstudio Linux Server与J2EE的集成》去开发java客户端连接Rserve的程序，用最新的类库包REngine.jar、Rserve.jar替换旧版。在增加SSL连接后，程序部分要稍作修改，这里主要讲一下修改的地方。

参考资料：

Stackoverflow.com: [Java Rconnection hangs when Rserve has TLS enabled](https://stackoverflow.com/questions/55345446/java-rconnection-hangs-when-rserve-has-tls-enabled) 。

REngin Github主页 [RConnection.java](https://github.com/s-u/REngine/blob/master/Rserve/RConnection.java)源码。

（1）、把自建CA的根证书导入JDK的受信任根证书库中。

Tomcat要使用它。JDK8与JDK11已经没有jre目录，为保持兼容可以自己生成，反正找到JDK的lib/security目录，cacerts文件在该目录下。

>cd /jkd8/jre/lib/security

>keytool -import -alias Jean -file D:/temp/ca.crt -keystore cacerts -storepass changeit

>keytool -list -keystore cacerts -storepass changeit

Linux上也是一样的：

# keytool -import -alias Jean -file /root/cert/ca.crt -keystore cacerts -storepass changeit

#keytool -list -keystore cacerts -storepass changeit

（2）、在Eclipse+Tomcat上开发Java Rserve客户端程序。

具体参考上面提到的第五章第四节第九小节，好久没有写java WEB程序，Eclipse具体配置可以参考[这篇文章](https://blog.csdn.net/yerenyuan_pku/article/details/51830104)。

主要的变化就是先根据Rserve的主机与端口号初始化一个SSLSocket，然后以用户名+密码远程登录，我用别人写好的org.math.R包稍作修改，代码示例如下：

自己写的RServeHelper.java:

package test;

import java.io.IOException;

import java.util.Properties;

import org.math.R.RserveSession;

import org.math.R.RserverConf;

import org.math.R.Rsession;

public class RServeHelper {

// private static boolean isWindows = false;

private static Rsession *rsession* = null;

private static String *host* = "124.223.110.20";

//public static String host="127.0.0.1";

public static String *prefix* = "../../../../home/jean/R/";

//public static String prefix="C:/Users/lenovo/Documents/Rscripts/";

public static String *rpics* = "/tmp/Rpics";

//public static String rpics="C:/Users/lenovo/Documents/Rpics";

public static Rsession initRserve() throws IOException {

Properties prop = new Properties();

prop.setProperty("tls", "true");

RserverConf rconf = new RserverConf(*host*, 6311, "user", "password", prop);

*rsession* = RserveSession.*newInstanceTry*(System.*out*, rconf);

return *rsession*;

}

public static Rsession getRsessionInstance() throws IOException {

if (*rsession* == null) {

*rsession* = *initRserve*();

}

return *rsession*;

}

public static void endRsession(Rsession rs) {

rs.end();

*rsession* = null;

}

}

别人写的org.math.R.RserverConf.java:

**package** org.math.R;

**import** java.io.IOException;

**import** java.net.BindException;

**import** java.net.ServerSocket;

**import** java.util.Properties;

**import javax.net.ssl.SSLSocket;**

**import** javax.net.ssl.SSLSocketFactory;

**import** org.rosuda.REngine.Rserve.RConnection;

**import** org.rosuda.REngine.Rserve.RserveException;

**public** **class** RserverConf {

**public** **static** String *DEFAULT\_RSERVE\_HOST* = "localhost";

RConnection connection;

**public** String host;

**public** **int** port;

**public** String login;

**public** String password;

//public String RLibPath;

**public** Properties properties;

//public String http\_proxy;

**public** RserverConf(String RserverHostName, **int** RserverPort, String login, String password, Properties props) {

**this**.host = RserverHostName;

**this**.port = RserverPort;

**this**.login = login;

**this**.password = password;

properties = props;

}

**public** **synchronized** RConnection connect() {

//Logger.err.print("Connecting " + toString()+" ... ");

TimeOut t = **new** TimeOut() {

**protected** Object defaultResult() {

**return** -2;

}

**protected** Object command() {

// Added by Jean 2022/03/25

String tls = "false";

**if** (properties != **null**) {

**try** {

tls = properties.getProperty("tls");

**if** (tls == **null** || tls != "true")

tls = "false";

} **catch** (Exception ex) {

}

}

**if** (tls == "false") {

**try** {

**if** (host == **null**) {

**if** (port > 0) {

connection = **new** RConnection(*DEFAULT\_RSERVE\_HOST*, port);

} **else** {

connection = **new** RConnection();

}

**if** (connection.needLogin()) {

connection.login(login, password);

}

} **else** {

**if** (port > 0) {

connection = **new** RConnection(host, port);

} **else** {

connection = **new** RConnection(host);

}

**if** (connection.needLogin()) {

connection.login(login, password);

}

}

**return** 0;

} **catch** (Exception ex) {

Log.*Err*.println("Failed to connect: " + ex.getMessage());

**return** -1;

}

// Added by Jean for tls connection 2022/03/25

} **else** {

SSLSocketFactory sslsocketfactory = (SSLSocketFactory) SSLSocketFactory.*getDefault*();

SSLSocket sslsocket;

**try** {

**if** (host == **null**) {

host = *DEFAULT\_RSERVE\_HOST*;

}

**if** (port > 0) {

} **else** {

port = 6311;

}

// connection = new RConnection();

sslsocket = (SSLSocket) sslsocketfactory.createSocket(host, port);

// Connect to the remote server.

connection = **new** RConnection(sslsocket);

**if** (connection.needLogin()) {

connection.login(login, password);

}

**return** 0;

} **catch** (Exception ex) {

Log.*Err*.println("Failed to connect: " + ex.getMessage());

**return** -1;

}

}

}

};

**try** {

t.execute(*CONNECT\_TIMEOUT*);

} **catch** (Exception e) {

Log.*Err*.println(" failed: " + e.getMessage());

}

**if** (connection != **null** && connection.isConnected()) {

**if** (properties != **null**) {

**for** (String p : properties.stringPropertyNames()) {

**try** {

connection.eval("Sys.setenv(" + p + "='" + properties.getProperty(p) + "')");

} **catch** (RserveException ex) {

ex.printStackTrace();

}

}

}

//Logger.err.println("Connection " + toString()+" succeded.");

**return** connection;

} **else** {

Log.*Err*.println("Connection " + toString() + " failed.");

**return** **null**;

}

}