语言实验数据预处理

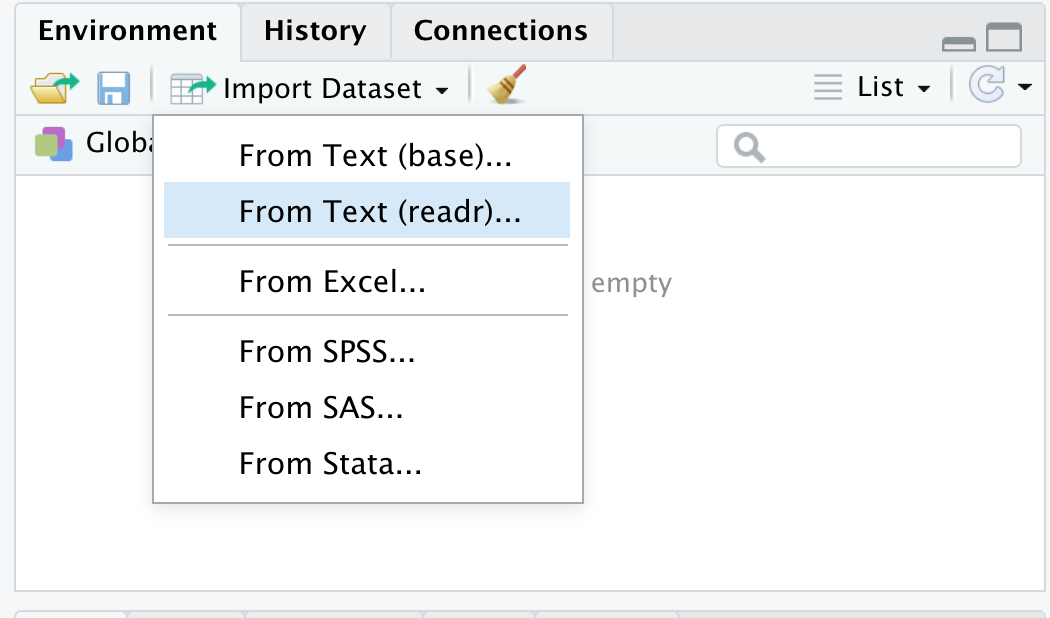
使用R语言进行数据预处理，涉及数据导入、清洗以及转化。本章中，首先介绍结构化数据的导入、清洗、转化、预览。

结构化数据是数据科学当中相对容易处理的一种数据类型。在语言研究中，实验数据通常是结构化数据。实验数据可以非常简单，例如一个教学实验中两个对照组各有30名学生，他们在实验前、中、后成绩的变化。这样的数据集，通常只有几十行和若干列。常规的图表工具如Excel也可以轻松的实现数据清洗和转化。实验数据也可以非常复杂，比如心理语言学实验中，一个被试可能有几百次反应，同时每个反应有对应的时间，实验条件等等。这样的数据集通常有几万行。使用Excel处理就容易出错，效率也不高。R语言中，我们通过使用数据处理的程序包中的相应函数，可以方便地对于数据进行筛选，重组，计算。

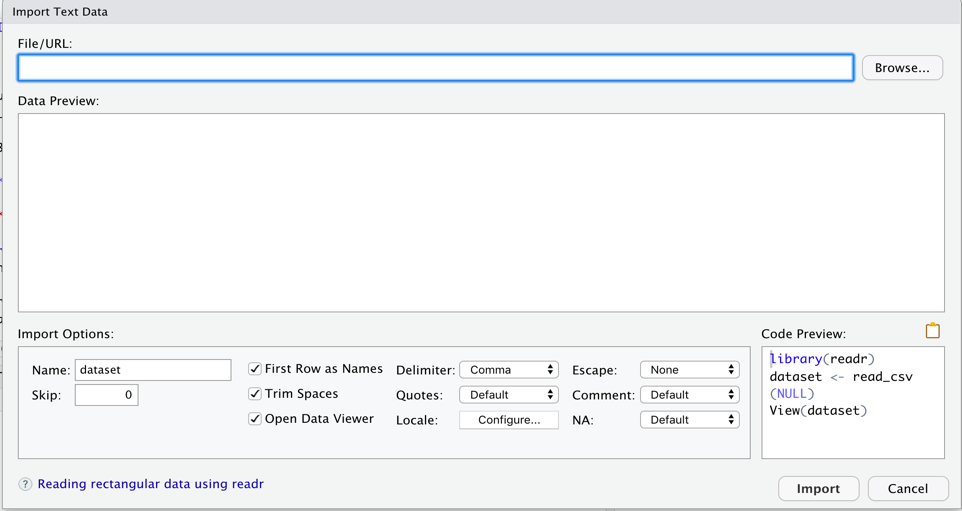
数据导入

结构化数据通常以.csv格式或者其他的分隔符（tab）格式存储。下面分别介绍如何将数据导入R-studio

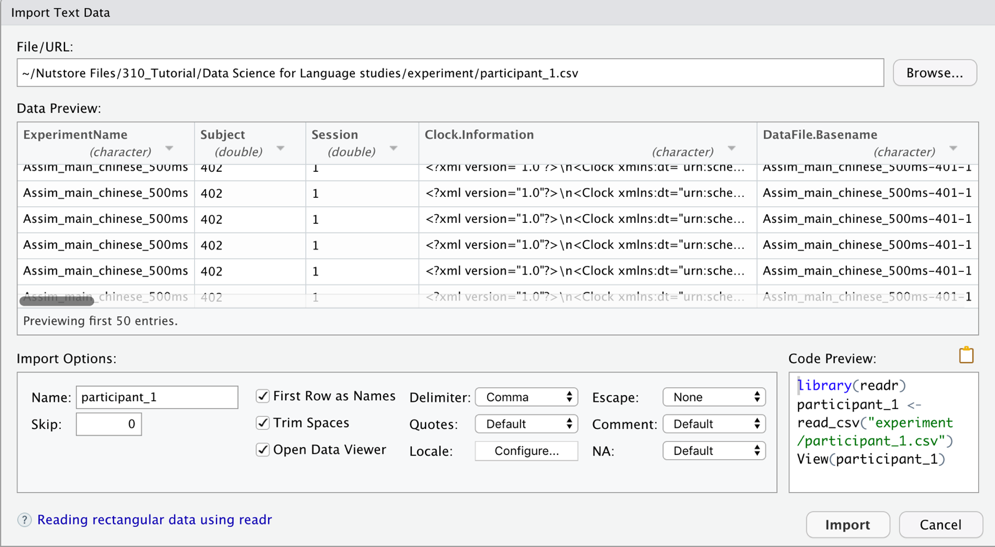
第一种方法：通过R-studio的图形化交互界面。选择import Dataset。对于csv文件选择，From Text（readr）。此时将弹出对话框。



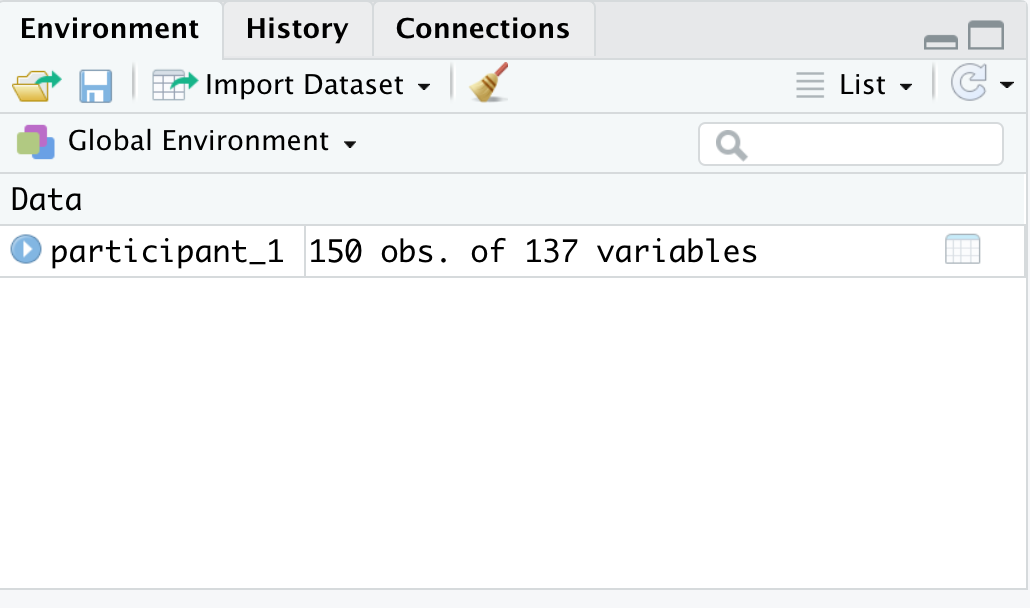
在对话框中，选择文件路径，选取文件。



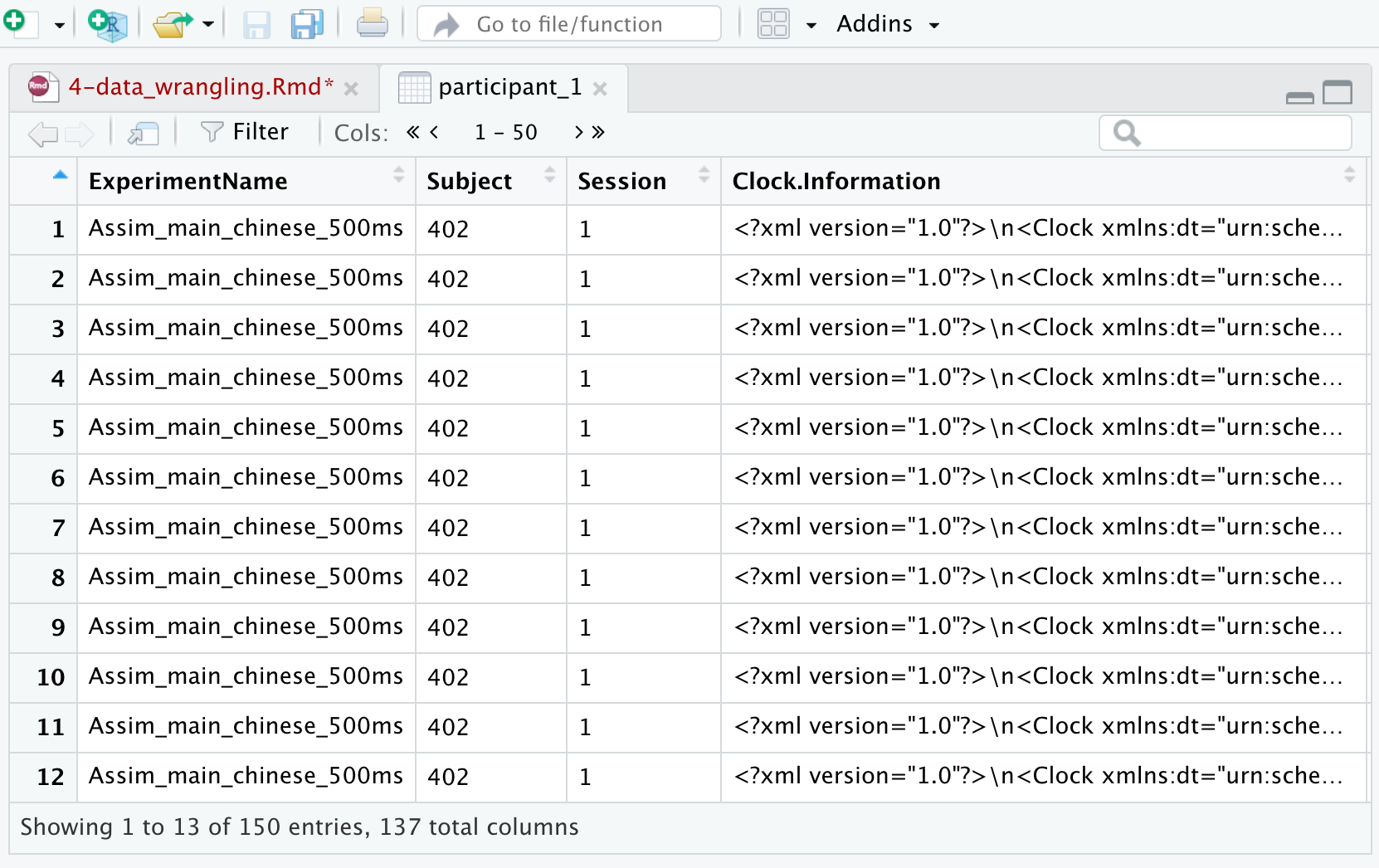
选择完文件后，对话框会显示已经导入的CSV文件的预览。这是点击Import即可。



导入成功后，在环境窗口显示有一个数据框变量，有150行和137列。



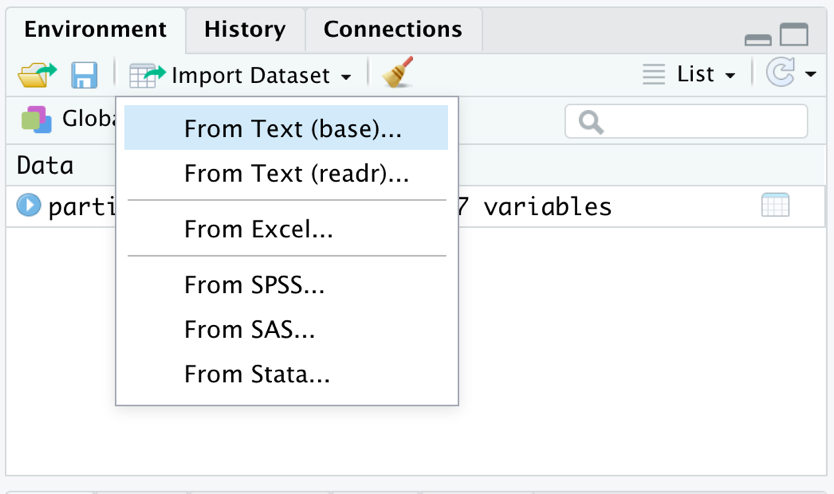
点击变量名称，可以在左边的数据窗口看到数据框的预览。



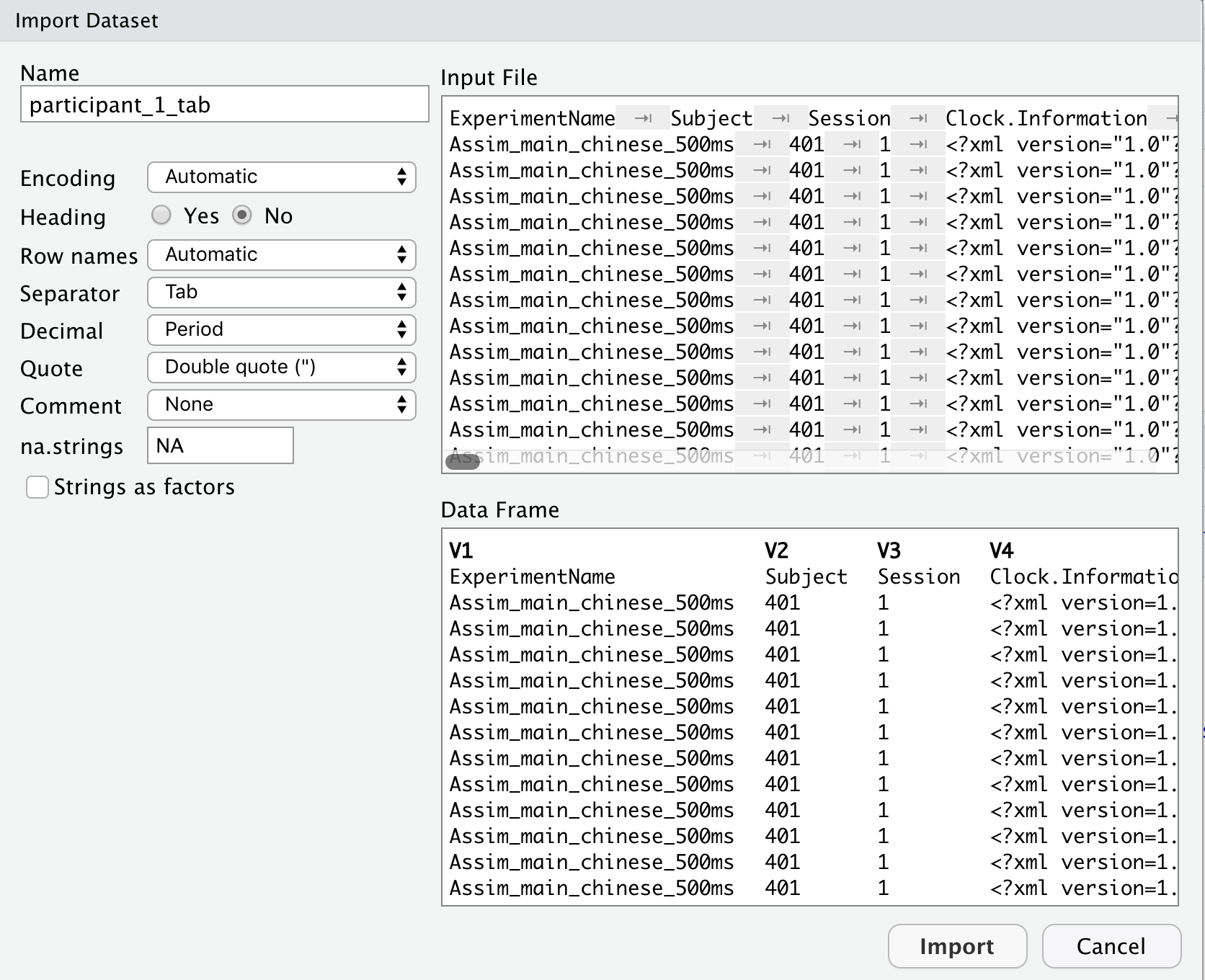
第二种方法导入，使用代码。可以拷贝右下角代码区的代码

TXT文件的导入

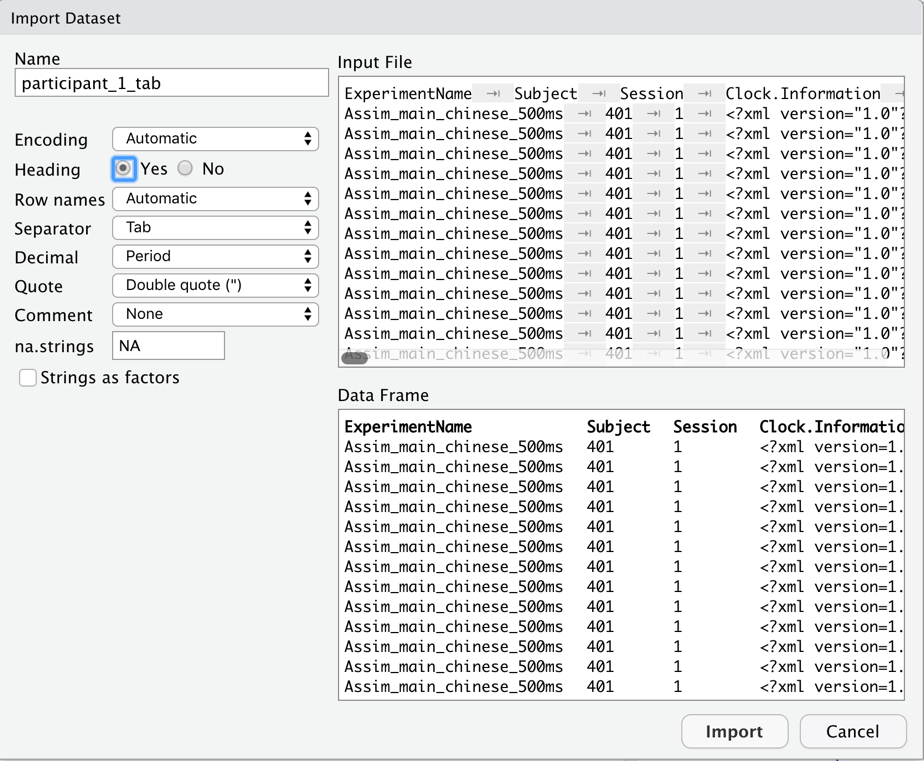
首先，选择import Dataset。如果是txt格式，选择From Text（readr）。此时将弹出对话框。



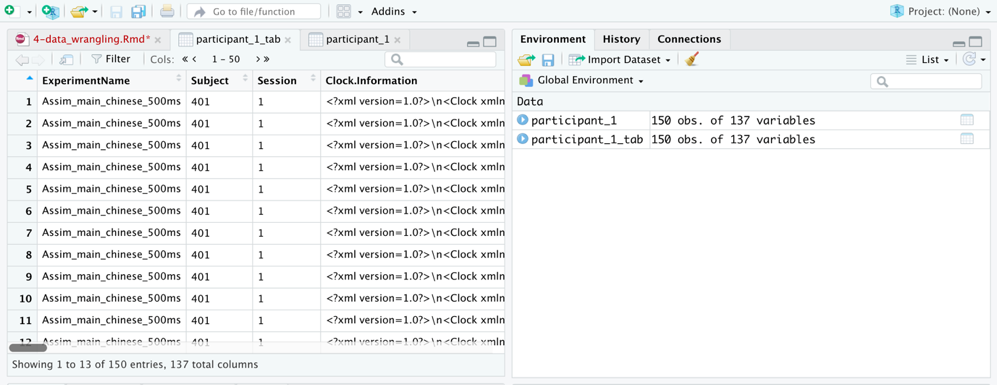
然后同样选择文件路径和文件。



注意此时右下角的数据框显示，列名称被当做第一行数据导入，这样会导致数据错乱。我们需要在左上角的Heading出点击yes。



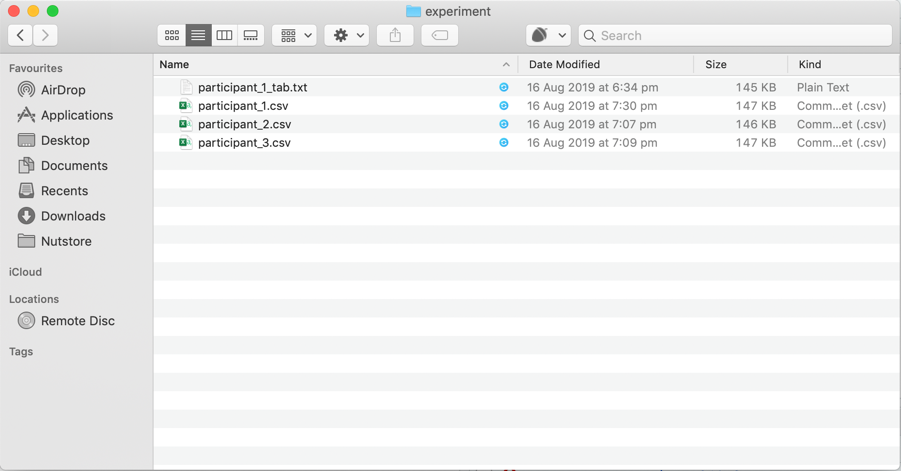
导入成功以后我们可以发现，环境变量中已经有了新的变量。点击可以激活预览视图（左边）。



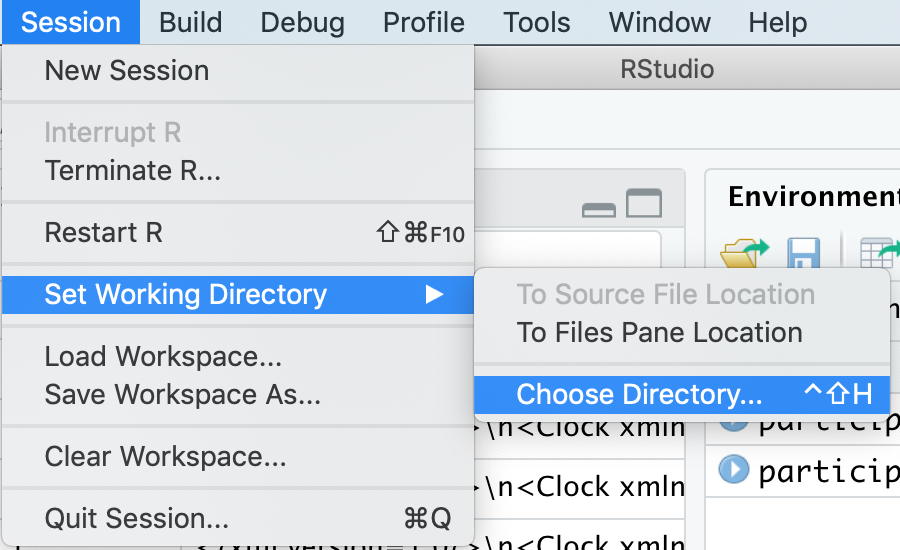
|  |  |  |
| --- | --- | --- |
| 文件类型 | R基础函数 | **readr 程序包** |
| csv | read.csv( ) | read\_csv( ) |
| txt | read.delim( ) | read\_delim( ) |
|  |  |  |

批量导入

很多时候，实验数据是以个人为单位存储的。如果有10个100个文件，我们需要写一个简单的循环语句来实现。



首先，我们要设置工作文件夹。



library(tidyverse)

full\_path = list.files(  
 path = "./experiment",  
 pattern = "\*.csv$",  
 recursive = TRUE,  
 full.names = TRUE  
)  
  
# 批量导入  
df = data.frame()  
bin = data.frame()  
for (i in 1:length(full\_path)){  
 # length函数自动获取文件个数  
 bin = read.csv(full\_path[i])  
 df = rbind(df, bin)  
}

## 预览数据框

在R当中有两个基本的函数，可以帮助我们快速地了解数据框中，变量个数、属性。

summary( )

str( )

或者你也可以将鼠标移动到变量名称也就是列名称的上方，在R-studio里面你可以看到该列的简单情况。

数据整理

数据顺利导入到R语言内部以后，我们要对数据进行一定的清理。从实验软件导出的原始数据中，包含了很多信息列（变量），但是这些信息并不都是我们感兴趣的变量，很多信息仅仅是软件记录的环境参数。因此，我们需要知道我们感兴趣的变量有哪些，并且从数据框中将之提取出来。

在我们刚刚导入这个数据集df当中，我们感兴趣的变量有：参与者的编号（Subject）、他们所听到的刺激内容（tone.Trial.）、他们所做出的反应（insex1.RESP）、他们的反应时（insex1.RT），实验组块的编号（Procedure.Block.）以及实验编号（ExperimentName）。

下面我们要使用tydiverse当中的select（）函数来将这些变量提取出来并重新命名。

#

df\_new = select (df, subject = "Subject",  
 stimuli = "tone.Trial.",  
 response = "insex1.RESP",  
 response\_rt = "insex1.RT",  
 block = "Procedure.Block.",  
 ExperimentName)  
select（）函数可以通过不同的方式提取变量，比如变量名称，变量所在的行号。

# 通过变量的名称提取  
head(select(df\_new, subject: response))

## subject stimuli response  
## 1 402 NA   
## 2 402 NA   
## 3 402 NA   
## 4 402 NA   
## 5 402 NA   
## 6 402 NA

# 通过列的编号提取  
head(select(df\_new, 1: 3))

## subject stimuli response  
## 1 402 NA   
## 2 402 NA   
## 3 402 NA   
## 4 402 NA   
## 5 402 NA   
## 6 402 NA

# 不选择某一列  
head(select(df\_new, -subject))

## stimuli response response\_rt block ExperimentName  
## 1 NA NA pracproc Assim\_main\_chinese\_500ms  
## 2 NA NA pracproc Assim\_main\_chinese\_500ms  
## 3 NA NA pracproc Assim\_main\_chinese\_500ms  
## 4 NA NA pracproc Assim\_main\_chinese\_500ms  
## 5 NA NA pracproc Assim\_main\_chinese\_500ms  
## 6 NA NA pracproc Assim\_main\_chinese\_500ms

# 改变列的名称。这里名称太长我们换成一个较为简洁的名称方便后续的编程  
head(rename(df\_new, exp = ExperimentName))

## subject stimuli response response\_rt block exp  
## 1 402 NA NA pracproc Assim\_main\_chinese\_500m  
## 2 402 NA NA pracproc Assim\_main\_chinese\_500m  
## 3 402 NA NA pracproc Assim\_main\_chinese\_500m  
## 4 402 NA NA pracproc Assim\_main\_chinese\_500m  
## 5 402 NA NA pracproc Assim\_main\_chinese\_500m  
## 6 402 NA NA pracproc Assim\_main\_chinese\_500m

选择观测值（行）

很多时候数据原始值当中，会存在一些错漏，或者我们根据研究问题，会选择其中一些观测值。此时我们可以使用filter（）函数来选取观测值（行）。

当我们有很多条件来进行筛选的时候，我们可以使用一些逻辑符号，将这些条件串联在一起。常用的逻辑运算符有或（|）、与（&）、非（！）。注意这些符号必须是英文状态下输入。

# 删除练习组（pracproc）的数据  
df\_new = filter(df\_new, block != "pracproc")  
head(df\_new)

## subject stimuli response response\_rt block ExperimentName  
## 1 402 33 f 672 block6 Assim\_main\_chinese\_500ms  
## 2 402 315 j 2831 block6 Assim\_main\_chinese\_500ms  
## 3 402 45 0 block6 Assim\_main\_chinese\_500ms  
## 4 402 21 0 block6 Assim\_main\_chinese\_500ms  
## 5 402 33 f 1041 block6 Assim\_main\_chinese\_500ms  
## 6 402 241 0 block6 Assim\_main\_chinese\_500ms

# 其次我们需要删除未作答的记录  
df\_new = filter(df\_new, !is.na(response))  
df\_new = filter(df\_new, response != "")  
head(df\_new)

## subject stimuli response response\_rt block ExperimentName  
## 1 402 33 f 672 block6 Assim\_main\_chinese\_500ms  
## 2 402 315 j 2831 block6 Assim\_main\_chinese\_500ms  
## 3 402 33 f 1041 block6 Assim\_main\_chinese\_500ms  
## 4 402 315 j 363 block6 Assim\_main\_chinese\_500ms  
## 5 402 21 f 1234 block6 Assim\_main\_chinese\_500ms  
## 6 402 45 j 322 block6 Assim\_main\_chinese\_500ms

# 挑选出答案为33和21的相应记录  
  
head(filter(df\_new, stimuli == 33 | stimuli == 21))

## subject stimuli response response\_rt block ExperimentName  
## 1 402 33 f 672 block6 Assim\_main\_chinese\_500ms  
## 2 402 33 f 1041 block6 Assim\_main\_chinese\_500ms  
## 3 402 21 f 1234 block6 Assim\_main\_chinese\_500ms  
## 4 402 33 f 150 block6 Assim\_main\_chinese\_500ms  
## 5 402 33 f 206 block6 Assim\_main\_chinese\_500ms  
## 6 402 21 f 52 block6 Assim\_main\_chinese\_500ms

head(filter(df\_new, stimuli %in% c(33, 21)))

## subject stimuli response response\_rt block ExperimentName  
## 1 402 33 f 672 block6 Assim\_main\_chinese\_500ms  
## 2 402 33 f 1041 block6 Assim\_main\_chinese\_500ms  
## 3 402 21 f 1234 block6 Assim\_main\_chinese\_500ms  
## 4 402 33 f 150 block6 Assim\_main\_chinese\_500ms  
## 5 402 33 f 206 block6 Assim\_main\_chinese\_500ms  
## 6 402 21 f 52 block6 Assim\_main\_chinese\_500ms

# 挑选出反应时小于或者大于200毫秒的记录  
  
head(filter(df\_new, response\_rt < 200))

## subject stimuli response response\_rt block ExperimentName  
## 1 402 45 j 109 block6 Assim\_main\_chinese\_500ms  
## 2 402 33 f 150 block6 Assim\_main\_chinese\_500ms  
## 3 402 21 f 52 block6 Assim\_main\_chinese\_500ms  
## 4 402 33 f 107 block5 Assim\_main\_chinese\_500ms  
## 5 402 33 f 153 block5 Assim\_main\_chinese\_500ms  
## 6 402 21 f 95 block2 Assim\_main\_chinese\_500ms

head(filter(df\_new, response\_rt > 200))

## subject stimuli response response\_rt block ExperimentName  
## 1 402 33 f 672 block6 Assim\_main\_chinese\_500ms  
## 2 402 315 j 2831 block6 Assim\_main\_chinese\_500ms  
## 3 402 33 f 1041 block6 Assim\_main\_chinese\_500ms  
## 4 402 315 j 363 block6 Assim\_main\_chinese\_500ms  
## 5 402 21 f 1234 block6 Assim\_main\_chinese\_500ms  
## 6 402 45 j 322 block6 Assim\_main\_chinese\_500ms

## 改变行的排列顺序

# 为了方便观察，有的时候需要设定记录排列的条件。这里按照stimuli和response\_rt两个变量排列。  
head(arrange(df\_new, stimuli, response\_rt))

## subject stimuli response response\_rt block ExperimentName  
## 1 418 21 g 6 block7 assim\_main\_vietnamese\_500ms  
## 2 418 21 g 51 block4 assim\_main\_vietnamese\_500ms  
## 3 402 21 f 52 block6 Assim\_main\_chinese\_500ms  
## 4 418 21 g 85 block1 assim\_main\_vietnamese\_500ms  
## 5 402 21 f 95 block2 Assim\_main\_chinese\_500ms  
## 6 418 21 g 99 block7 assim\_main\_vietnamese\_500ms

# 这里按照stimuli和block倒序排列。  
head(arrange(df\_new, stimuli, desc(block)))

## subject stimuli response response\_rt block ExperimentName  
## 1 402 21 f 609 block7 Assim\_main\_chinese\_500ms  
## 2 402 21 f 224 block7 Assim\_main\_chinese\_500ms  
## 3 402 21 f 286 block7 Assim\_main\_chinese\_500ms  
## 4 402 21 f 1136 block7 Assim\_main\_chinese\_500ms  
## 5 418 21 g 6 block7 assim\_main\_vietnamese\_500ms  
## 6 418 21 g 331 block7 assim\_main\_vietnamese\_500ms

## 数据的转化

当我们完成了变量的选择，并且筛选出我们需要的符合要求的记录之后，我们可以基于现有的数据进行一些转化。比如，在一些时候，我们需要对反应时数据，进行取对数的操作。

# 我们需要对反应时数据，进行取对数的操作  
head(mutate(df\_new, response\_rt\_new = log(response\_rt)))

## subject stimuli response response\_rt block ExperimentName  
## 1 402 33 f 672 block6 Assim\_main\_chinese\_500ms  
## 2 402 315 j 2831 block6 Assim\_main\_chinese\_500ms  
## 3 402 33 f 1041 block6 Assim\_main\_chinese\_500ms  
## 4 402 315 j 363 block6 Assim\_main\_chinese\_500ms  
## 5 402 21 f 1234 block6 Assim\_main\_chinese\_500ms  
## 6 402 45 j 322 block6 Assim\_main\_chinese\_500ms  
## response\_rt\_new  
## 1 6.510258  
## 2 7.948385  
## 3 6.947937  
## 4 5.894403  
## 5 7.118016  
## 6 5.774552

或者我们需要从字符串变量当中提取一些信息。比如，在这里我们需要提取实验的一个条件是500毫秒还是2000毫秒。

# 从*ExperimentName中提取2000和500*  
  
head(mutate(df\_new, ISI = str\_extract(ExperimentName, "2000|500")))

## subject stimuli response response\_rt block ExperimentName ISI  
## 1 402 33 f 672 block6 Assim\_main\_chinese\_500ms 500  
## 2 402 315 j 2831 block6 Assim\_main\_chinese\_500ms 500  
## 3 402 33 f 1041 block6 Assim\_main\_chinese\_500ms 500  
## 4 402 315 j 363 block6 Assim\_main\_chinese\_500ms 500  
## 5 402 21 f 1234 block6 Assim\_main\_chinese\_500ms 500  
## 6 402 45 j 322 block6 Assim\_main\_chinese\_500ms 500

# you can paste two variabls together  
  
head(transmute(df\_new, ISI = paste(stimuli, block, sep = "\_")))

## ISI  
## 1 33\_block6  
## 2 315\_block6  
## 3 33\_block6  
## 4 315\_block6  
## 5 21\_block6  
## 6 45\_block6

# you can recode a variable  
  
head( mutate(df\_new, block = recode(block,   
 block1 = "ss", block2 = "ss",  
 block3 = "sd", block4 = "sd",  
 block5 = "ds", block6 = "ds",  
 block7 = "dd")))

## subject stimuli response response\_rt block ExperimentName  
## 1 402 33 f 672 ds Assim\_main\_chinese\_500ms  
## 2 402 315 j 2831 ds Assim\_main\_chinese\_500ms  
## 3 402 33 f 1041 ds Assim\_main\_chinese\_500ms  
## 4 402 315 j 363 ds Assim\_main\_chinese\_500ms  
## 5 402 21 f 1234 ds Assim\_main\_chinese\_500ms  
## 6 402 45 j 322 ds Assim\_main\_chinese\_500ms

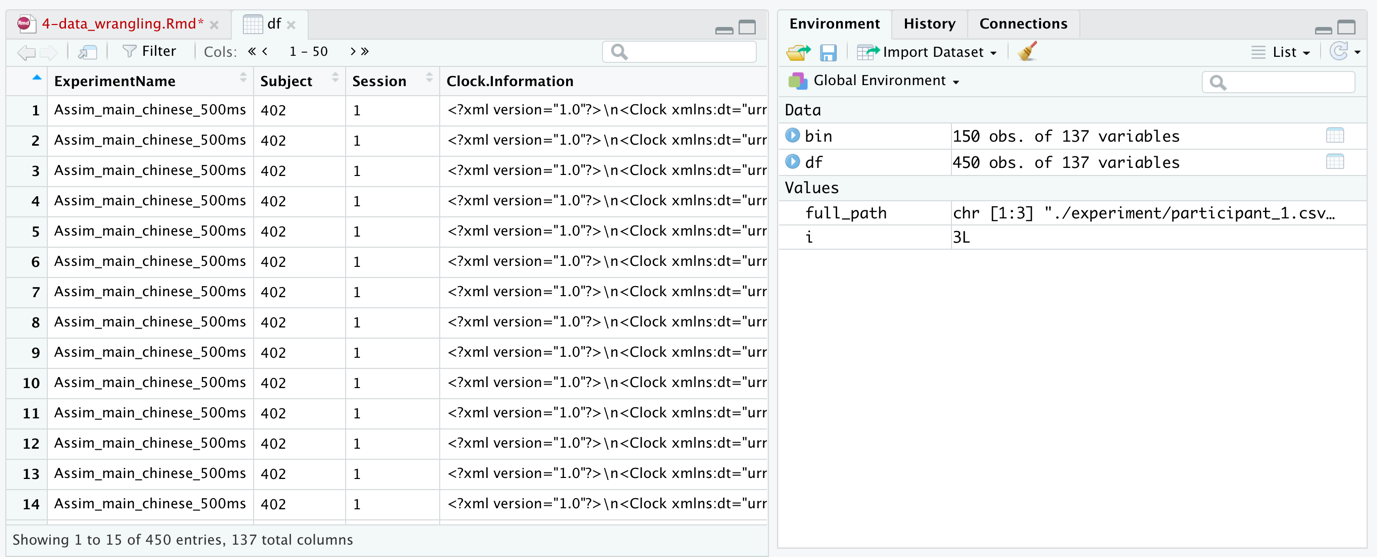
## 

## 管道pipeline “%>%”

到目前为止，数据分析过程中，函数都是单独使用的。因此，在每个函数使用过程当中，我们需要标明相应数据框名称。为了提高数据分析效率，减少有可能出现的人工错误，提高数据分析的可再生性。在这里推荐使用管道编程的方法。通过管道相应的符号“%>%”，我们可以在管道的一头输入原始数据，让数据按照顺序，在不同的函数之间传递。最后，处理完成的新数据，从管道的另一头输出。

下面我们展示如何将上文中的数据处理过程，通过管道的形式进行编程。同时我们使用管道所构建的命令语块。可以在后续分析中，转换为一个自定义函数。进一步提高数据处理的效率。

df\_final = df %>%  
 # selecting the columns we need  
 select(., subject = "Subject",  
 stimuli = "tone.Trial.",  
 response = "insex1.RESP",  
 response\_rt = "insex1.RT",  
 block = "Procedure.Block.",  
 exp = ExperimentName)%>%  
 #filtering out useless data  
 filter(block != "pracproc" & !is.na(response) & response != "")%>%  
 # generating new variables based on old variables  
 mutate(ISI = str\_extract(exp, "2000|500"),  
 block = recode(block,   
 block1 = "ss", block2 = "ss",  
 block3 = "sd", block4 = "sd",  
 block5 = "ds", block6 = "ds",  
 block7 = "dd"))



自定义函数

为了进一步提高提高数据处理的效率，我们可以将数据处理流程中，反复使用的代码块定义为一个函数。比如，在处理上面的实验数据中，我们可以将数据清洁的过程定义为一个函数。

data\_clean = function(input){

#copy and paste the chunk we wrote earlier

# change this line "df\_final = df %>% to

input %>%

# selecting the columns we need

select(., subject = "Subject",

stimuli = "tone.Trial.",

response = "insex1.RESP",

response\_rt = "insex1.RT",

block = "Procedure.Block.",

exp = ExperimentName)%>%

#filtering out useless data

filter(block != "pracproc" & !is.na(response) & response != "")%>%

# generating new variables based on old variables

mutate(ISI = str\_extract(exp, "2000|500"),

block = recode(block,

block1 = "ss", block2 = "ss",

block3 = "sd", block4 = "sd",

block5 = "ds", block6 = "ds",

block7 = "dd")) -> output

return(output)

}

这样我们，可以通过调用函数来实现数据处理。

participant\_1\_clean = data\_clean(participant\_1)

participant\_2\_clean = data\_clean(participant\_2)