phuse·share
connect·share·advance

Single Day Event

Data Exchange between SAS and R for NLP Feature Engineering

Lee Wan,
Tigermed

phuse.global

Agenda

- Introduction
- Data Exchange Challenges
- Methods for Exchanging Data between SAS and R
- Best Practices for Data Exchange
- Conclusion

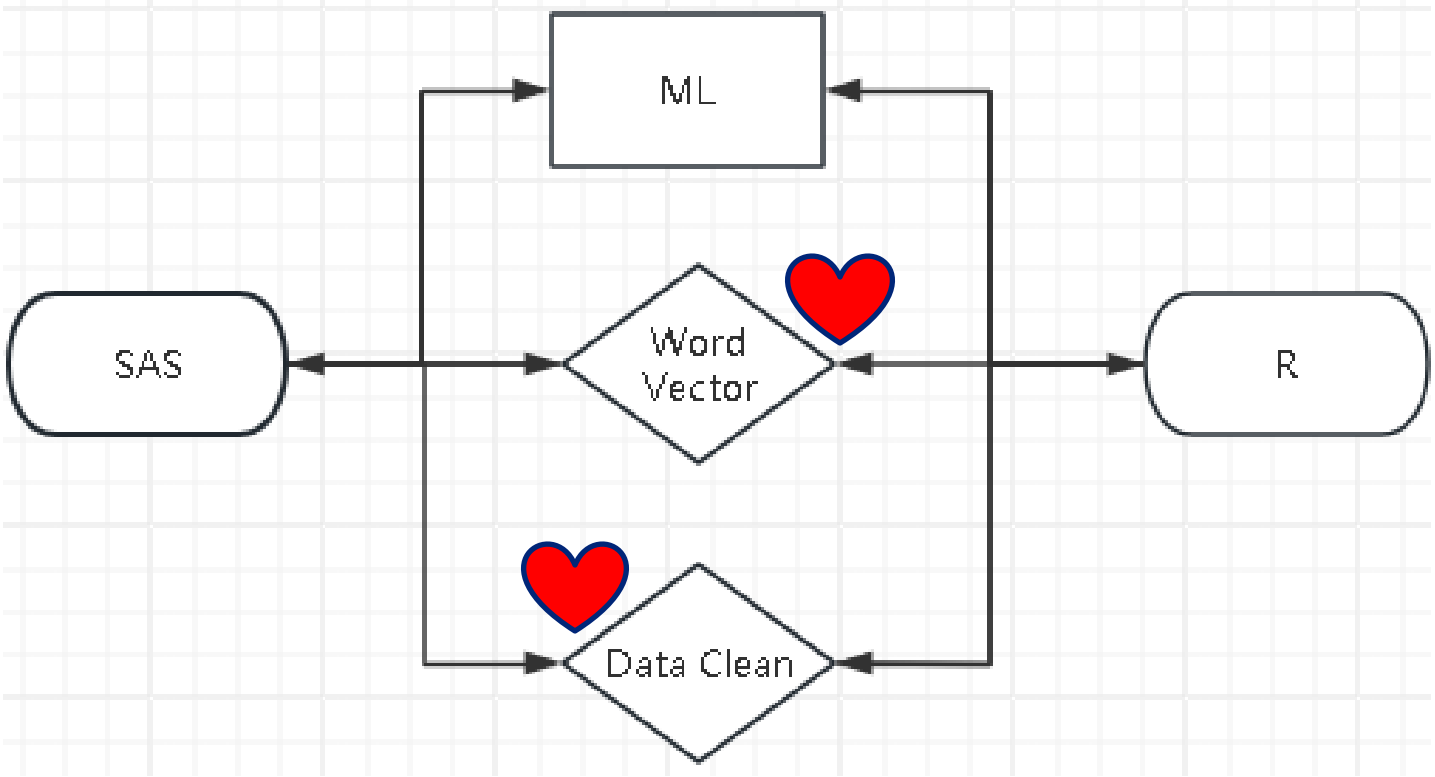
Introduction

Text Cleaning

Word Vector Generation

Feature Extraction

Introduction

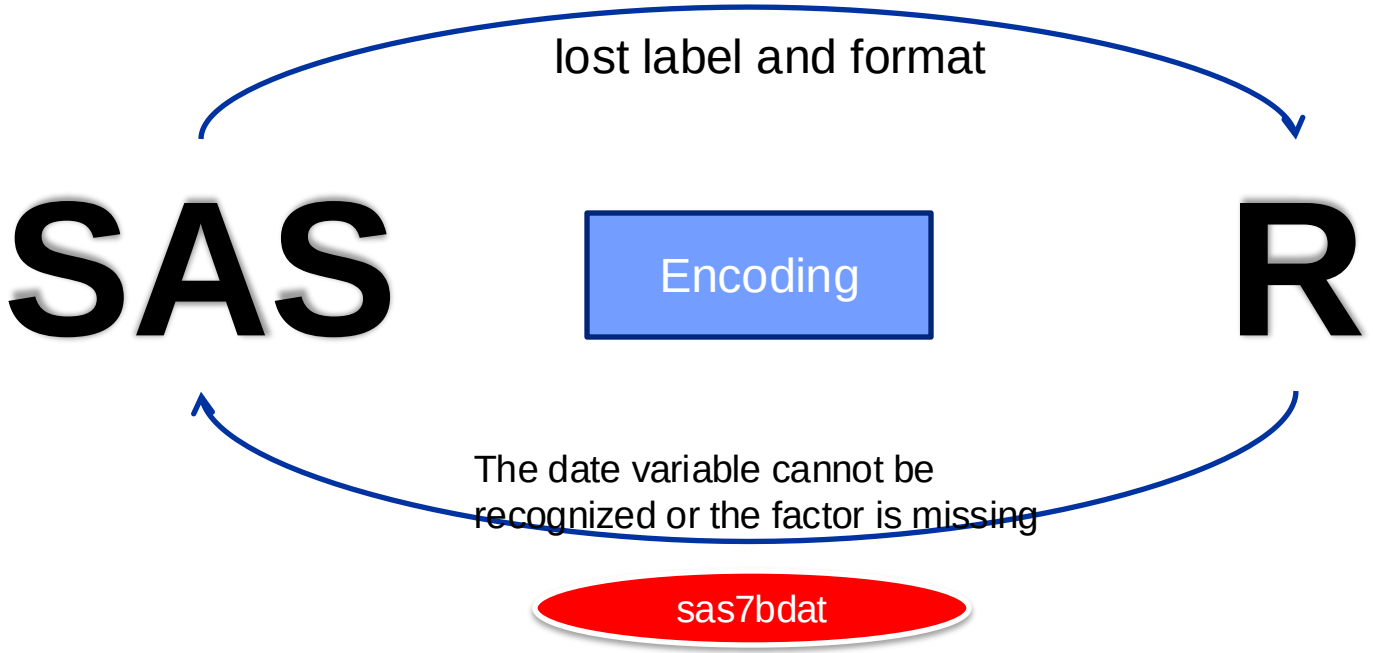


Differences in Data Structures

SAS
Label
Format
No Date

R
No Label
No Format
Date
Factor

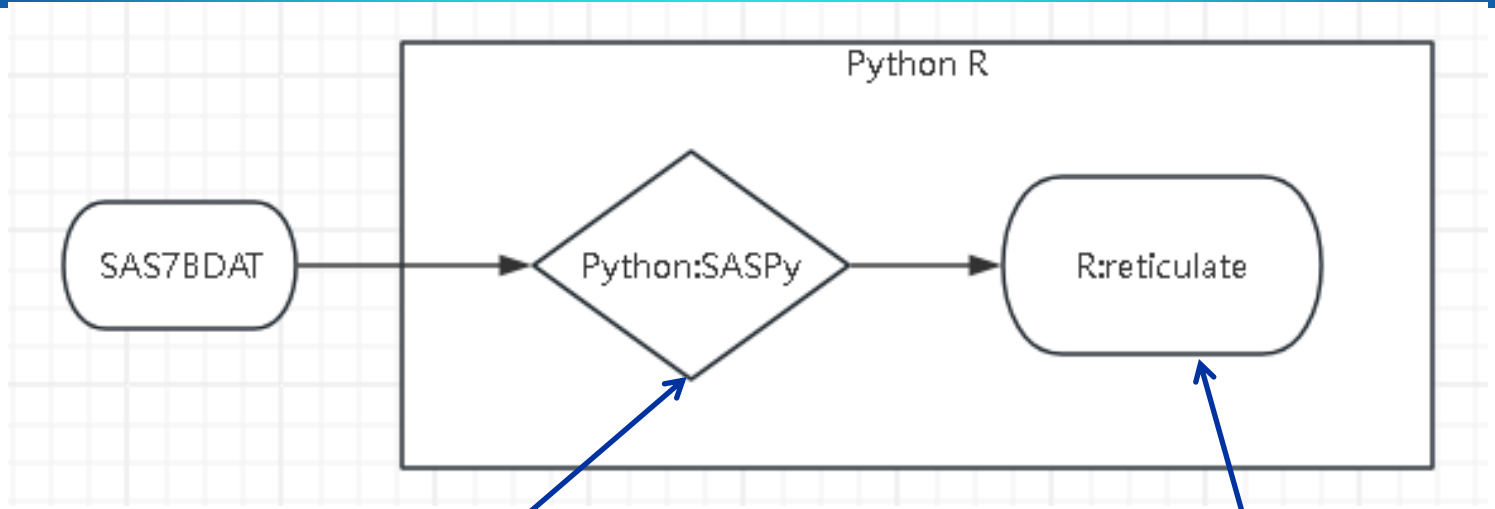
Challenges when Converting Between R and SAS



Methods for Exchanging Data between SAS and R

Methods	Good	Bad	Code or Packages
CSV\Excel	Simple and universal format	Cannot preserve variable labels and formats in SAS	SAS: proc export R:openxlsx
JSON (JavaScript Object Notation)	Suitable for exchanging structured and unstructured data, especially when dealing with nested structured data	Cannot preserve variable labels and formats in SAS	SAS:proc json R:jsonlite or rjson

Use SASPy for Data Exchange




```
import saspy
sas = saspy.SASsession()
df = sas.sasdata('mydata', 'mylib').to_df()
```

```
library(reticulate)
py_run_string("df")  # 从Python环境获取数据框
```

R Packages for SAS Integration

- Haven Package

```
library(haven)
data <- read_sas("data.sas7bdat")
```

- SASxport Package

```
library(SASxport)
write.xport(data, file="data.xpt")
```

Using XML or JSON to Preserve Variable Metadata (Labels and Formats)

XML

libname {xx} xmlv2

```
<DataSet>
  <MetaData>
    <Variable>
      <Name>id</Name>
      <Label>Identifier</Label>
      <Format>Integer</Format>
    </Variable>
    <Variable>
      <Name>name</Name>
      <Label>Full Name</Label>
      <Format>String</Format>
    </Variable>
    <Variable>
      <Name>age</Name>
      <Label>Age in Years</Label>
      <Format>Integer</Format>
    </Variable>
  </MetaData>
```

```
  <Data>
    <Row>
      <id>1</id>
      <name>John Doe</name>
      <age>30</age>
    </Row>
    <Row>
      <id>2</id>
      <name>Jane Smith</name>
      <age>25</age>
    </Row>
    <Row>
      <id>3</id>
      <name>Emily Johnson</name>
      <age>40</age>
    </Row>
  </Data>
</DataSet>
```

Using XML or JSON to Preserve Variable Metadata (Labels and Formats)

JSON

```
filename jsonout temp;

proc json out=jsonout pretty;
  export ae / nosastags;
run;
```

```
{
  "metadata": [
    {
      "name": "id",
      "label": "Identifier",
      "format": "Integer"
    },
    {
      "name": "name",
      "label": "Full Name",
      "format": "String"
    },
    {
      "name": "age",
      "label": "Age in Years",
      "format": "Integer"
    }
  ],
```

```
  "data": [
    {
      "id": 1,
      "name": "John Doe",
      "age": 30
    },
    {
      "id": 2,
      "name": "Jane Smith",
      "age": 25
    },
    {
      "id": 3,
      "name": "Emily Johnson",
      "age": 40
    }
  ]
}
```

Using XML or JSON to Preserve Variable Metadata (Labels and Formats)

Feature	JSON	XML
Simplicity	More concise, smaller file size	Larger files, more complex structure
Readability	Easier for humans to read	Less readable compared to JSON
Compatibility	Supports many programming languages and tools	Widely supported, better for hierarchical data
Metadata Support	Easily stores both data and metadata together	Suitable for storing more complex metadata
Parsing Speed	Faster for parsing large datasets	Slower for parsing large files

Best Practices for Data Exchange

- Ensuring Data Integrity
- Choosing the Right Data Format
- Documenting Data Transfers
- Validating Data Post-Transfer
- Handling Encoding and Locale Issues
- Automating Data Exchange

Case Study: NLP Feature Engineering with SAS

Edit Distance

- COMPGED

```
data sim;
   text1='ATERM ';
   text2='AETERM';
   dis=COMPGED(text1,text2)/100;
   sim=1-dis/max(length(text1),length(text2));
run;
```

VIEWTABLE: Work.Sim

	text1	text2	dis	sim
1	ATERM	AETERM	1	0.8333333333

```
array v1(200) $1 _temporary_;
v=1;
do until(v>length(var));
  v1(v)=substr(var,v,1);
  v=v+1;
end;

array c1(200) $1 _temporary_;
c=1;
do until(c>length(col));
  c1(c)=substr(col,c,1);
  c=c+1;
end;

*edit distance;
array m{201,201} _temporary_;
i=1;j=1;
m{1,1}=0;
do until(i>length(var));
  m(i+1,1)=i;
  i=i+1;
end;
do until(j>length(col));
  m(1,j+1)=j;
  j=j+1;
end;
p=2;q=2;
do until(p>length(var)+1);
  do until(q>length(col)+1);
    if v1(p-1)=c1(q-1) then flag=0;
    else flag=1;
    m{p,q}=min(m{p-1,q}+1,m{p,q-1}+1,m{p-1,q-1}+flag);
    q=q+1;
  end;
  p=p+1;
  q=2;
end;
distant=m{length(var)+1,length(col)+1};
similar=1-distant/max(length(var),length(col));
```

Case Study: NLP Feature Engineering with SAS

Cosine similarity

1、Create a public pool

```
data test;
     length col1 col2 $100;
     col1='APPLE';col2='APPPPPLE';output;
run;

data temp;
     set test;
     length col $200;
     col=cats(col1,col2);
     array word[*] $1 word1-word200 ;
     i=1;
     do while (col^="");
        word[i]=substr(col,1,1);
        col=compress(substr(col,2),word[i]);
        i=i+1;
     end;
```

2、Generate word vectors

```
u=1;
l=1;
array cc(200) c1-c200;
do until(word(u)='' and word(u+1)='');
  cc(u)=0;
  do until(l>length(col1));
    if word(u)=substr(col1,l,1) then cc(u)=cc(u)+1;
    l=l+1;
  end;
  l=1;
  u=u+1;
end;
```

3、Calculate cosine value

```
x=1;
do until(cc(x)=.);
  cosf=cosf+cc(x)*dd(x);
  cosz1=cosz1+cc(x)*cc(x);
  cosz2=cosz2+dd(x)*dd(x);
  x=x+1;
end;
cos=round(cosf/(sqrt(cosz1)*sqrt(cosz2)),0.00000001);
```

Conclusion

Best Practices	Future Directions
Ensure Data Integrity	Explore more seamless automation between tools
Verify the accuracy of data and metadata during transfers, ensuring that no information (like variable labels, formats) is lost or altered.	Implement more sophisticated automation tools for faster and more accurate data exchanges.
Use the Right Tools	Develop Richer NLP and Machine Learning Ecosystems in R or SAS
Tools like Haven streamline the exchange process by allowing direct reads and writes between SAS and R.	Focus on building stronger NLP and machine learning capabilities directly within R or SAS, reducing the need for frequent data transfers.
Automation for Efficiency	Investigate Data Format Standardization
Automate the transfer processes with scripts to save time and reduce manual errors.	Create standardized processes across teams for formats like JSON or XML to retain data formats and encodings.
Address Metadata and Encoding Issues	Integrate Machine Learning for Data Handling
Use XML, JSON, or other methods to preserve variable labels, formats, and encoding to ensure consistency.	Automate checks on encoding or format integrity using ML techniques for detecting data issues during exchange.

Q&A

The Global Healthcare Data Science Community

Contact Channels

UK +44 1843 609600
office@phuse.global
phuse.global

Social Media

@phusetwitta
/phusebook
/phusetube
/company/phuse