

APL64: Using RNET

Contents

Summary	3
 RNET	3
Notes On Examples	3
Creating the R instance in APL64	3
R Packages.....	4
 RNET Syntax.....	4
 RNET Actions	4
CreateDataFrame Actions	5
Cdf: Create DataFrame From Data Matrix	5
CdfNm: Create DataFrame From Nested Matrix.....	10
Set DataFrame Row or Column Names Separately.....	12
Set DataFrame Row or Column Names with Unicode Glyphs	13
Evaluate	14
Syntax:  RNET 'Evaluate' textToEvaluate [subValue1] [subValue2]...	14
Evaluate with an APL64 multi-line string script	15
Value Substitution into the Evaluate Expression	16
Transfer Evaluation Control between R and APL64 (APL: prefix).....	17
GetSymbol	18
Syntax: aplValue $\leftarrow$  RNET 'GetSymbol' symbolName [subValue1] [subValue2]...	18
Value Substitution into the  RNET 'Get' action R-expression.....	21
GGPLOT2.....	22
Help	26
InitR	27
Plot	27
Example: Simple Plot.....	28
Example: Include Plot in  wi Form.....	30

SetSymbol.....	31
Creating and Executing R-Scripts	33
More Examples	35
Set and Get Symbols: <code>⎕rNet 'Set'</code> and <code>⎕rNet 'Get'</code>	35
SetSymbol: Create Rank-2 Character Matrix from APL String Data	35
SetSymbol: Create Rank-2 Character Matrix from APL Character Data	36
Set: Create R-Vector from APL Text.....	37
Set: Create Rank-2 R-Matrix using APL Data	37
Set: Create Nested R-List Containing Mixed Data Types.....	38
Set: Create Nexted R-List Containing Mixed Data Types: APL Array of Rank > 2 is Raveled .	38
Evaluate: <code>⎕←⎕RNET 'Eval'</code>	38
Evaluate: Arithmetic Expression Without R-Symbol Assignment.....	38
Evaluate: Create Numeric R-Vector With R-Symbol Assignment.....	39
Evaluate: Create Rank-2 R-Matrix in R-Engine	39
Evaluate: Create Simple R-List Containing Mixed Data Types.....	40
Evaluate: Create an R-factor	40
Creating DataFrames	41
Creating a DataFrame two ways	41
Create R-DataFrame and obtain R-String with R Statements	43
Accessing Dataframe Columns	44
Using R-Engine for <i>ad hoc</i> Calculations.....	47
Check the Data Type of an R symbol	48
Check if an R-object is a vector	49
Creating and Executing an R-Function with One Argument	50
Creating an Executing an R-Function with Multiple Arguments	50
T-Test.....	51
Plot	52
R-Plot of Random Data.....	52
R-Plot of Beta Distribution	55

Statistical Values	57
Anova Model	59
Getting Information about the R Installation under Program Control.....	60
R may not throw an Exception	64
Resources	65
What is R?.....	65
The R-Project	65
Installing R	65
Compatibility with APL+Win.....	69
R-Tutorials.....	69
R.Net.....	69

Summary

□RNET

The APL64 system function □RNET is a .Net interface to the R statistical and graphics toolkit. □RNET is cross-platform, and uses the x64 R-engine for greater capacity and performance.

There is only one instance of the R engine when using □RNET in an instance of APL64. The R-Engine is automatically disposed when the APL64 instance ends. □RNET is available in the developer and runtime versions of APL64.

Suggestions for installing R are available [here](#).

Notes On Examples

Creating the R instance in APL64

The first time the □RNET system function is used in an APL64 instance, the □RNET 'InitR' action must be used to specify the installation folder of the x64 R version installed on the workstation.

The examples may not illustrate the initial, required use of the □RNET 'InitR' action. For examples which illustrate the □RNET 'InitR' action, the installationPathOfR may need to be modified to reflect the installation of R on the workstation.

R Packages

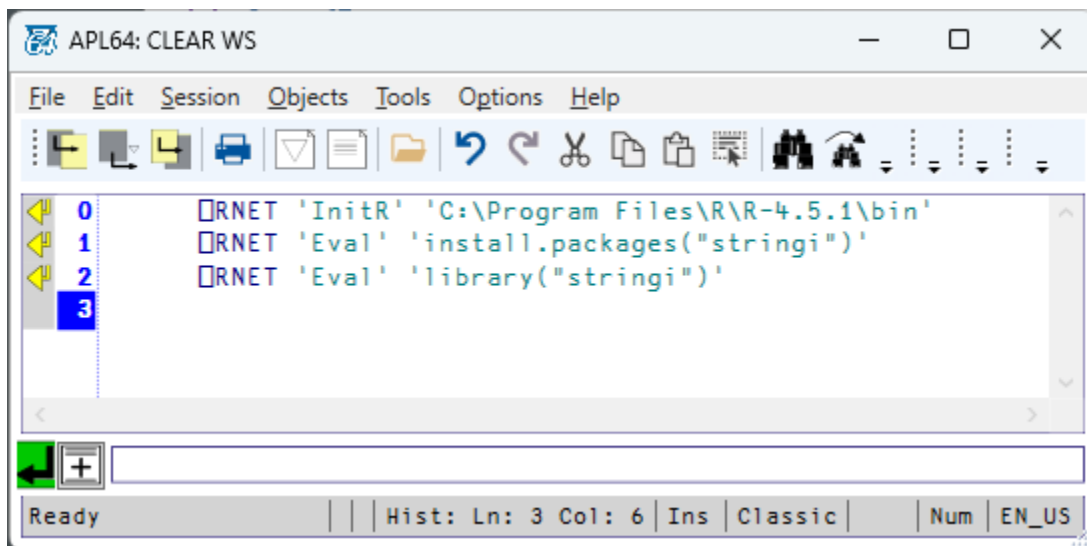
R packages are extensions to the R toolkit. R packages are installed using the R expression `install.Packages(packageName)`. There are additional arguments to [install.Packages\(\)](#) to specify the web server source. Using `install.Packages()` may present a list of web servers where the R-package is available. Installed R packages are loaded using R expression [library\(packageName\)](#).

Some examples require that specific R packages are installed and loaded, but the R expressions to install and load those packages may not be included in the example source code.

It may be necessary to run APL64 with elevated credentials, to successfully install an R-package.

Example:

```
☐ RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin\x64'  
☐ RNET 'Eval' 'install.packages("stringi")'  
☐ RNET 'Eval' 'library("stringi")'
```



☐ RNET Syntax

☐ RNET is a monadic system function.

[result] ← ☐ RNET action [actionArgumentsArray]

☐ RNET Actions

The ☐ RNET action text is case-insensitive.

R symbols are case sensitive

CreateDataFrame Actions

An R-Dataframe is an R data structure which is analogous to an APL64 rank-2 matrix.

- The `□RNET 'Cdf'` action will create an R-dataframe using the supplied APL64 matrix. The APL64 programmer can optionally supply row and column names. If row and column names are not supplied, they will be created by this action.
- The `□RNET 'CdfNm'` action will create an R-dataframe using the supplied APL64 matrix. The row and column names of the dataframe are assumed to be the first row and first column of the supplied (nested) APL64.

The `□RNET` actions, `Cdf` and `CdfNm` to create a dataframe have these arguments:

- `dfName` is a text vector or string scalar which will be the name of the resulting dataframe within the R environment.
- `dataMatrix` is an APL64 rank-2 array.
- `checkRows` is a Boolean scalar. If 1, R checks for consistency in the length of the rows and their names, ensuring all columns have the same number of elements.
- `checkCols` is a Boolean scalar. If 1, R checks the names of the variables (columns) to ensure they are syntactically valid and unique.
- `stringAsFactors` is a Boolean scalar. If 1, text data in the input will be converted to R-factors.

For any column of the APL64 data matrix used to create the dataframe, the elements of that column must have the same R data type.

All APL64 text data type elements are converted to the .Net string data type which is compatible with the R text data type.

When the dataframe is created, each column of the APL64 source data matrix is independently checked for its data type to determine Boolean data type columns, if any.

APL and .Net store arrays in row-major order. R stores arrays in column-major order. This behavior of R may cause a display of dataframe to appear transposed compared to the source APL or .Net array. The rows of an APL or .Net array are considered the columns of an R array.

Cdf: Create DataFrame From Data Matrix

Syntax:

```
□RNET 'Cdf' dfName dataMat checkRows checkCols stringAsFactors [colNames] [rowNames]
```

`dataMat` is an APL rank-2 matrix. The element values in this matrix are not used to create row or column names.

colNames is an optional argument. If provided, it is an array of text-valued elements. If not provided, default column names will be created: Col1, ...

rowNames is an optional argument. If provided, it is an array of text-valued elements. If not provided, default column names will be provided: Row1, ...

Example:

```
s<- 'Prod1' 'lb' 5.44 23.2 1
s<-s[ ]over 'Prod2' 'oz' 3.42 0.0 0
s<-s[ ]over «Prod3» 'kg' 2.01 2.2 0
s<-s[ ]over 'Prod4' 'ml' 13.99 10 1
s<-s[ ]over 'Prod5' 'ctn' 30.01 0 0
s
[ ]RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin\x64'
[ ]RNET 'Cdf' 'df1' s 1 1 0
[ ]RNET 'Get' 'df1'
[ ]RNET 'Eval' 'p1=print(df1)'
[ ]RNET 'Get' 'p1'
[ ]RNET 'eval' 'p2=print(sapply(df1, class))'
[ ]RNET 'Get' 'p2'
```

```
APL64: CLEAR WS  
File Edit Session Objects Tools Options Help  
  
0 s←'Prod1' 'lb' 5.44 23.2 1  
1 s←s⊖over 'Prod2' 'oz' 3.42 0.0 0  
2 s←s⊖over «Prod3» 'kg' 2.01 2.2 0  
3 s←s⊖over 'Prod4' 'ml' 13.99 10 1  
4 s←s⊖over 'Prod5' 'ctn' 30.01 0 0  
5 s  
6 Prod1 lb 5.44 23.2 1  
7 Prod2 oz 3.42 0.0 0  
8 Prod3 kg 2.01 2.2 0  
9 Prod4 ml 13.99 10.0 1  
10 Prod5 ctn 30.01 0.0 0  
11 ⎕RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin'  
12 ⎕RNET 'Cdf' 'df1' s 1 1 0  
13 ⎕RNET 'Get' 'df1'  
14 Row1 Row2 Row3 Row4 Row5  
15 Col1 Prod1 Prod2 Prod3 Prod4 Prod5  
16 Col2 lb oz kg ml ctn  
17 Col3 5.44 3.42 2.01 13.99 30.01  
18 Col4 23.20 0.00 2.20 10.00 0.00  
19 Col5 1.00 0.00 0.00 1.00 0.00  
20 ⎕RNET 'Eval' 'p1=print(df1)'  
21 ⎕RNET 'Get' 'p1'  
22 Row1 Row2 Row3 Row4 Row5  
23 Col1 Prod1 Prod2 Prod3 Prod4 Prod5  
24 Col2 lb oz kg ml ctn  
25 Col3 5.44 3.42 2.01 13.99 30.01  
26 Col4 23.20 0.00 2.20 10.00 0.00  
27 Col5 1.00 0.00 0.00 1.00 0.00  
28 ⎕RNET 'eval' 'p2=print(sapply(df1, class))'  
29 ⎕RNET 'Get' 'p2'  
30 character character numeric numeric logical  
31 |
```

Example:

```
s<-Prod1'lb' 5.44 23.2 1
s<-s over 'Prod2' 'oz' 3.42 0.0 0
s<-s over «Prod3» 'kg' 2.01 2.2 0
s<-s over 'Prod4' 'ml' 13.99 10 1
s<-s over 'Prod5' 'ctn' 30.01 0 0
```

S

```
colNames←'Product Type' 'Unit' 'Unit Cost' 'Bulk Disc Pct' 'Available'
```

```
□ RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin\x64'
```

```
□ RNET 'Cdf' 'df1' s 1 1 0 colNames
```

```
□ RNET 'Get' 'df1'
```

```
□ RNET 'Eval' 'p1=print(df1)'
```

```
□ RNET 'Get' 'p1'
```

```
□ RNET 'eval' 'p2=print(sapply(df1, class))'
```

```
□ RNET 'get' 'p2'
```

```
0 s←'Prod1' '1b' 5.44 23.2 1
1 s←s∘over 'Prod2' 'oz' 3.42 0.0 0
2 s←s∘over «Prod3» 'kg' 2.01 2.2 0
3 s←s∘over 'Prod4' 'ml' 13.99 10.0 1
4 s←s∘over 'Prod5' 'ctn' 30.01 0.0 0
5 s
6 Prod1 1b 5.44 23.2 1
7 Prod2 oz 3.42 0.0 0
8 Prod3 kg 2.01 2.2 0
9 Prod4 ml 13.99 10.0 1
10 Prod5 ctn 30.01 0.0 0
11 colNames←'Product Type' 'Unit' 'Unit Cost' 'Bulk Disc Pct' 'Available'
12 □ RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin\x64'
13 □ RNET 'Cdf' 'df1' s 1 1 0 colNames
14 □ RNET 'Get' 'df1'
15 Row1 Row2 Row3 Row4 Row5
16 Product.Type Prod1 Prod2 Prod3 Prod4 Prod5
17 Unit 1b oz kg ml ctn
18 Unit.Cost 5.44 3.42 2.01 13.99 30.01
19 Bulk.Disc.Pct 23.20 0.00 2.20 10.00 0.00
20 Available 1.00 0.00 0.00 1.00 0.00
21 □ RNET 'Eval' 'p1=print(df1)'
22 □ RNET 'Get' 'p1'
23 Row1 Row2 Row3 Row4 Row5
24 Product.Type Prod1 Prod2 Prod3 Prod4 Prod5
25 Unit 1b oz kg ml ctn
26 Unit.Cost 5.44 3.42 2.01 13.99 30.01
27 Bulk.Disc.Pct 23.20 0.00 2.20 10.00 0.00
28 Available 1.00 0.00 0.00 1.00 0.00
29 □ RNET 'eval' 'p2=print(sapply(df1, class))'
30 □ RNET 'get' 'p2'
31 character character numeric numeric logical
32
```

Example:

```
s<- 'lb' 5.44 23.2 1
s<-s[ ]over 'oz' 3.42 0.0 0
s<-s[ ]over 'kg' 2.01 2.2 0
s<-s[ ]over 'ml' 13.99 10 1
s<-s[ ]over 'ctn' 30.01 0 0
s
colNames<- 'Unit' 'Unit Cost' 'Bulk Disc Pct' 'Available'
rowNames<- 'Prod1' 'Prod2' «Prod3» 'Prod4' 'Prod5'
[ ] RNET 'Cdf' 'df1' s 1 1 0 colNames rowNames
[ ] RNET 'Get' 'df1'
[ ] RNET 'Eval' 'p1=print(df1)'
[ ] RNET 'Get' 'p1'
[ ] RNET 'Eval' 'p2=print(sapply(df1, class))'
p[ ]<-[ ] RNET 'Get' 'p2'
```

```

0 s←'lb' 5.44 23.2 1
1 s←s⊂over 'oz' 3.42 0.0 0
2 s←s⊂over 'kg' 2.01 2.2 0
3 s←s⊂over 'ml' 13.99 10 1
4 s←s⊂over 'ctn' 30.01 0 0
5 s
6 lb 5.44 23.2 1
7 oz 3.42 0.0 0
8 kg 2.01 2.2 0
9 ml 13.99 10.0 1
10 ctn 30.01 0.0 0
11 colNames←'Unit' 'Unit Cost' 'Bulk Disc Pct' 'Available'
12 rowNames←'Prod1' 'Prod2' «Prod3» 'Prod4' 'Prod5'
13 ⍋RNET 'Cdf' 'df1' s 1 1 0 colNames rowNames
14 ⍋RNET 'Get' 'df1'
15      Prod1 Prod2 Prod3 Prod4 Prod5
16 Unit      lb   oz   kg   ml   ctn
17 Unit.Cost  5.44  3.42  2.01 13.99 30.01
18 Bulk.Disc.Pct 23.20 0.00 2.20 10.00 0.00
19 Available   1.00  0.00 0.00 1.00 0.00
20 ⍋RNET 'Eval' 'p1=print(df1)'
21 ⍋RNET 'Get' 'p1'
22      Prod1 Prod2 Prod3 Prod4 Prod5
23 Unit      lb   oz   kg   ml   ctn
24 Unit.Cost  5.44  3.42  2.01 13.99 30.01
25 Bulk.Disc.Pct 23.20 0.00 2.20 10.00 0.00
26 Available   1.00  0.00 0.00 1.00 0.00
27 ⍋RNET 'Eval' 'p2=print(sapply(df1, class))'
28 p←⍋RNET 'Get' 'p2'
29 character numeric numeric logical
30 4
31 |

```

Ready | Hist: Ln: 31 Col: 6 | Ins | Classic | Cap | Num | EN_US

CdfNm: Create DataFrame From Nested Matrix

Syntax: `⍋RNET 'CDF' dfName nestedMat checkRows checkCols stringAsFactors`

dfName is a text vector or string scalar

nestedMat is an APL rank-2 matrix. The first row and first columns of nestedMatrix will become the row and column names in the resulting dataframe. The 'inner' rows and columns of

nestedMatrix will become the data elements of the resulting dataframe determine Boolean data type columns, if any.

checkRows is a Boolean scalar. If 1, R checks for consistency in the length of the rows and their names, ensuring all columns have the same number of elements.

checkCols is a Boolean scalar. If 1, R checks the names of the variables (columns) to ensure they are syntactically valid and unique.

stringAsFactors is a Boolean scalar. If 1, text data in the input will be converted to R-factors.

Example:

```
s<-'ProductType' 'Unit' 'Unit Cost' 'BulkDiscPct' 'Discontinued'
s<-s[ ]over 'Prod1' 'lb' 5.44 23.2 1
s<-s[ ]over 'Prod2' 'oz' 3.42 0.0 0
s<-s[ ]over «Prod3» 'kg' 2.01 2.2 0
s<-s[ ]over 'Prod4' 'ml' 13.99 10 1
s<-s[ ]over 'Prod5' 'ctn' 30.01 0 0
s
[ ]RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin\x64'
[ ]RNET 'CDFNM' 'df1' s 1 1 0
ρ[ ]←[ ]RNET 'Get' 'df1'
[ ]RNET 'Eval' 'p1=print(df1)'
ρ[ ]←[ ]RNET 'Get' 'p1'
[ ]RNET 'eval' 'p2=print(sapply(df1, class))'
ρ[ ]←[ ]RNET 'Get' 'p2'
```


Y←2 3p16

☐ RNET 'Cdf' 'df' Y 1 1 0

'Default row and column names:'

☐ RNET 'Get' 'df'

☐ RNET 'Set' 'colNames' ('C1' 'C2' 'C3')

☐ RNET 'Eval' 'colnames(df) = colNames'

☐ RNET 'Set' 'rowNames' ('R1' 'R2')

☐ RNET 'Eval' 'rownames(df) = rowNames'

'Separately set row and column names:'

☐ RNET 'Get' 'df'

```
0      SetRowColNames
1      Default row and column names:
2      Row1 Row2
3      Col1  1  4
4      Col2  2  5
5      Col3  3  6
6      Separately set row and column names:
7      R1 R2
8      C1  1  4
9      C2  2  5
10     C3  3  6
11
```

Set DataFrame Row or Column Names with Unicode Glyphs

UnicodeChars

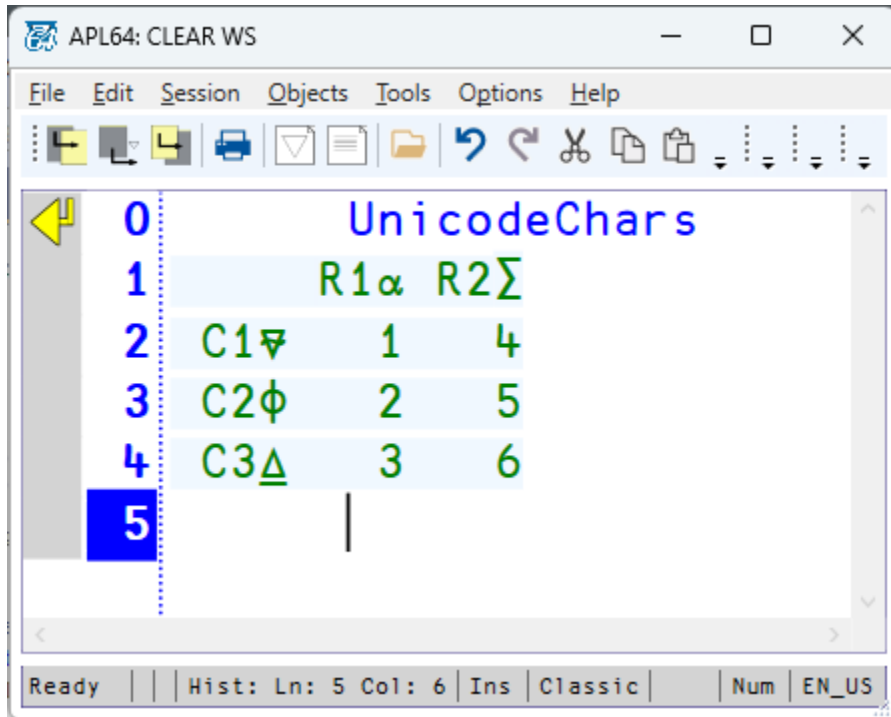
☐ RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin\x64'

Y←2 3p16

☐ RNET 'Cdf' 'df' Y 1 1 0

☐ RNET 'Eval' 'colnames(df) = c("C1▽", "C2⊕", "C3△")'

☐ RNET 'Eval' 'rownames(df) = c("R1α", "R2\u00a2")'
☒ U2211: Σ
☐ RNET 'Get' 'df'



Evaluate

Synonyms: Eval, Ev

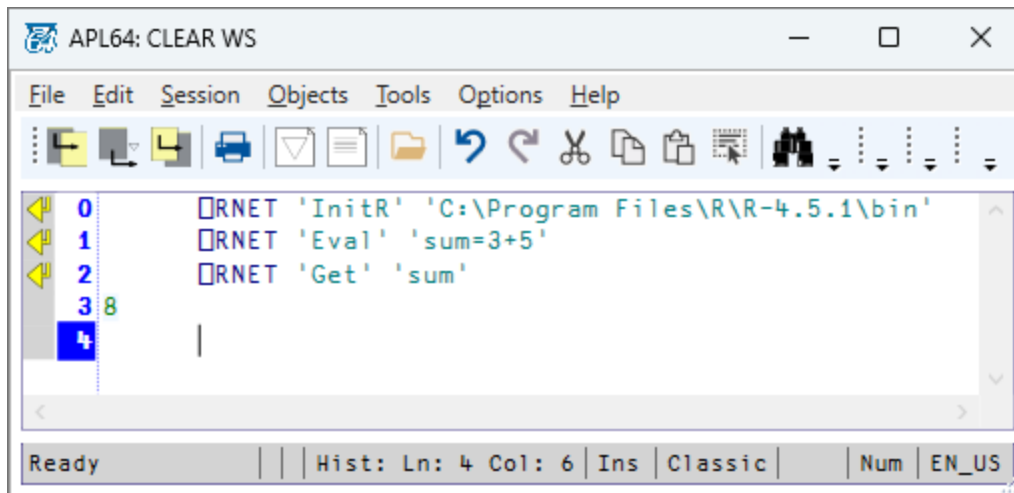
The Evaluate action will cause the R-engine to evaluate the 'textToEvaluate' within the R environment. The 'textToEvaluate' can be an APL64 multi-line string script.

Evaluate returns no result. To obtain the result of an R-expression, assign it to an R-variable, and use the ☐ RNET 'Get' action to obtain the result value in APL64. Some R data types have no representation in APL.

Syntax: ☐ RNET 'Evaluate' textToEvaluate [subValue1] [subValue2]...

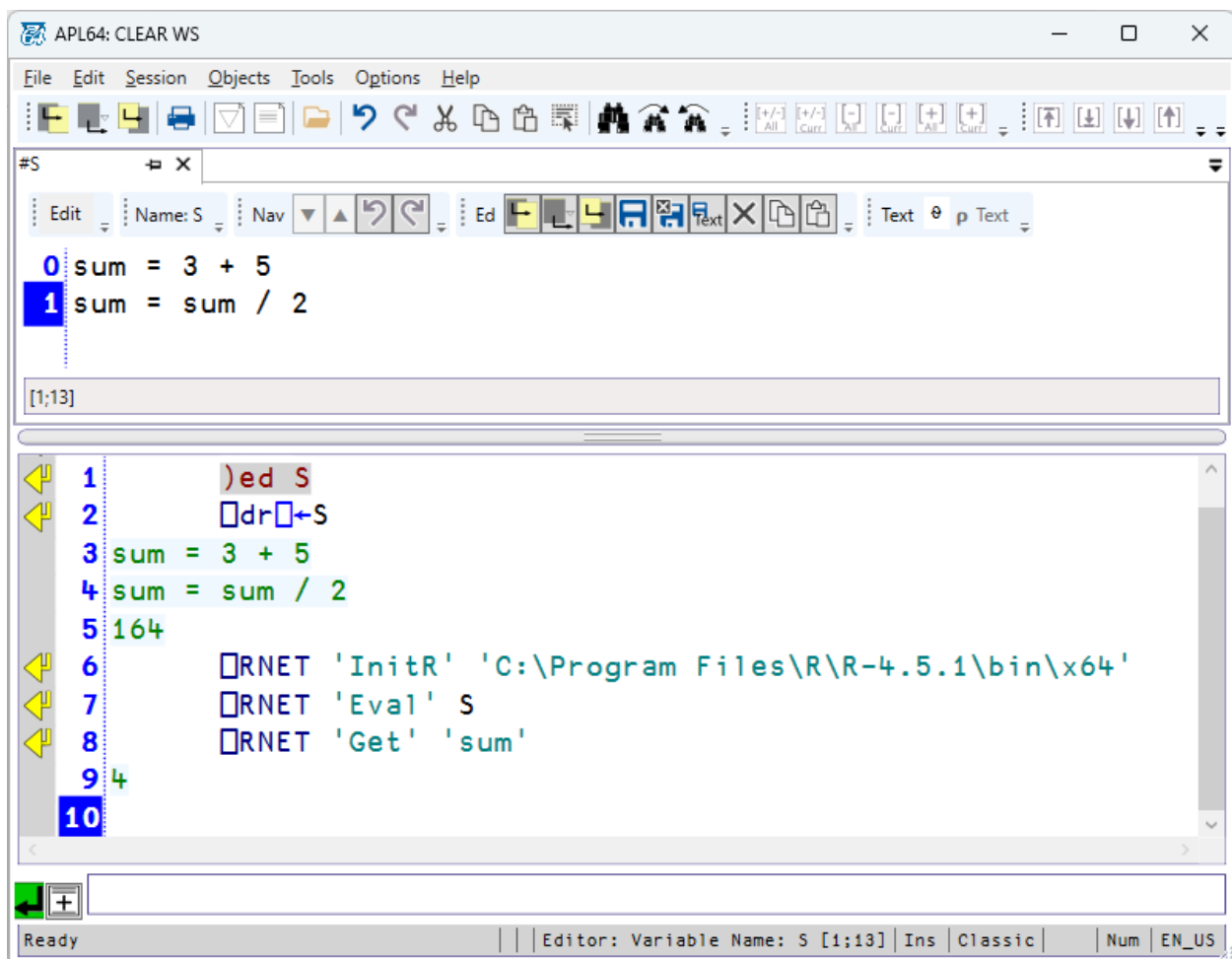
Example:

☐ RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin\x64'
☐ RNET 'Eval' 'sum=3+5'
☐ RNET 'Get' 'sum'

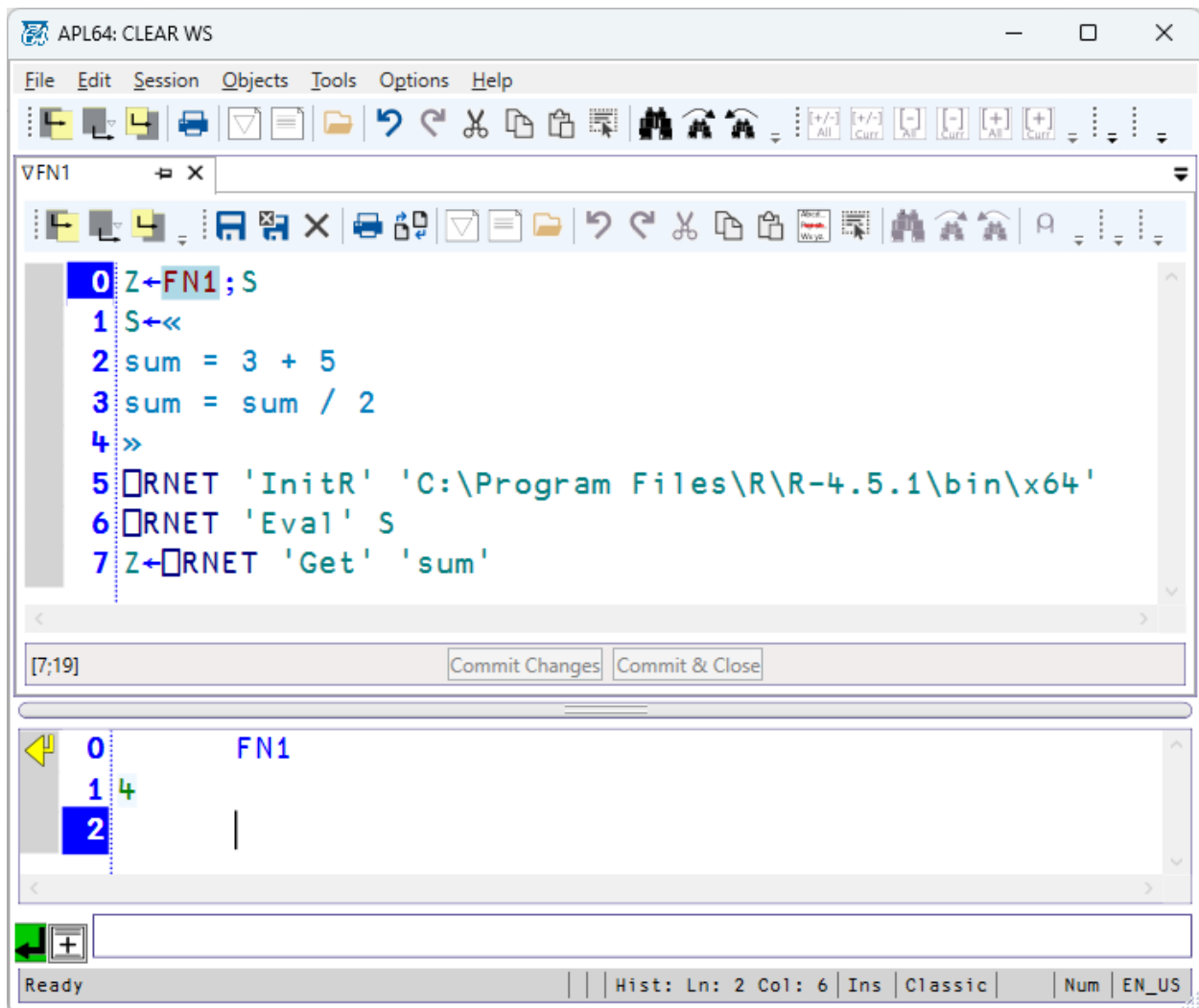


Evaluate with an APL64 multi-line string script

Example: Script created using the eAPL64 variable editor and used *ad hoc*



Example: Script created and used within an APL64 programmer-designed function

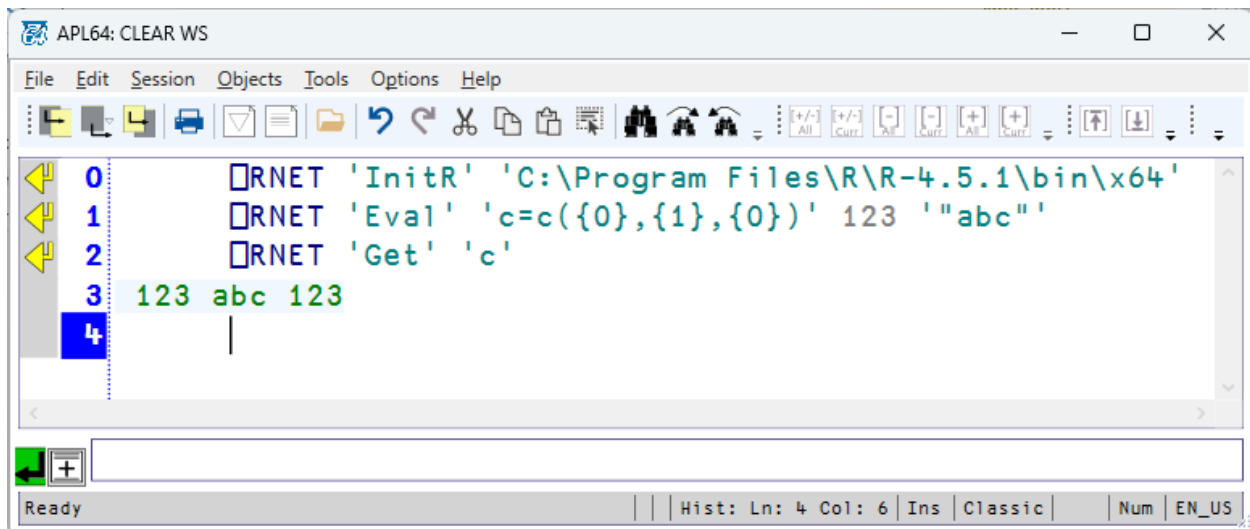


Value Substitution into the Evaluate Expression

APL64 values may be substituted into the evaluation expression. The substitution location in the Evaluation expression is indicated by {n}, where n is the index, origin zero, of substitution value. Substitution values may be used more than once. All substitutions are performed prior to evaluation of the evaluation expression.

Example

☐	RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin\x64'
☐	RNET 'Eval' 'c=c({0},{1},{0})' 123 ""abc""
☐	RNET 'Get' 'c'



Transfer Evaluation Control between R and APL64 (APL: prefix)

Within an APL64 multi-line script, it is possible to temporarily transfer evaluation control from the R server environment to the APL64 client environment when that script is evaluated. To evaluate a script line within the APL64 client environment, prefix the APL64 executable statement on that line with 'APL:'.

Example: Using a multi-line string script

```
Z←GetRoots input;S
S←««
APL: □RNET 'Set' 'input' input
sqRt = input^0.5
cbRt = input^(1/3)
APL: Z←(□RNET 'Get' 'sqRt') (□RNET 'Get' 'cbRt')
»»
□RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin\x64'
□RNET 'Eval' S
```

The screenshot shows the APL64: CLEAR WS window. The top menu bar includes File, Edit, Session, Objects, Tools, Options, and Help. Below the menu is a toolbar with various icons. The main window displays a script for the GetRoots function:

```

0 Z←GetRoots input;S
1 S←«««
2 APL: □RNET 'Set' 'input' input
3 sqRt = input^0.5
4 cbRt = input^(1/3)
5 APL: Z+((□RNET 'Get' 'sqRt') (□RNET 'Get' 'cbRt'))
6 »»»
7 □RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin\x64'
8 □RNET 'Eval' S

```

Below the script, the execution results are shown:

```

0      □dr□←GetRoots 2
1  1.414213562  1.25992105
2  326
3

```

The status bar at the bottom indicates "Ready" and "Hist: Ln: 3 Col: 6 Ins Classic Num EN_US".

GetSymbol

Synonyms: GetSym, Get

The □RNET GetSymbol action fetches an APL64 representation from the R environment of the data associated with an R symbolName. symbolName can be an R-expression, which will be evaluated for its result. Some R data types have no representation in APL.

Syntax: apIValue ← □RNET 'GetSymbol' symbolName [subValue1] [subValue2]...

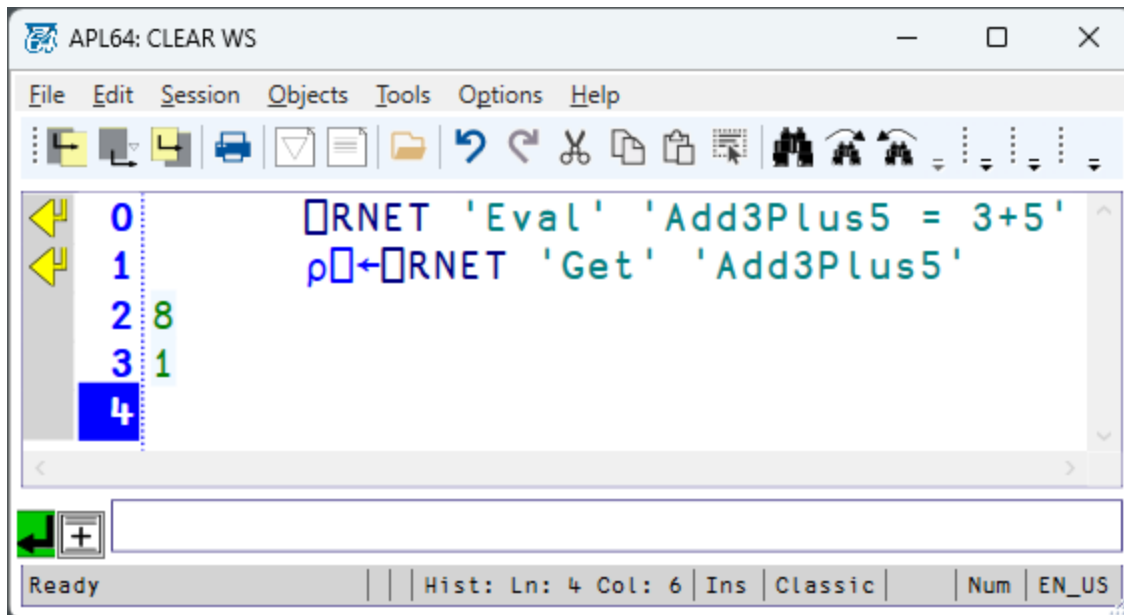
Example:

In this example Add3Plus5 is an R-symbol name.

```

□RNET 'Eval' 'Add3Plus5 = 3+5'
ρ□←□RNET 'Get' 'Add3Plus5'

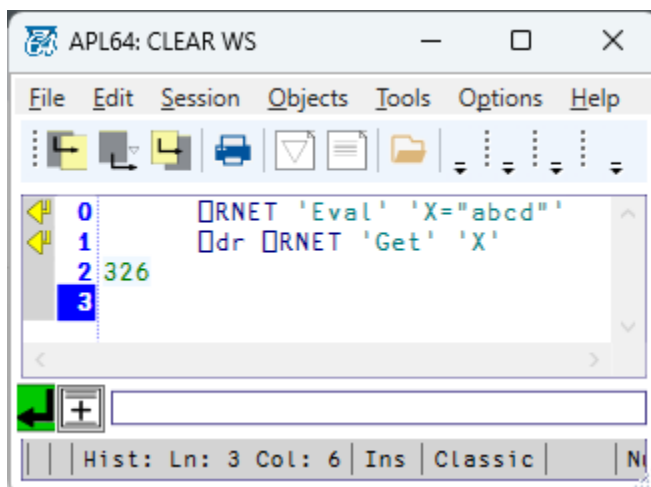
```



Example:

In this example X is an R-symbol name.

```
⎕RNET 'Eval' 'X="abcd"'
⎕dr ⎕RNET 'Get' 'X'
```



Example:

R has one text data type. When APL64 data is 'round tripped' through the R instance, the result may have a different APL64 datatype and shape. The appropriate R environment must be established for proper rendering of some Unicode glyphs.

```
⎕RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin\x64'
```

```

☐ RNET 'Set' 'cV' 'abcd'
☐ dr☐ ← cV ← ☐ RNET 'Get' 'cV'
pcV
☐ RNET 'Set' 'uV' 'AΣBπC'
☐ dr☐ ← uV ← ☐ RNET 'Get' 'uV'
puV
☐ RNET 'Set' 'tV' 'Abc Def Ghi'
☐ dr☐ ← tV ← ☐ RNET 'Get' 'tV'
ptV

```

The screenshot shows the APL64: CLEAR WS application window. The menu bar includes File, Edit, Session, Objects, Tools, Options, and Help. The toolbar contains various icons for file operations, editing, and execution. The main workspace displays a sequence of commands and their results, numbered 0 through 19. The commands are as follows:

```

0 ☐ RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin\x64'
1 ☐ RNET 'Set' 'cV' 'abcd'
2 ☐ dr☐ ← cV ← ☐ RNET 'Get' 'cV'
3 abcd
4 326
5 pcV
6 1
7 ☐ RNET 'Set' 'uV' 'AΣBπC'
8 ☐ dr☐ ← uV ← ☐ RNET 'Get' 'uV'
9 A?BpC
10 326
11 puV
12 1
13 ☐ RNET 'Set' 'tV' 'Abc Def Ghi'
14 ☐ dr☐ ← tV ← ☐ RNET 'Get' 'tV'
15 Abc Def Ghi
16 326
17 ptV
18 1
19

```

The status bar at the bottom indicates 'Ready' and provides navigation information: 'Hist: Ln: 19 Col: 6 Ins Classic Num EN_US'.

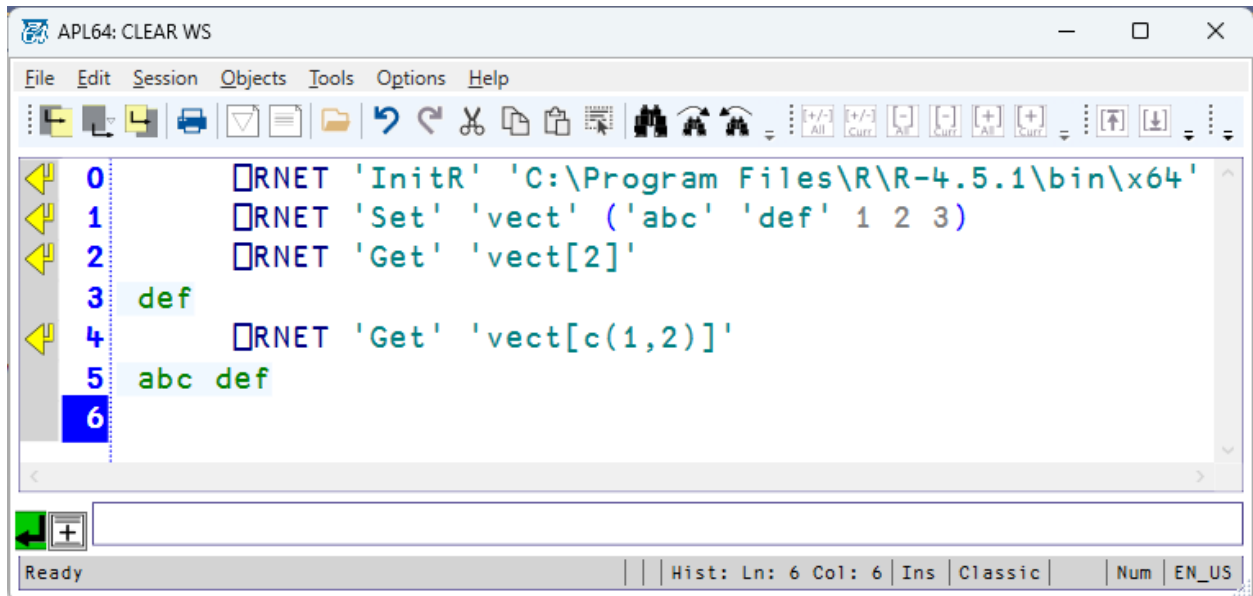
Example:

This example illustrates the ☐ RNET 'Get' action with R-expressions.

```

☐ RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin\x64'
☐ RNET 'Set' 'vect' ('abc' 'def' 1 2 3)
☐ RNET 'Get' 'vect[2]'
☐ RNET 'Get' 'vect[c(1,2)]'

```



Value Substitution into the ☐ RNET 'Get' action R-expression

APL64 values may be substituted into the R-expression used in the ☐ RNET 'Get' action. The substitution location in the R-expression is indicated by {n}, where n is the index, origin zero, of substitution value. Substitution values may be used more than once. If the R-expression emits more than one value, only the last emitted value is returned by the ☐ RNET 'Get' action.

```

☐ RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin\x64'
☐ RNET 'Set' 'vect' ("abc" "def" «ghij» 1 2 3.456)
☐ RNET 'Get' 'vect'
☐ RNET 'Get' 'vect[c(2,6)]'
☐ RNET 'Get' 'vect[c({0},{1})]' 2 6
☐ RNET 'Set' 'intV' (t10)
S←'vect[c({0},{1})]',☐ tcnl,☐ tclf,'intV'
ρ☐ ←☐ RNET 'Get' S 2 3
S←'intV',☐ tcnl,☐ tclf,'vect[c({0},{1})]'
ρ☐ ←☐ RNET 'Get' S 2 3

```

```

0  □RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin\x64'
1  □RNET 'Set' 'vect' ('abc' 'def' «ghij» 1 2 3.456)
2  □RNET 'Get' 'vect'
3  abc def ghij 1 2 3.456
4  □RNET 'Get' 'vect[c(2,6)]'
5  def 3.456
6  □RNET 'Get' 'vect[c({0},{1})]' 2 6
7  def 3.456
8  □RNET 'Set' 'intV' (10)
9  S←'vect[c({0},{1})]',□tcln1,□tclf,'intV'
10 ρ□←RNET 'Get' S 2 3
11 1 2 3 4 5 6 7 8 9 10
12 10
13 S←'intV',□tcln1,□tclf,'vect[c({0},{1})]'
14 ρ□←RNET 'Get' S 2 3
15 def ghij
16 2
17 |

```

GGPLOT2

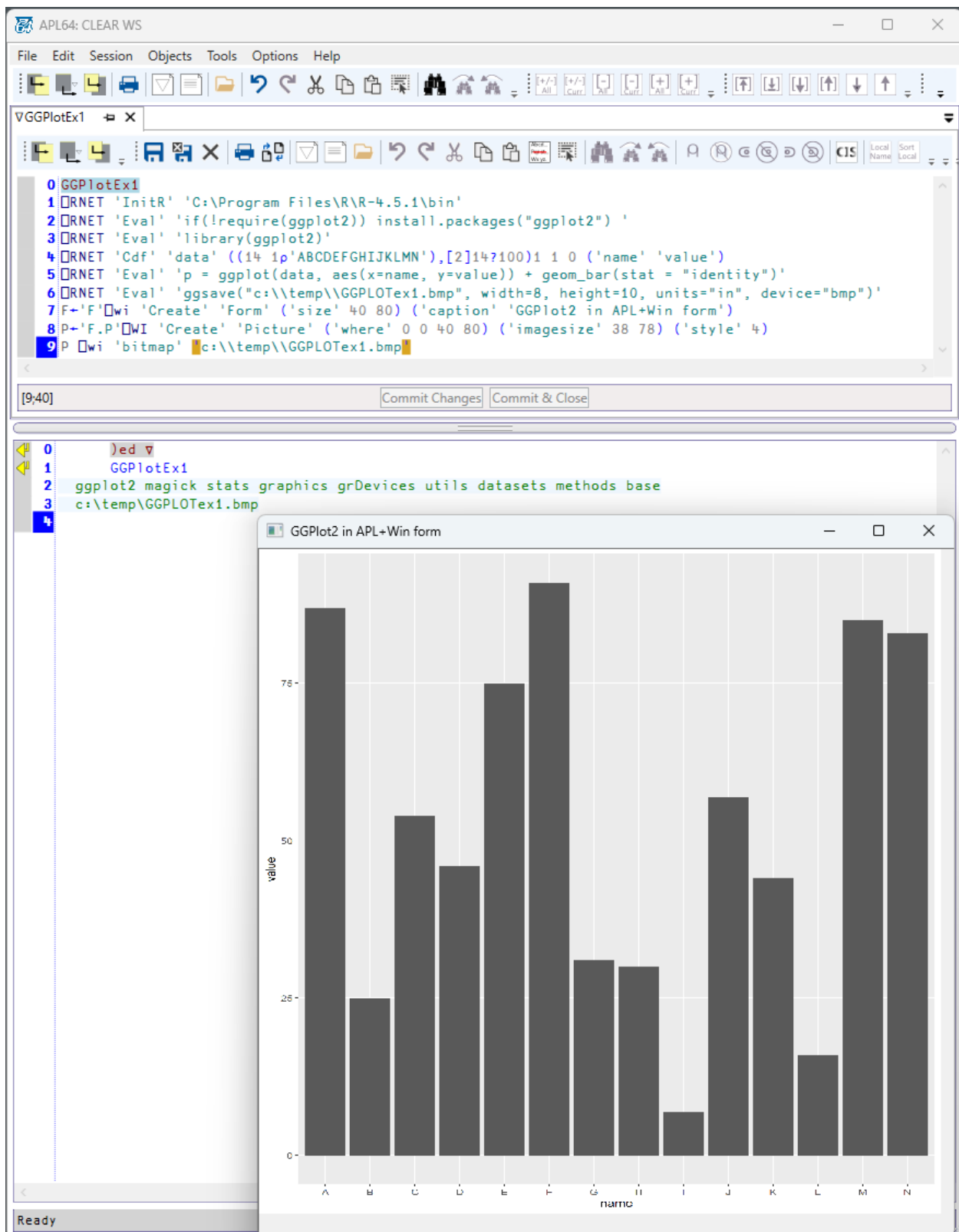
[GGPlot2](#) is so simple to use in □RNET that no □RNET 'GGPlot2' action is necessary. The typical GGPlot2 workflow involves:

- Using □RNET Eval or Set actions to pass APL64 data to the R instance
- If desired, use the □RNET Eval or Set actions to assign a GGPlot2 result to an R variable.
- Using the □RNET Eval action to run the R-function [ggsave](#) to render the GGPlot2 to a file in the desired format and dimensions and file type.
- The file-based result of GGPlot2 can be included in an APL64 □wi Form using the □WI Picture and bitmap features.
- For short R-scripts use □RNET repetitively. For longer, multi-line R-scripts use the APL64 string continuation structure, script←««...»», within a function.

Example: Create this function and run it:

GGPlotEx1

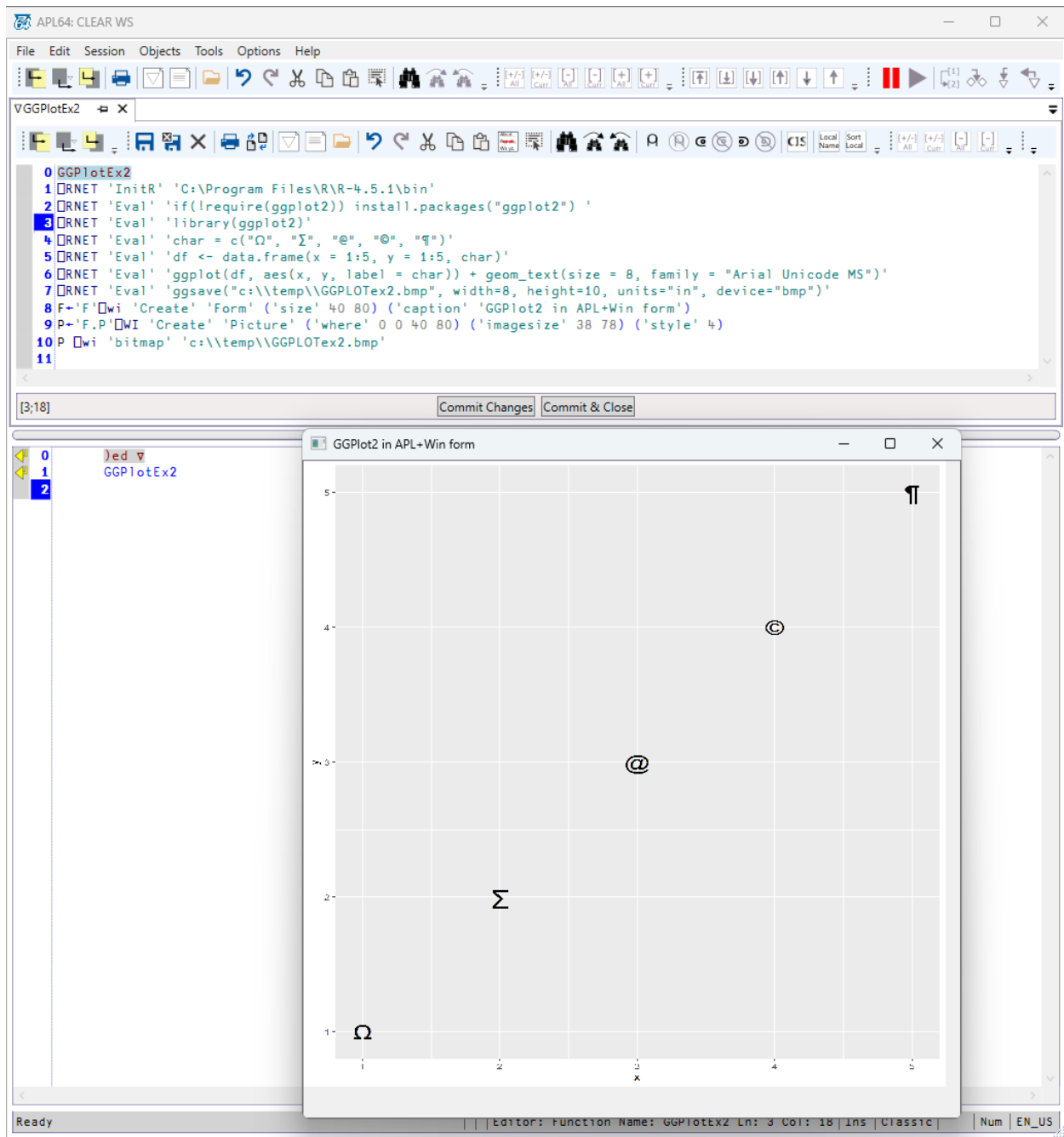
```
☐ RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin\x64'  
☐ RNET 'Eval' 'if(!require(ggplot2)) install.packages("ggplot2") '  
☐ RNET 'Eval' 'library(ggplot2)'  
☐ RNET 'Cdf' 'data' ((14 1p'ABCDEFGHJKLMN'),[2]14?100)1 1 0 ('name' 'value')  
☐ RNET 'Eval' 'p = ggplot(data, aes(x=name, y=value)) + geom_bar(stat = "identity")'  
☐ RNET 'Eval' 'ggsave("c:\\temp\\GGPLOTex1.bmp", width=8, height=10, units="in",  
device="bmp")'  
F←'F'☐wi 'Create' 'Form' ('size' 40 80) ('caption' 'GGPlot2 in APLNow32 form')  
P←'F.P'☐Wl 'Create' 'Picture' ('where' 0 0 40 80) ('imagesize' 38 78) ('style' 4)  
P ☐wi 'bitmap' 'c:\temp\GGPLOTex1.bmp'
```



Example: Create this function and run it:

GGPlotEx2

```
☐ RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin\x64'  
☐ RNET 'Eval' 'if(!require(ggplot2)) install.packages("ggplot2") '  
☐ RNET 'Eval' 'library(ggplot2)'  
☐ RNET 'Eval' 'char = c("Ω", "Σ", "@", "©", "¶")'  
☐ RNET 'Eval' 'df <- data.frame(x = 1:5, y = 1:5, char)'  
☐ RNET 'Eval' 'ggplot(df, aes(x, y, label = char)) + geom_text(size = 8, family = "Arial Unicode MS")'  
☐ RNET 'Eval' 'ggsave("c:\\temp\\GGPLOTex2.bmp", width=8, height=10, units="in",  
device="bmp")'  
F←'F'☐wi 'Create' 'Form' ('size' 40 80) ('caption' 'GGPlot2 in APLNow32 form')  
P←'F.P'☐Wl 'Create' 'Picture' ('where' 0 0 40 80) ('imagesize' 38 78) ('style' 4)  
P ☐wi 'bitmap' 'c:\temp\GGPLOTex2.bmp'
```

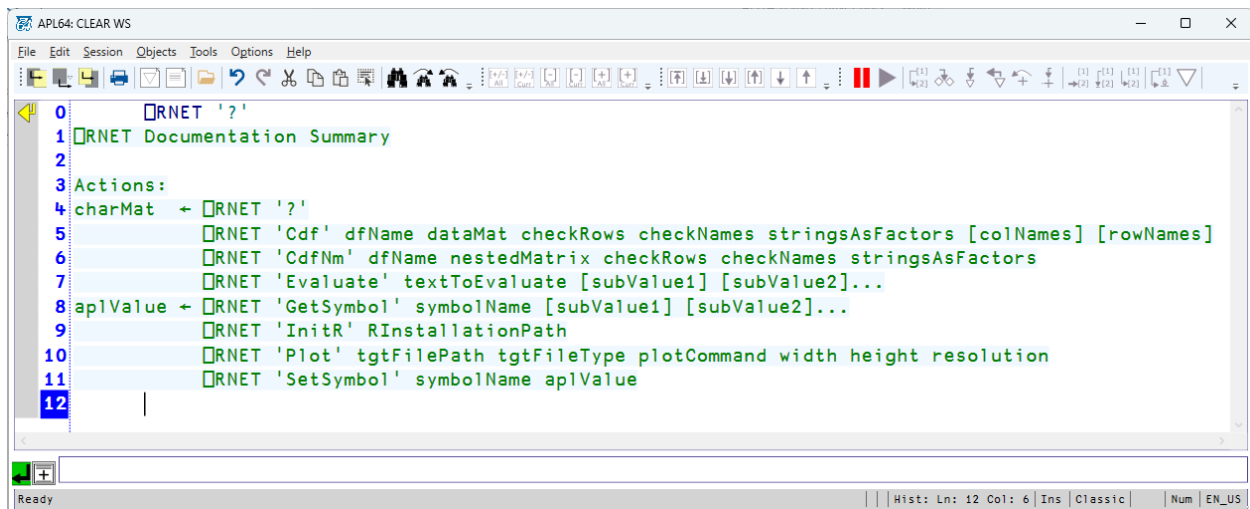


See [here](#) for many more GGLOT2 examples.

Help

Synonym : ?

Syntax: charMat ← ☐ RNET '?'



InitR

Syntax: `□RNET 'InitR' installationPathOfR`

This action creates an R-engine in the APL64 instance. This action must be used once in an APL64 instance before any other `□RNET` actions are used. The `installationPathOfR` value will depend on how R was installed on the target workstation.

Example: For recent installations of R on Windows 11:

```
□RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin\x64'
```

Example: For some installations of R environment variables are set, so that an abbreviated argument to the `InitR` action is acceptable:

```
□RNET 'InitR' 'C:\Program Files\R\R-4.5.1'
```

Plot

The R Plot functionality is largely replaced by the superior [GGPlot2](#) functionality. The `□RNET Plot` action is not designed for use with `GGPlot2`. The [magick](#) R-package is required to use the `□RNET Plot` action, and it will be installed the first time this action is used.

Syntax: `□RNET 'Plot' tgtFilePath tgtFileType plotCommand width height resolution`

`tgtFilePath` is the full path of the file which contains the `byte[]` array for the plot

`tgtFileType`

The supported target file types for the `□RNET Plot` action are described [here](#). `□RNET` does not validate the `tgtFileType`, because the list is dynamic. If the APL64 programmer-supplied

tgtFileType is not supported, the `RNET Plot` action creates a png-format file, the output file is empty, or an exception is thrown. The `RNET Plot` action does not modify the file extension of the tgtFilePath. Some supported tgtFileTypes:

- BMP: Bitmap, a simple uncompressed image format.
- GIF: Graphics Interchange Format, known for supporting animations and simple images.
- JPEG/JPG: Joint Photographic Experts Group, a lossy compressed format widely used for digital photography
- PDF: Portable Document Format, a document format that can contain images.
- PNG: Portable Network Graphics, a lossless format often used for web graphics.
- TIFF: Tagged Image File Format, a versatile format often used for high-quality images and printing.

plotCommand: Text of the plot commands to be evaluated. Multiple lines in the plotCommand must be separated by newline characters. The APL64 string continuation structure.

plotCommand←««...»». is convenient for creating the plotCommand.

width is the desired width of the plot image in the units applicable to the tgtFileType

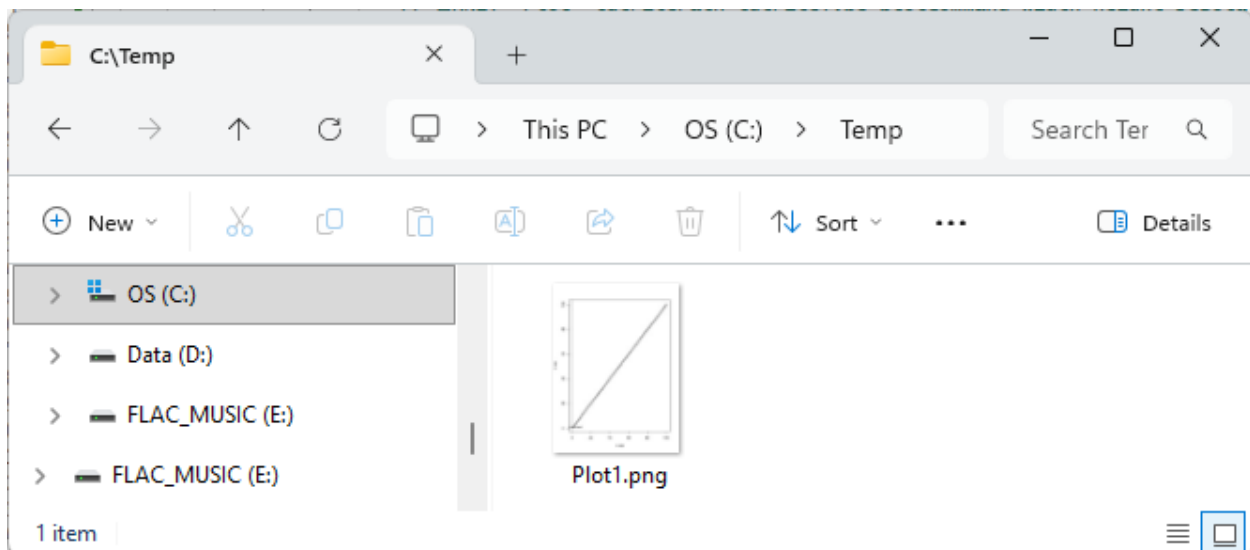
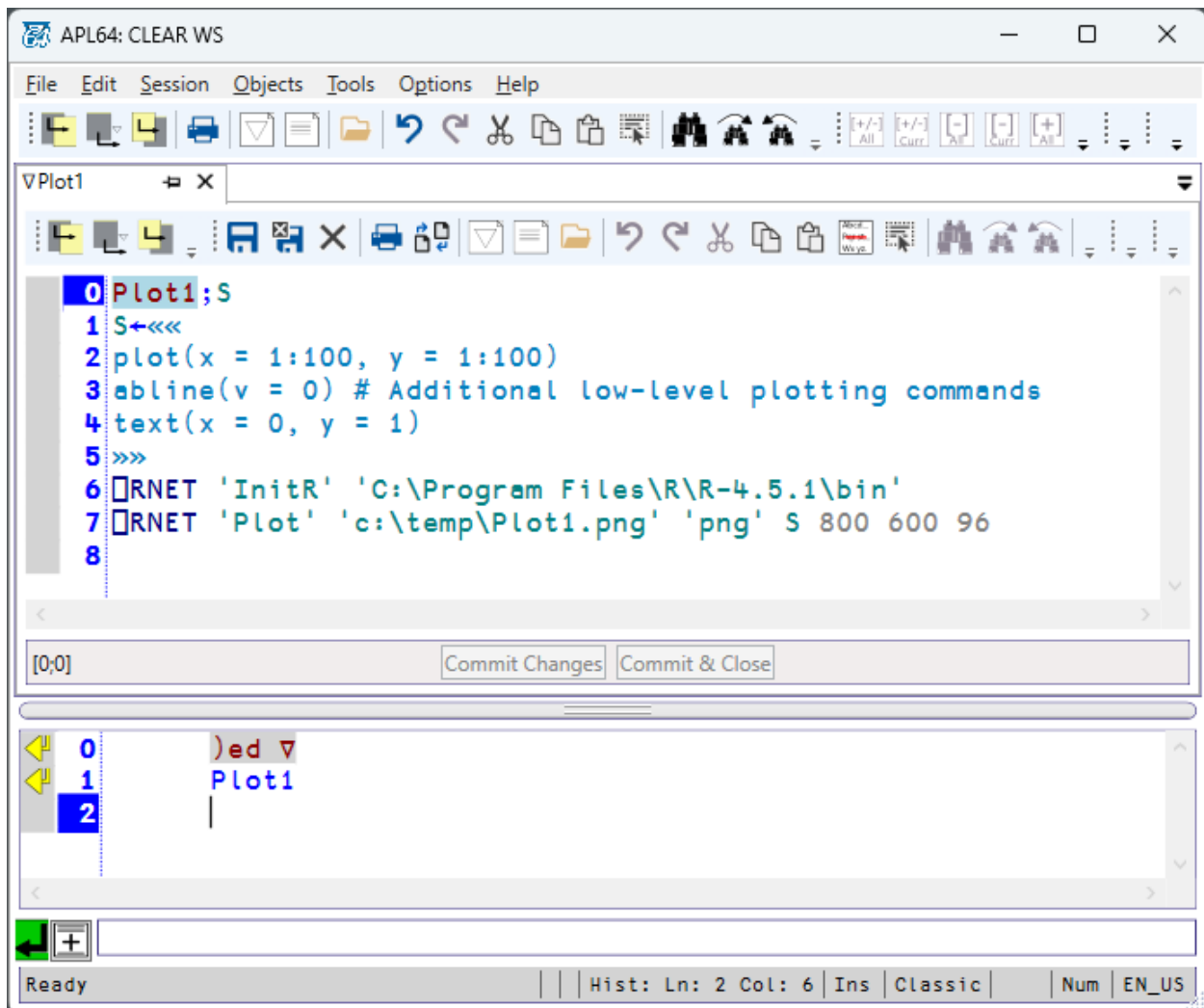
height is the desired height of the plot image in the units applicable to the tgtFileType

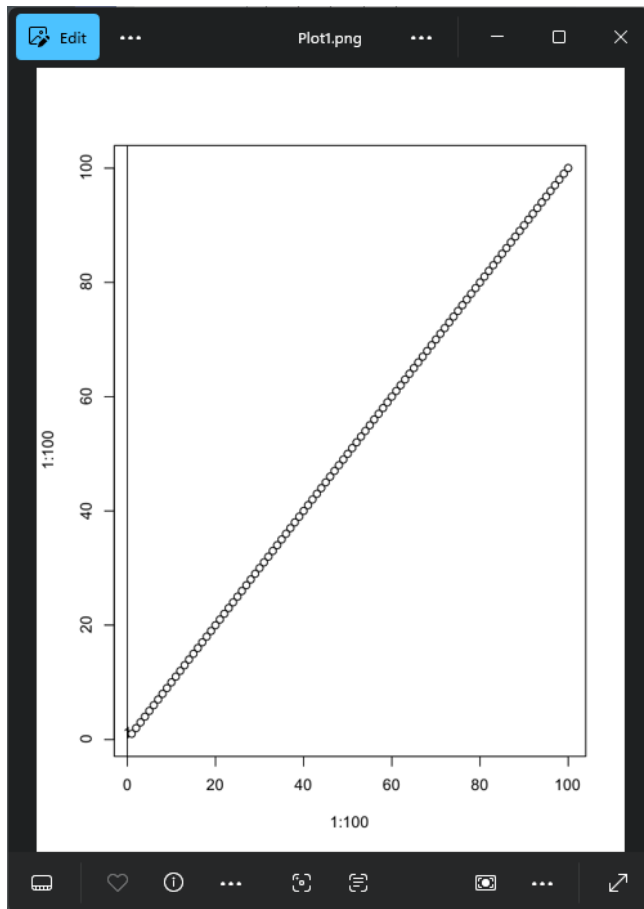
resolution is the desired resolution of the plot image in the units applicable to the tgtFileType

Example: Simple Plot

Create this function and run it:

```
Plot1;S
S←««
plot(x = 1:100, y = 1:100)
abline(v = 0) # Additional low-level plotting commands
text(x = 0, y = 1)
»»
RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin\x64'
RNET 'Plot' 'c:\temp\Plot1.png' 'png' S 800 600 96
```



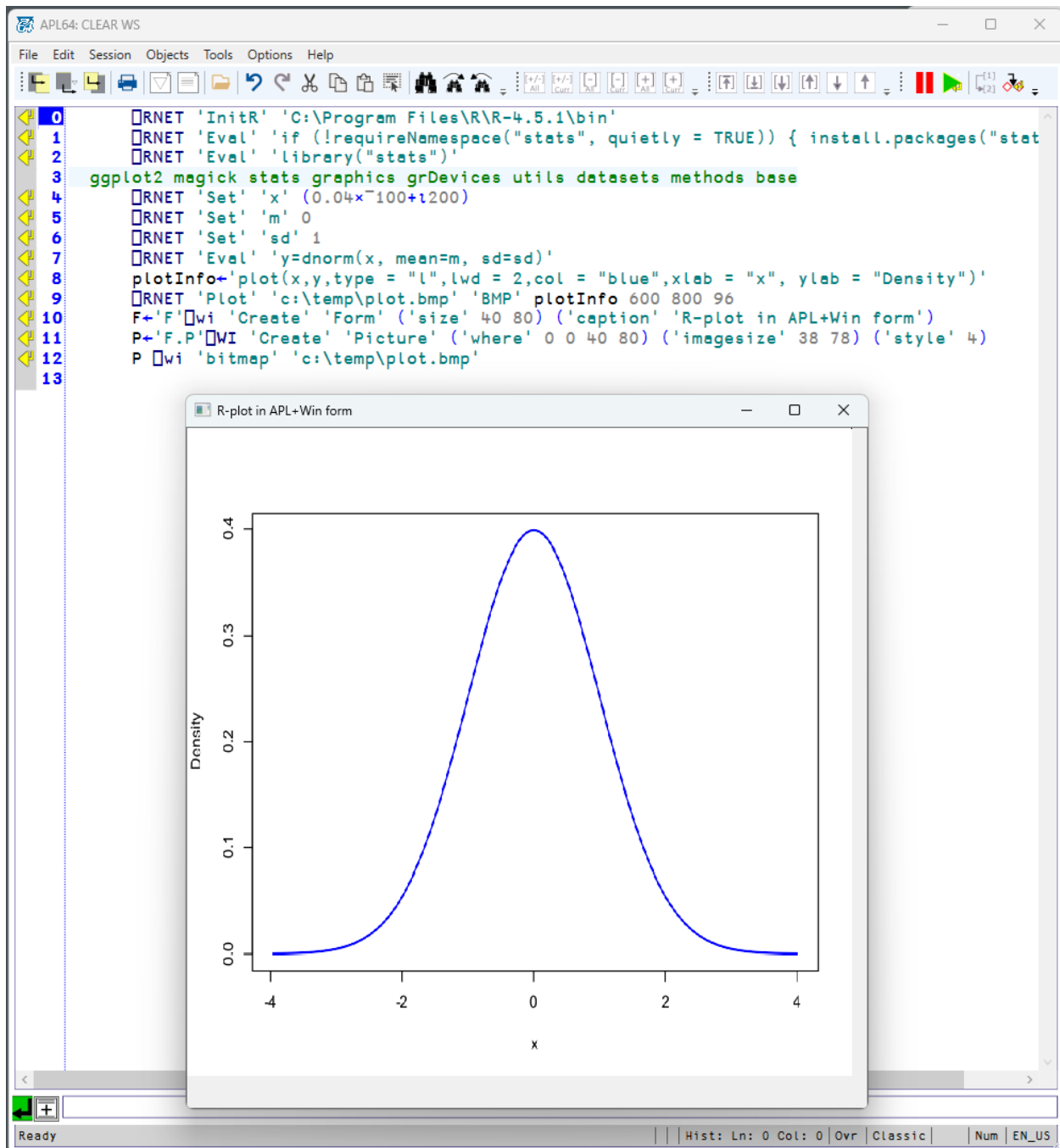


Example: Include Plot in ☐wi Form

```

☐ RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin\x64'
☐ RNET 'Eval' 'if (!requireNamespace("stats", quietly = TRUE)) { install.packages("stats")}'
☐ RNET 'Eval' 'library("stats")'
☐ RNET 'Set' 'x' (0.04×~100+1200)
☐ RNET 'Set' 'm' 0
☐ RNET 'Set' 'sd' 1
☐ RNET 'Eval' 'y=dnorm(x, mean=m, sd=sd)'
plotInfo←'plot(x,y,type = "l",lwd = 2,col = "blue",xlab = "x", ylab = "Density")'
☐ RNET 'Plot' 'c:\temp\plot.bmp' 'BMP' plotInfo 600 800 96
F←'F'☐wi 'Create' 'Form' ('size' 40 80) ('caption' 'R-plot in APL+Win form')
P←'F.P'☐WI 'Create' 'Picture' ('where' 0 0 40 80) ('imagesize' 38 78) ('style' 4)
P ☐wi 'bitmap' 'c:\temp\plot.bmp'

```



SetSymbol

Synonyms: SetSym, Set

The `⎕RNET Set` action is used to pass APL64 data to the R instance in APL64. APL64 supports Int32, double, Boolean and text data. All forms of APL64 text data are converted to the R text data type.

Syntax: `rNetInstanceName ⎕RNET 'SetSymbol' symbolName aplValue`

Example: APL64 scalar

```
⎕RNET 'Set' 'Add3Plus5' '3+5'  
ρ⎕←⎕RNET 'Get' 'Add3Plus5'
```

The screenshot shows the APL64: CLEAR WS window. The code is entered in the editor as follows:

```
0 ⎕RNET 'Set' 'Add3Plus5' '3+5'  
1 ρ⎕←⎕RNET 'Get' 'Add3Plus5'  
2  
3  
4
```

The results are displayed on the left side of the editor:

```
2 3+5  
3 1  
4
```

The status bar at the bottom shows: Ready | Hist: Ln: 4 Col: 6 | Ins | Classic | Num | EN_US

Example: APL64 vector

```
⎕RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin\x64'  
⎕RNET 'Set' 'vec' ("abc" «pqrst» 1 0 3.454 8 2)  
⎕dr"⎕←⎕RNET 'Get' 'vec'
```

The screenshot shows the APL64: CLEAR WS window. The code is entered in the editor as follows:

```
0 ⎕RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin\x64'  
1 ⎕RNET 'Set' 'vec' ("abc" «pqrst» 1 0 3.454 8 2)  
2 ⎕dr"⎕←⎕RNET 'Get' 'vec'  
3  
4  
5
```

The results are displayed on the left side of the editor:

```
3 abc pqrst 1 0 3.454 8 2  
4 162 162 645 645 645 645 645  
5
```

The status bar at the bottom shows: Ready | Hist: Ln: 5 Col: 6 | Ins | Classic | Num | EN_US

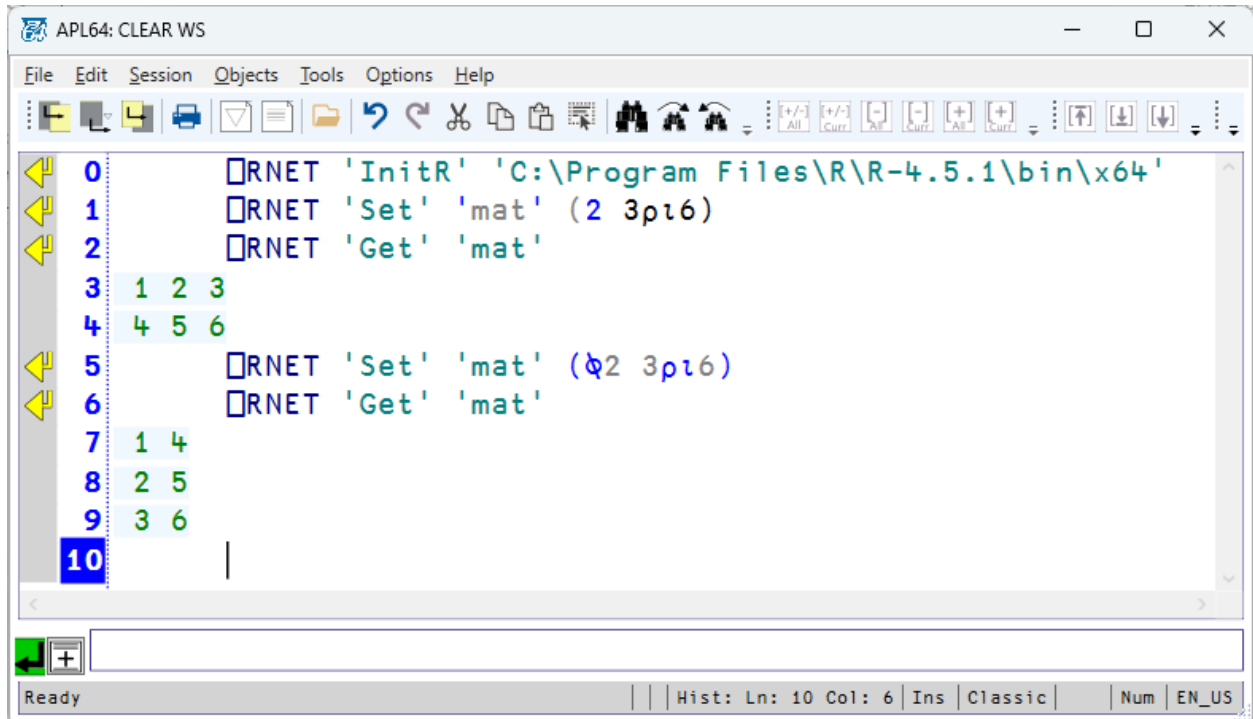
Example: APL64 matrix

```
⎕RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin\x64'  
⎕RNET 'Set' 'mat' (2 3ρ16)
```

```

☐ RNET 'Get' 'mat'
☐ RNET 'Set' 'mat' (⊘2 3p16)
☐ RNET 'Get' 'mat'

```



Example: APL64 homogeneous array

Creating and Executing R-Scripts

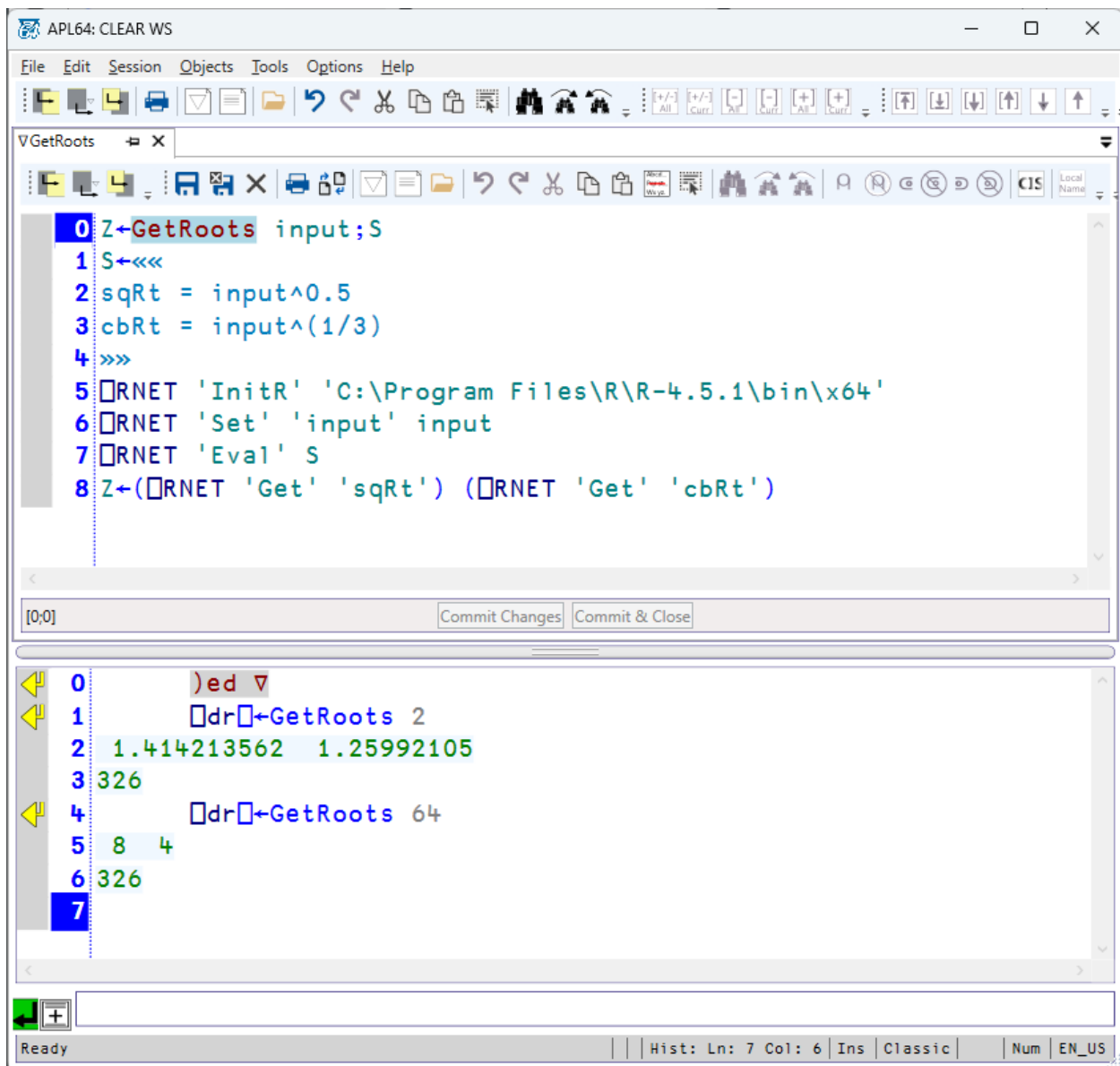
A multi-line R-script can be created in APL64 using the string continuation structure:
`varName←««...»»`. Each line of the script is evaluated individually.

Example: Create this function, containing a multi-line R-script, and run it:

```

Z←GetRoots input;S
S←««
sqRt = input^0.5
cbRt = input^(1/3)
»»
☐ RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin\x64'
☐ RNET 'Set' 'input' input
☐ RNET 'Eval' S
Z←(☐ RNET 'Get' 'sqRt') (☐ RNET 'Get' 'cbRt')

```

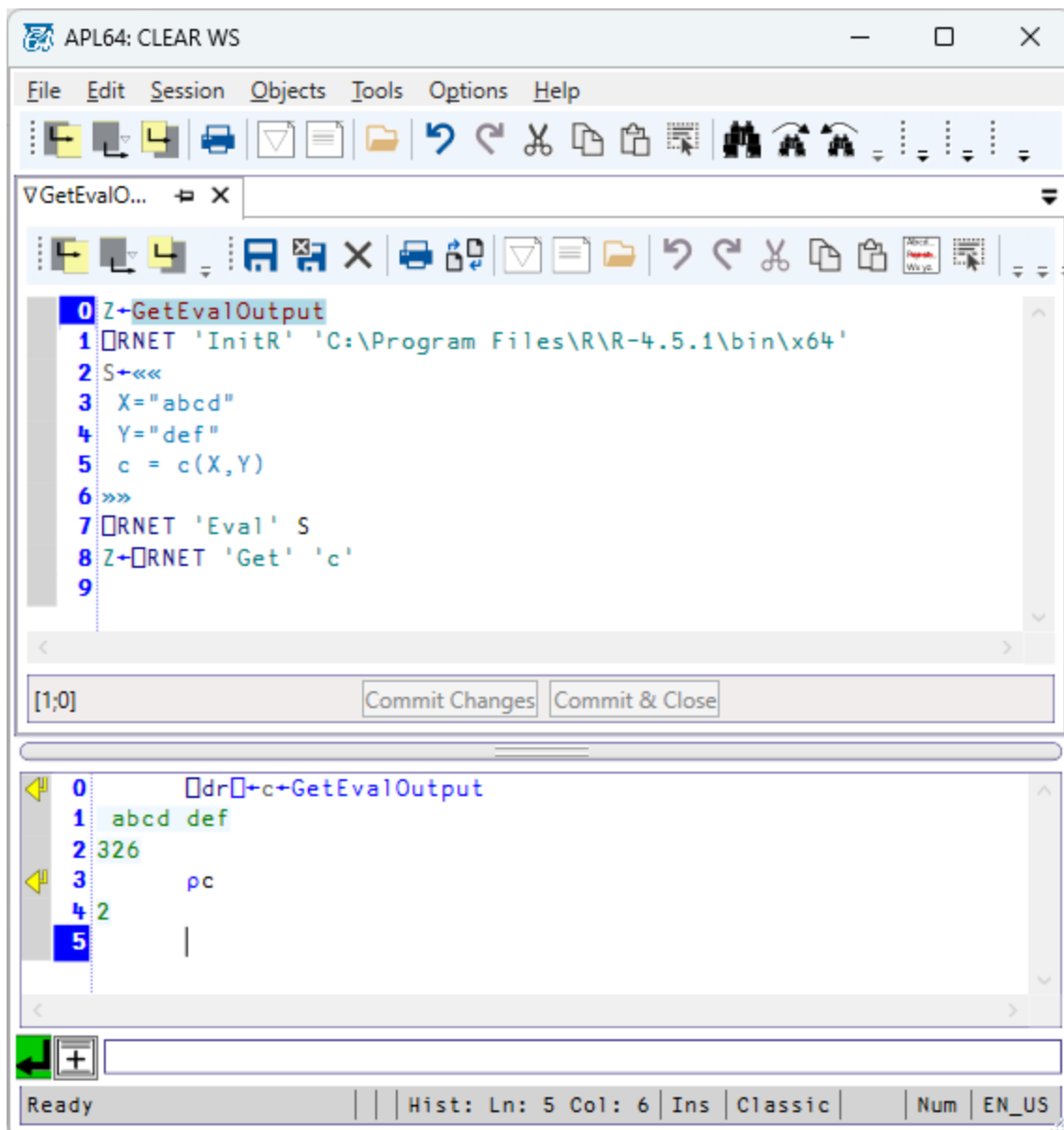


Example: Create this function and run it:

```

Z←GetEvalOutput
⎕RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin\x64'
S←««
X="abcd"
Y="def"
c = c(X,Y)
»»
⎕RNET 'Eval' S
Z←⎕RNET 'Get' 'c'

```



More Examples

Set and Get Symbols: `⍎rNet 'Set'` and `⍎rNet 'Get'`

SetSymbol: Create Rank-2 Character Matrix from APL String Data

```

M←2 3p«abc»
⍎rNet 'Set' 'M' M
⍎dr"⍎←M1←⍎rNet 'Get' 'M'

```

```

0      M←2 3p<<abc>>
1      ⍎rNet 'Set' 'M' M
2      ⍎dr''⍎←M1←⍎rNet 'Get' 'M'
3      abc abc abc
4      abc abc abc
5      162 162 162
6      162 162 162
7

```

Ready | Hist: Ln: 7 Col: 6 | Ins | Classic | Num | EN_US

SetSymbol: Create Rank-2 Character Matrix from APL Character Data

```

⍎dr''⍎←M←2 3p<'abc'
⍎rNet 'Set' 'M' M
⍎dr''⍎←M1←⍎rNet 'Get' 'M'

```

```

0      ⍎rNet 'Set' 'M' M
1      ⍎dr''⍎←M1←⍎rNet 'Get' 'M'
2      abc abc abc
3      abc abc abc
4      162 162 162
5      162 162 162
6

```

Ready | Hist: Ln: 6 Col: 6 | Ins | Classic | Num | EN_US

Set: Create R-Vector from APL Text

```

rNet 'Set' 'vectorOfTextVectors' ('abcd' «pqrst»)
dr'' ← vectorOfTextVectors ← rNet 'Get' 'vectorOfTextVectors'

```

The screenshot shows the APL4: CLEAR WS window. The title bar reads "APL4: CLEAR WS". The menu bar includes "File", "Edit", "Session", "Objects", "Tools", "Options", and "Help". The toolbar contains various icons for file operations, editing, and execution. The code editor displays the following APL code:

```

0      ⍵rNet 'Set' 'vectorOfTextVectors' ('abcd' «pqrst»)
1      ⍵dr"⍵←vectorOfTextVectors←⍵rNet 'Get' 'vectorOfTextVectors'
2      abcd pqrst
3      162 162
4      |

```

The command line at the bottom is empty, and the status bar shows "Ready", "Hist: Ln: 4 Col: 6", "Ins", "Classic", "Num", and "EN_US".

Set: Create Rank-2 R-Matrix using APL Data

```

□rNet 'Set' 'A' (2 3ρ16)
ρ□←A←□rNet 'Get' 'A'

```

APL64: CLEAR WS

File Edit Session Objects Tools Options Help

0 rNet 'Set' 'A' (2 3 pi 6)

1 p←A←rNet 'Get' 'A'

2 1 2 3

3 4 5 6

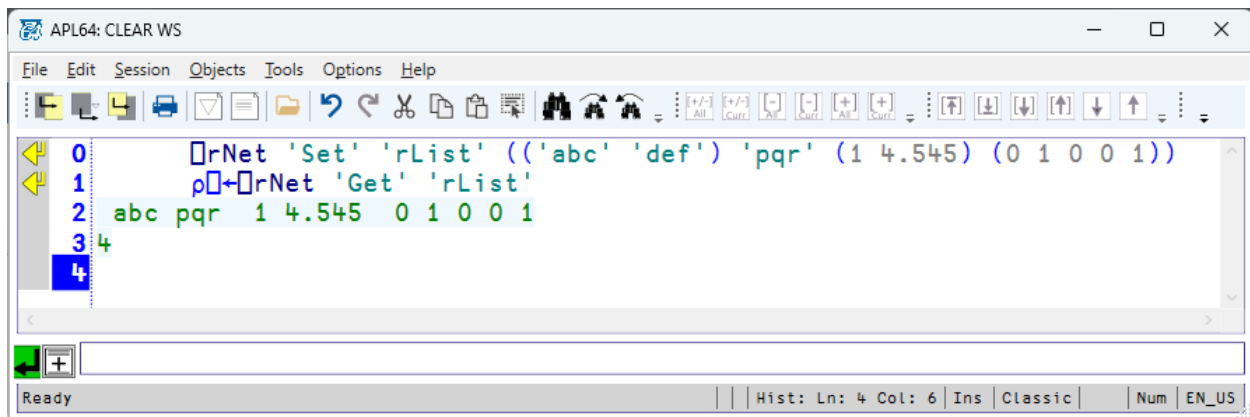
4 2 3

5

Ready | Hist: Ln: 5 Col: 6 | Ins | Classic | Num | EN_US

Set: Create Nested R-List Containing Mixed Data Types

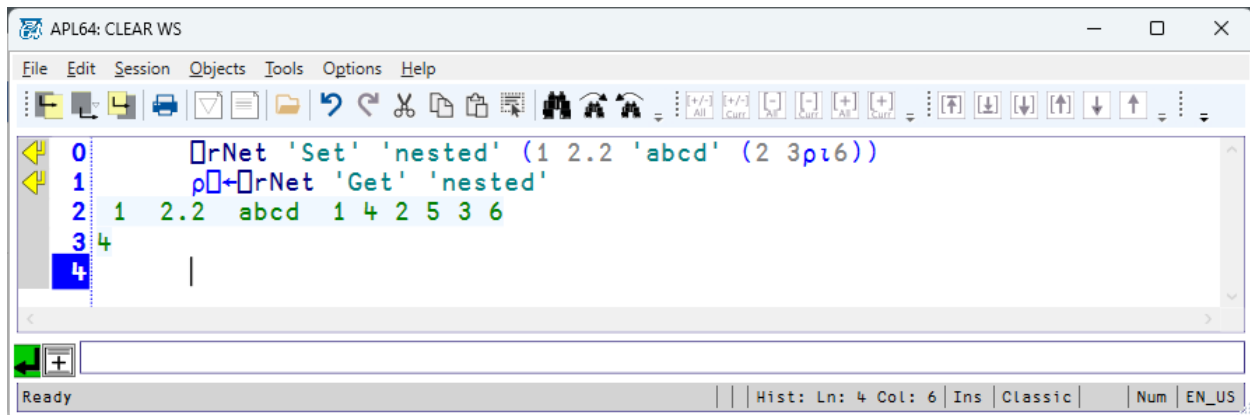
```
☐ rNet 'Set' 'rList' (('abc' 'def') 'pqr' (1 4.545) (0 1 0 0 1))  
☐ rNet 'Get' 'rList'
```



Set: Create Nexted R-List Containing Mixed Data Types: APL Array of Rank > 2 is Raveled

A nested R-List can contain R-Lists, but not arrays of rank greater than 1.

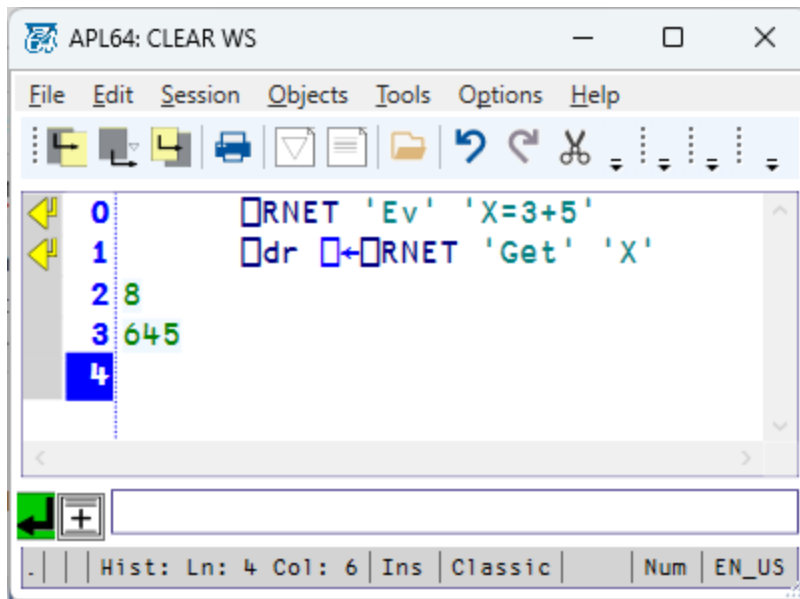
```
☐ rNet 'Set' 'nested' (1 2.2 'abcd' (2 3p16))  
☐ rNet 'Get' 'nested'
```



Evaluate: ☐←☐RNET 'Eval'

Evaluate: Arithmetic Expression Without R-Symbol Assignment

```
☐ RNET 'Ev' 'X=3+5'  
☐ dr ☐←☐ RNET 'Get' 'X'
```

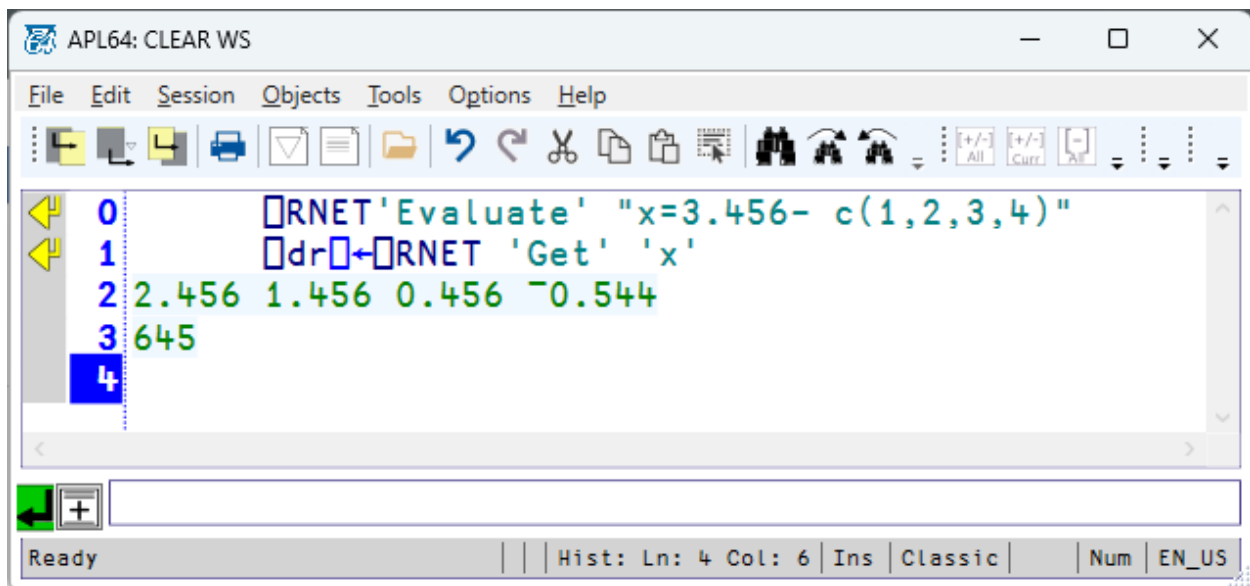


Evaluate: Create Numeric R-Vector With R-Symbol Assignment

```

☐ RNET 'Evaluate' "x=3.456- c(1,2,3,4)"
☐ dr [ ] ← ☐ RNET 'Get' 'x'

```

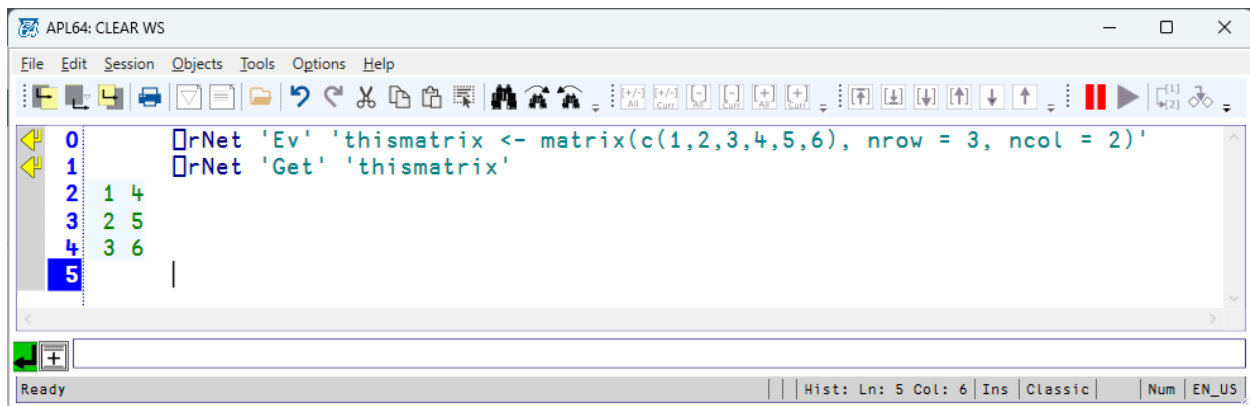


Evaluate: Create Rank-2 R-Matrix in R-Engine

```

☐ rNet 'Ev' 'thismatrix <- matrix(c(1,2,3,4,5,6), nrow = 3, ncol = 2)'
☐ rNet 'Get' 'thismatrix'

```

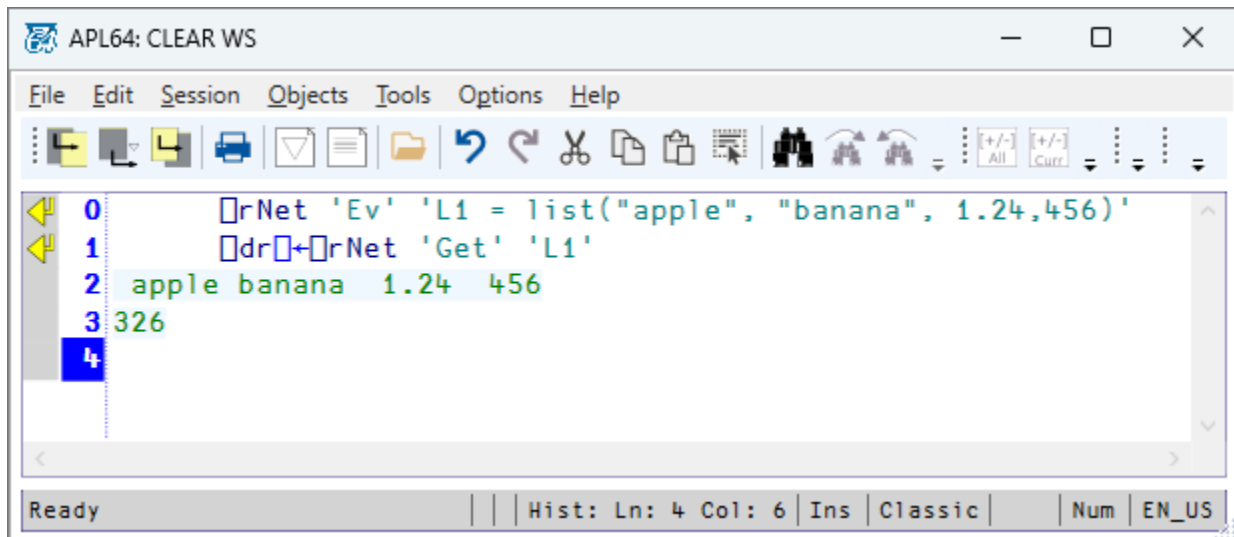


Evaluate: Create Simple R-List Containing Mixed Data Types

```

⍶rNet 'Ev' 'L1 = list("apple", "banana", 1.24,456)'
⍶dr⍶←⍶rNet 'Get' 'L1'

```

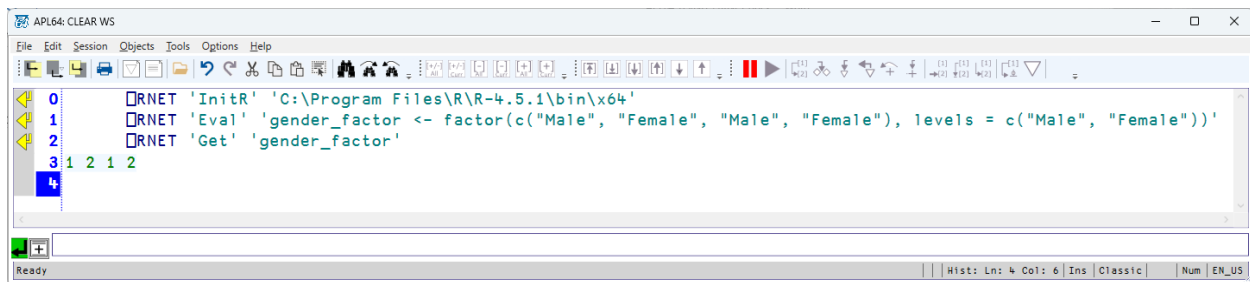


Evaluate: Create an R-factor

```

⍶RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin\x64'
⍶RNET 'Eval' 'gender_factor <- factor(c("Male", "Female", "Male", "Female"), levels = c("Male", "Female"))'
⍶RNET 'Get' 'gender_factor'

```



Creating DataFrames

Creating a DataFrame two ways

Using the ☐ RNET 'Cdf' action

```

dfName<-'df2'
dataMat<-(1 'koala' 'Australia' 21)
dataMat<-dataMat ☐ OVER 2 'hedgehog' 'Italy' 18
dataMat<-dataMat ☐ OVER 3 'sloth' 'Peru' 17
dataMat<-dataMat ☐ OVER 4 'panda' 'China' 10
dataMat
rowNames<-'r1' 'r2' 'r3' 'r4'
colNames<-'rating' 'animal' 'country' 'avg_sleep_hours'
☐ RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin\x64'
☐ RNET 'CDF' dfName dataMat 1 1 0 colNames rowNames
☐ RNET 'Get' dfName
☐ RNET 'Eval' "p1=print(df2)"
☐ RNET 'Get' 'p1'

```

APL64: CLEAR WS

File Edit Session Objects Tools Options Help

0 dfName←'df2'

1 dataMat←(1 'koala' 'Australia' 21)

2 dataMat←dataMat □OVER 2 'hedgehog' 'Italy' 18

3 dataMat←dataMat □OVER 3 'sloth' 'Peru' 17

4 dataMat←dataMat □OVER 4 'panda' 'China' 10

5 dataMat

6 1 koala Australia 21

7 2 hedgehog Italy 18

8 3 sloth Peru 17

9 4 panda China 10

10 rowNames←'r1' 'r2' 'r3' 'r4'

11 colNames←'rating' 'animal' 'country' 'avg_sleep_hours'

12 □RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin\x64'

13 □RNET 'CDF' dfName dataMat 1 1 0 colNames rowNames

14 □RNET 'Get' dfName

15

	r1	r2	r3	r4
rating	1	2	3	4
animal	koala	hedgehog	sloth	panda
country	Australia	Italy	Peru	China
avg_sleep_hours	21	18	17	10

20 □RNET 'Eval' "p1=print(df2)"

21 □RNET 'Get' 'p1'

22

	r1	r2	r3	r4
rating	1	2	3	4
animal	koala	hedgehog	sloth	panda
country	Australia	Italy	Peru	China
avg_sleep_hours	21	18	17	10

27

Ready | Hist: Ln: 27 Col: 6 | Ins | Classic | Num | EN_US

The same dataframe created without using the □RNET 'CreateDataFrame' action:

```

☐ RNET 'Eval' "rating <- 1:4"
☐ RNET 'Eval' "animal <- c('koala', 'hedgehog', 'sloth', 'panda')"
☐ RNET 'Eval' "country <- c('Australia', 'Italy', 'Peru', 'China')"
☐ RNET 'Eval' "avg_sleep_hours <- c(21, 18, 17, 10)"
☐ RNET 'Eval' "rowNames=c('r1', 'r2', 'r3', 'r4')"
☐ RNET 'Eval' "df = data.frame(rating, animal, country, avg_sleep_hours, stringsAsFactors=FALSE,
row.names=rowNames)"
ρ☐←☐ RNET 'Get' 'df2'

```

The screenshot shows the APL64: CLEAR WS interface. The command window contains the following RNET statements:

```

0 ☐ RNET 'Eval' "rating <- 1:4"
1 ☐ RNET 'Eval' "animal <- c('koala', 'hedgehog', 'sloth', 'panda')"
2 ☐ RNET 'Eval' "country <- c('Australia', 'Italy', 'Peru', 'China')"
3 ☐ RNET 'Eval' "avg_sleep_hours <- c(21, 18, 17, 10)"
4 ☐ RNET 'Eval' "rowNames=c('r1', 'r2', 'r3', 'r4')"
5 ☐ RNET 'Eval' "df = data.frame(rating, animal, country, avg_sleep_hours, stringsAsFactors=FALSE, row.names=rowNames)"
6 ρ☐←☐ RNET 'Get' 'df2'

```

The result window displays the data frame:

	r1	r2	r3	r4
rating	1	2	3	4
animal	koala	hedgehog	sloth	panda
country	Australia	Italy	Peru	China
avg_sleep_hours	21	18	17	10

The status bar at the bottom indicates: Ready | Hist: Ln: 13 Col: 6 | Ins | Classic | Num | EN_US

Create R-DataFrame and obtain R-String with R Statements

```

☐ rNet 'Ev' "cases <- expand.grid(x=c('a','b','c'), y=1:3)"
☐ dr"☐←☐ rNet 'Get' 'cases'
☐ rNet 'Ev' "casesY=cases[, 'y']"
☐ dr"☐←☐ rNet 'Get' 'casesY'

```

```

APL64: CLEAR WS
File Edit Session Objects Tools Options Help
0  ⎕rNet 'Ev' "cases <- expand.grid(x=c('a','b','c'), y=1:3)"
1  ⎕dr"⎕←⎕rNet 'Get' 'cases'
2  1 2 3 4 5 6 7 8 9
3  x 1 2 3 1 2 3 1 2 3
4  y 1 1 1 2 2 2 3 3 3
5  162 162 162 162 162 162 162 162 162
6  162 323 323 323 323 323 323 323 323
7  162 323 323 323 323 323 323 323 323
8  ⎕rNet 'Ev' "casesY=cases[, 'y']"
9  ⎕dr"⎕←⎕rNet 'Get' 'casesY'
10 1 1 1 2 2 2 3 3 3
11 323 323 323 323 323 323 323 323 323
12

```

Accessing Dataframe Columns

Create this function and run it:

```

DataFrameColumns;s
s<- 'Prod1' 'lb' 5.44 23.2 1
s<-s⎕over 'Prod2' 'oz' 3.42 0.0 0
s<-s⎕over «Prod3» 'kg' 2.01 2.2 0
s<-s⎕over 'Prod4' 'ml' 13.99 10 1
s<-s⎕over 'Prod5' 'ctn' 30.01 0 0
s
colNames<- 'Product Type' 'Unit' 'Unit Cost' 'Bulk Disc Pct' 'Available'
rowNames<- 'P1' 'P2' 'P3' 'P4' 'P5'
⎕RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin\x64'
⎕RNET 'Cdf' 'df1' s 1 1 0 colNames rowNames
⎕RNET 'Get' 'df1'
⎕RNET 'Eval' 'p1=print(df1)'
⎕RNET 'Get' 'p1'
⎕RNET 'eval' 'p2=print(sapply(df1, class))'
⎕RNET 'get' 'p2'
⎕RNET 'Eval' 'v1=df1[2]'
⎕RNET 'Get' 'v1'
⎕RNET 'Eval' 'v2=df1[[2]]'
⎕RNET 'Get' 'v2'
⎕RNET 'Eval' 'v3=df1[[3]]'
⎕RNET 'Get' 'v3'
⎕RNET 'Eval' 'v4=df1["Unit"]'
⎕RNET 'Get' 'v4'

```

☐ RNET 'Eval' 'v5=df1[["Unit.Cost"]]'

☐ RNET 'Get' 'v5'

☐ RNET 'Eval' 'v6=df1\$Available'

☐ RNET 'Get' 'v6'

APL64: CLEAR WS

File Edit Session Objects Tools Options Help

▼ DataFrame...

```

0 DataFrameColumns:s
1 s←'Prod1' 'lb' 5.44 23.2 1
2 s←s⊞over 'Prod2' 'oz' 3.42 0.0 0
3 s←s⊞over «Prod3» 'kg' 2.01 2.2 0
4 s←s⊞over 'Prod4' 'ml' 13.99 10 1
5 s←s⊞over 'Prod5' 'ctn' 30.01 0 0
6 s
7 colNames←'Product Type' 'Unit' 'Unit Cost' 'Bulk Disc Pct' 'Available'
8 rowNames←'P1' 'P2' 'P3' 'P4' 'P5'
9 ⌹RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin\x64'
10 ⌹RNET 'Cdf' 'df1' s 1 1 0 colNames rowNames
11 ⌹RNET 'Get' 'df1'
12 ⌹RNET 'Eval' 'n1=print(df1)'
13 ⌹RNET System function {⌹RNET}
14 ⌹RNET ly(df1, class))'
15 ⌹RNET
16 ⌹RNET 'Eval' 'v1=df1[2]'
17 ⌹RNET 'Get' 'v1'
18 ⌹RNET 'Eval' 'v2=df1[[2]]'
19 ⌹RNET 'Get' 'v2'
20 ⌹RNET 'Eval' 'v3=df1[[3]]'
21 ⌹RNET 'Get' 'v3'
22 ⌹RNET 'Eval' 'v4=df1["Unit"]'
23 ⌹RNET 'Get' 'v4'
24 ⌹RNET 'Eval' 'v5=df1[["Unit.Cost"]]'
25 ⌹RNET 'Get' 'v5'
26 ⌹RNET 'Eval' 'v6=df1$Available'
27 ⌹RNET 'Get' 'v6'
28

```

[0;16] Commit Changes Commit & Close

0 DataFrameColumns

```

1 Prod1 lb 5.44 23.2 1
2 Prod2 oz 3.42 0.0 0
3 Prod3 kg 2.01 2.2 0
4 Prod4 ml 13.99 10.0 1
5 Prod5 ctn 30.01 0.0 0
6
7 Product.Type Prod1 Prod2 Prod3 Prod4 Prod5
8 Unit lb oz kg ml ctn
9 Unit.Cost 5.44 3.42 2.01 13.99 30.01
10 Bulk.Disc.Pct 23.20 0.00 2.20 10.00 0.00
11 Available 1.00 0.00 0.00 1.00 0.00
12
13 Product.Type Prod1 Prod2 Prod3 Prod4 Prod5
14 Unit lb oz kg ml ctn
15 Unit.Cost 5.44 3.42 2.01 13.99 30.01
16 Bulk.Disc.Pct 23.20 0.00 2.20 10.00 0.00
17 Available 1.00 0.00 0.00 1.00 0.00
18 character character numeric numeric logical
19 P1 P2 P3 P4 P5
20 Unit lb oz kg ml ctn
21 lb oz kg ml ctn
22 5.44 3.42 2.01 13.99 30.01
23 P1 P2 P3 P4 P5
24 Unit lb oz kg ml ctn
25 5.44 3.42 2.01 13.99 30.01
26 1 0 0 1 0
27

```

Ready

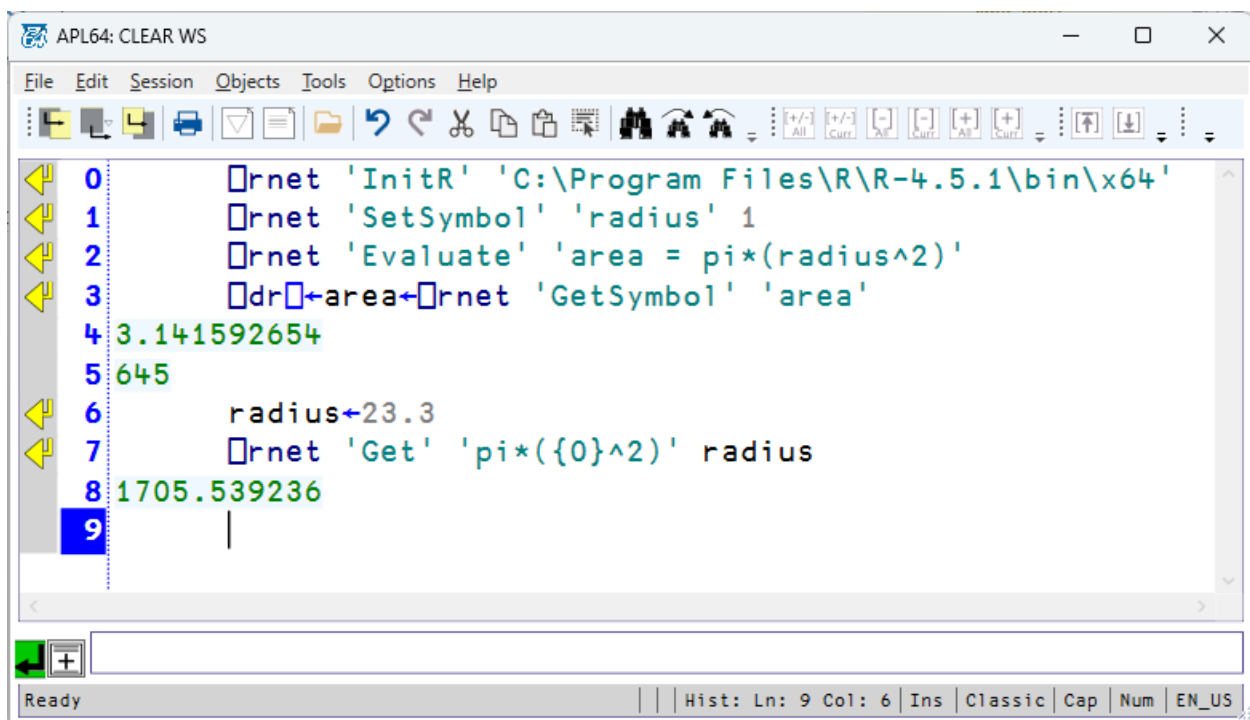
Hist: Ln: 27 Col: 6

Ins Classic Num EN_US

Using R-Engine for *ad hoc* Calculations

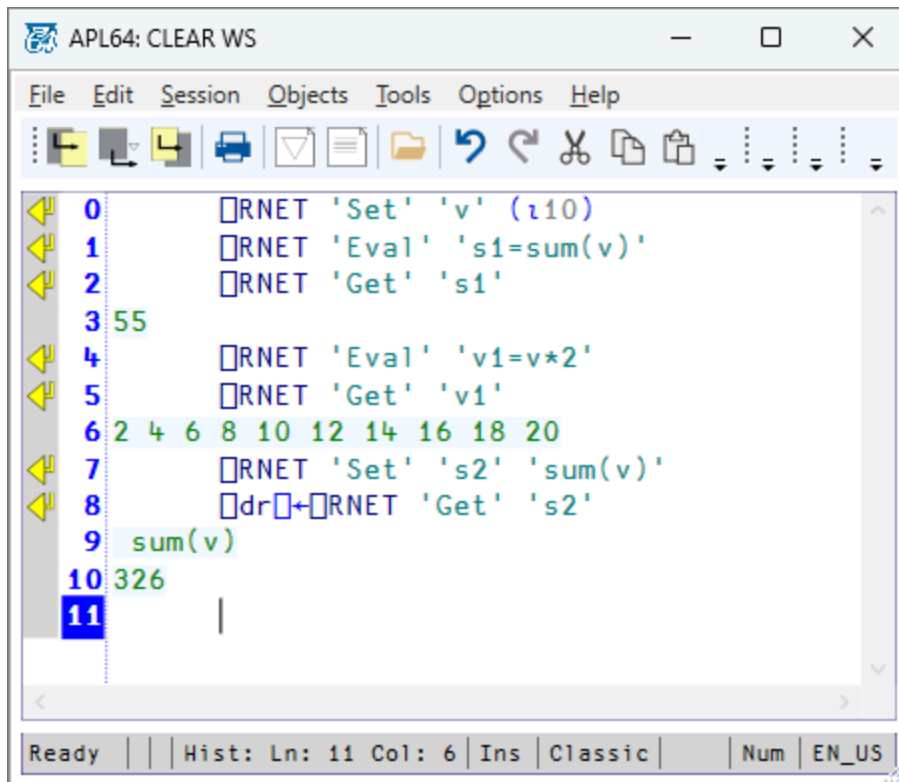
In this example, the input value, radius, is provided to the R engine and used to calculate the area of a circle using the R engine Evaluate action. The ☐rnet 'Get' action with value substitution can simplify the operation:

```
☐rnet 'InitR' 'C:\Program Files\R\R-4.5.1\bin\x64'  
☐rnet 'SetSymbol' 'radius' 1  
☐rnet 'Evaluate' 'area = pi*(radius^2)'  
☐dr☐←area←☐rnet 'GetSymbol' 'area'  
radius←23.3  
☐rnet 'Get' 'pi*({0}^2)' radius
```



In this example the R sum() and multiply actions are used:

```
☐RNET 'Set' 'v' (10)  
☐RNET 'Eval' 's1=sum(v)'  
☐RNET 'Get' 's1'  
☐RNET 'Eval' 'v1=v*2'  
☐RNET 'Get' 'v1'  
☐RNET 'Set' 's2' 'sum(v)'  
☐dr☐←☐RNET 'Get' 's2'
```



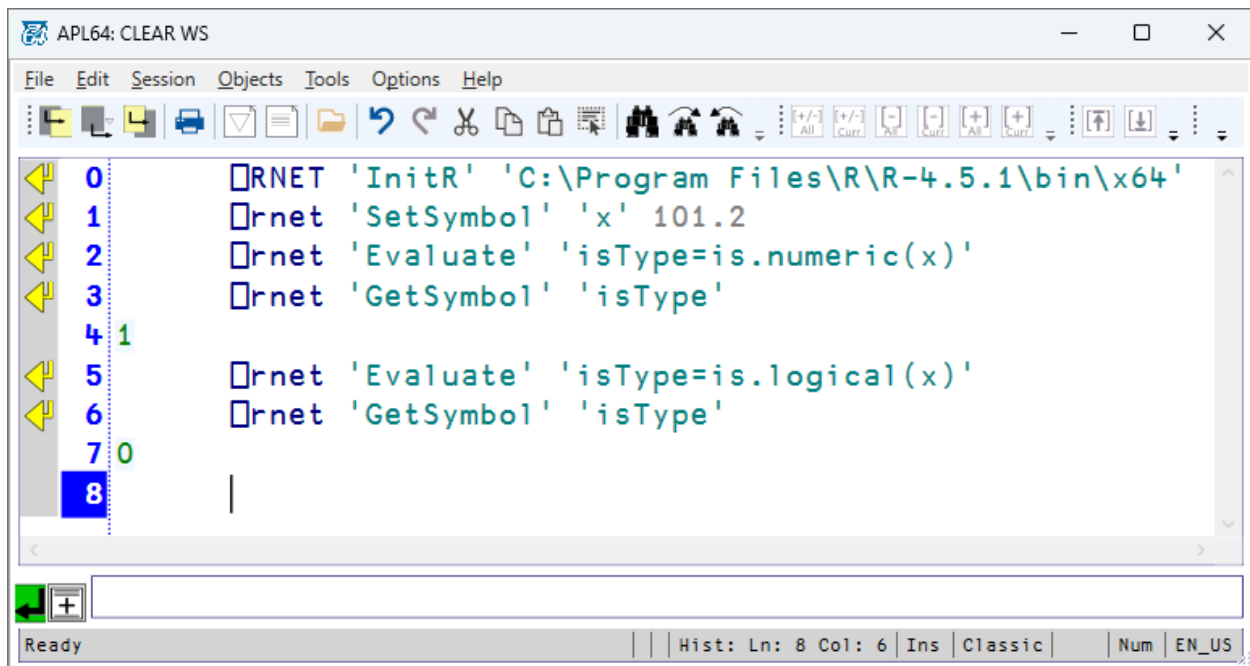
The screenshot shows the APL64: CLEAR WS window. The menu bar includes File, Edit, Session, Objects, Tools, Options, and Help. The toolbar contains icons for file operations and editing. The main text area displays the following RNET code and execution results:

```
0 [RNET 'Set' 'v' (10)
1 [RNET 'Eval' 's1=sum(v)'
2 [RNET 'Get' 's1'
3 55
4 [RNET 'Eval' 'v1=v*2'
5 [RNET 'Get' 'v1'
6 2 4 6 8 10 12 14 16 18 20
7 [RNET 'Set' 's2' 'sum(v)'
8 [dr[←[RNET 'Get' 's2'
9 sum(v)
10 326
11 |
```

The status bar at the bottom shows: Ready | | Hist: Ln: 11 Col: 6 | Ins | Classic | Num | EN_US

Check the Data Type of an R symbol

- ☐ RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin\x64'
- ☐ rnet 'SetSymbol' 'x' 101.2
- ☐ rnet 'Evaluate' 'isType=is.numeric(x)'
- ☐ rnet 'GetSymbol' 'isType'
- ☐ rnet 'Evaluate' 'isType=is.logical(x)'
- ☐ rnet 'GetSymbol' 'isType'

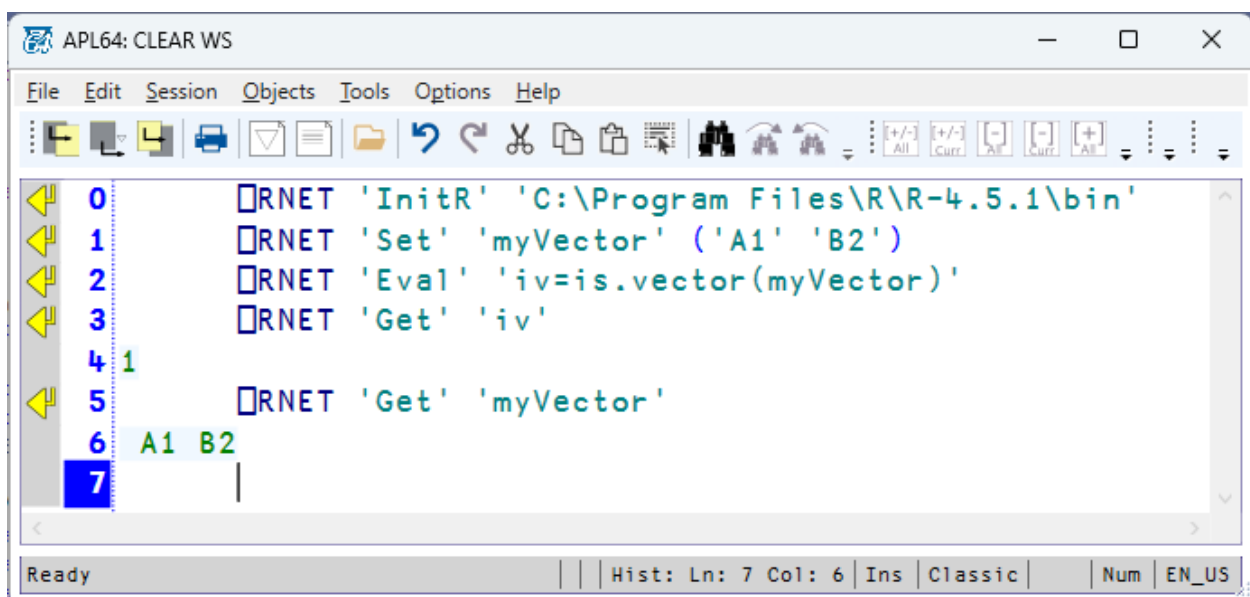


Check if an R-object is a vector

```

☐ RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin'
☐ RNET 'Set' 'myVector' ('A1' 'B2')
☐ RNET 'Eval' 'iv=is.vector(myVector)'
☐ RNET 'Get' 'iv'
☐ RNET 'Get' 'myVector'

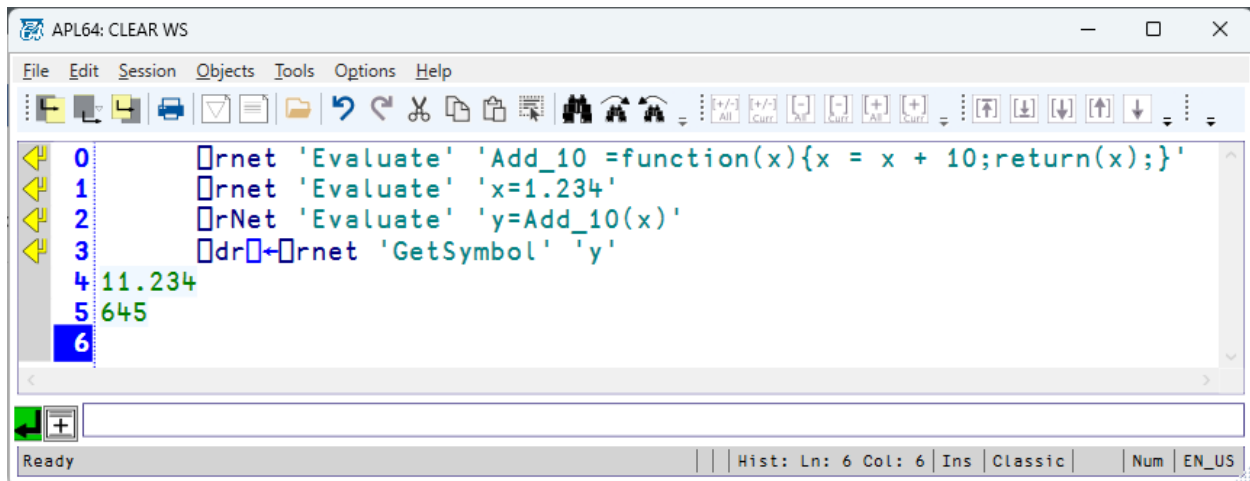
```



Creating and Executing an R-Function with One Argument

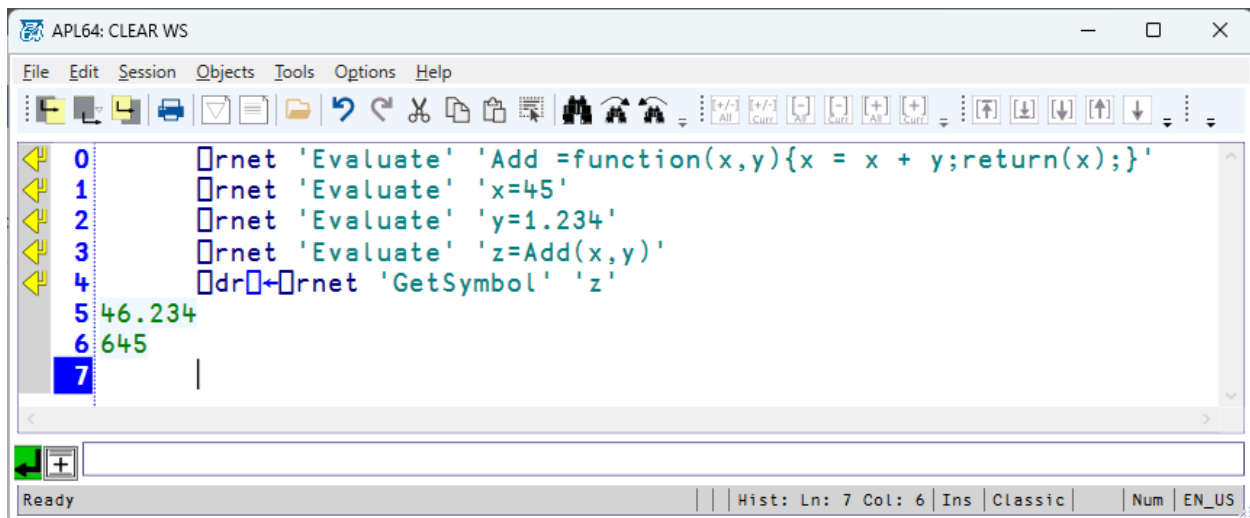
Sometimes it is convenient to create a user-defined function in R so that it can be executed several times during the APL64 session.

```
☐ rnet 'Evaluate' 'Add_10 =function(x){x = x + 10;return(x);}'  
☐ rnet 'Evaluate' 'x=1.234'  
☐ rNet 'Evaluate' 'y=Add_10(x)'  
☐ dr ☐ ← ☐ rnet 'GetSymbol' 'y'
```



Creating an Executing an R-Function with Multiple Arguments

```
☐ rnet 'Evaluate' 'Add =function(x,y){x = x + y;return(x);}'  
☐ rnet 'Evaluate' 'x=45'  
☐ rnet 'Evaluate' 'y=1.234'  
☐ rnet 'Evaluate' 'z=Add(x,y)'  
☐ dr ☐ ← ☐ rnet 'GetSymbol' 'z'
```



T-Test

```

⎕rNet 'Set' 'grp1' (30.02 29.99 30.11 29.97 30.01 29.99)
⎕rNet 'Set' 'grp2' (29.89 29.93 29.72 29.98 30.02 29.98)
⎕rNet 'Ev' 'tTest = t.test(grp1, grp2)'
10 1p⎕rNet 'Get' 'tTest'
p⎕rNet 'Get' 'tTest'

```

```

0  ⎕rNet 'Set' 'grp1' (30.02 29.99 30.11 29.97 30.01 29.99)
1  ⎕rNet 'Set' 'grp2' (29.89 29.93 29.72 29.98 30.02 29.98)
2  ⎕rNet 'Ev' 'tTest = t.test(grp1, grp2)'
3  10 1p⎕rNet 'Get' 'tTest'
4      1.959005808
5      7.03055996
6      0.09077332429
7  -0.01956908965 0.2095690896
8      30.015 29.92
9      0
10     0.04849398588
11     two.sided
12     Welch Two Sample t-test
13     grp1 and grp2
14  p⎕rNet 'Get' 'tTest'
15 10
16

```

Ready | Hist: Ln: 16 Col: 6 | Ins | Classic | Num | EN_US

Plot

R-Plot of Random Data

In this example two R-plots are overlaid in the same chart. The first is a histogram of sample 'average' values and the second is a line chart of the same 'average' values illustrating the anticipated observation of the Central Limit Theorem.

Create this APL function and run it:

```

CentrallimitChart
⌕ APLNext 20140222
:TRY *
chartFnm←«chartFnm.bmp»
⎕nfe 'Delete' chartFnm
⎕RNET'SetSymbol' 'chartFnm' chartFnm
⎕RNET'Evaluate' 'bmp(chartFnm)'
⌕ ↑ Set the destination of the chart bitmap

```

```

trialVals←1E-4×16 1000p?“(×/16 1000)p×/16 1000
⌘↑ Prepare trial values using APL64 roll function for pseudo-random trials
⌘↑ In a production environment the trialVals would come from a simulation or experiment
□RNET'SetSymbol' 'trialVals' trialVals
⌘↑ Send trialVals to R
□RNET'Evaluate' 'mns=cbind(trialVals[1,],apply(trialVals [1:4,], 2, mean),apply(trialVals
[1:16,], 2, mean))'
⌘↑ Compute means of samples 1, 4, 16 and assign them to the r-Value mns
□RNET'Evaluate' 'meds=cbind(trialVals [1,],apply(trialVals [1:4,], 2, median),apply(trialVals
[1:16,], 2, median))'
⌘↑ Compute the medians of samples of 1, 4, 16 and assign them to the r-Value meds
□RNET'Evaluate' 'hist(mns[,3],main ="Means of samples of size 16",xlab = "Size 16 means",
las=1,col="darkred", breaks=50, prob = TRUE)'
□RNET'Evaluate' 'lines(density(mns[,3]),col="blue")'
⌘↑ Define the R-plot type, axis labels, colors, etc.
□RNET'Evaluate' 'dev.off()'
⌘↑ Write the chart data to the target bitmap file
F←'F'□wi 'Create' 'Form' ('size' 40 80) ('caption' 'R-plot in APL64')
P←'F.P'□wi 'Create' 'Picture' ('where' 0 0 40 80) ('size' 30 60) ('style' 0)
P□wi 'bitmap' 'chartFnm.bmp'
sink←F□wi 'Wait'
⌘↑ Display the chart in a modal APL form
:CATCH
'CentralLimitChart failed: ',□dm
:ENDTRY

```

When the APL64 ‘CentralLimitChart’ function is run, an APL64 form is displayed containing the JPEG-format image containing the R-plot. Since the ‘trial’ data used in this example is based on the APL64 ‘deal’ function, running this function, a few times gives a graphic representation of the APL64 random number generator results.

APL64: CLEAR WS

File Edit Session Objects Tools Options Help

VCentralLi...

```

0  CentralLimitChart
1  APLNext 20140222
2  :TRY *
3  chartFnm←«chartFnm.bmp»
4  ⎕nf 'Delete' chartFnm
5  ⎕RNET'SetSymbol' 'chartFnm' chartFnm
6  ⎕RNET'Evaluate' 'bmp(chartFnm)'
7  ⎕t Set the destination of the chart bitmap
8  trialVals←1E-4×16 1000p?“(×/16 1000)p×/16 1000
9  ⎕t Prepare trial values using APL64 roll function for pseudo-random trials
10 ⎕t In a production environment the trialVals would come from a simulation or experiment
11 ⎕RNET'SetSymbol' 'trialVals' trialVals
12 ⎕t Send trialVals to R
13 ⎕RNET'Evaluate' 'mns=cbind(trialVals[1,],apply(trialVals [1:4,], 2, mean),apply(trialVals [1:16,], 2, mean))'
14 ⎕t Compute means of samples 1, 4, 16 and assign them to the r-Value mns
15 ⎕RNET'Evaluate' 'meds=cbind(trialVals [1,],apply(trialVals [1:4,], 2, median),apply(trialVals [1:16,], 2, median))'
16 ⎕t Compute the medians of samples of 1, 4, 16 and assign them to the r-Value meds
17 ⎕RNET'Evaluate' 'hist(mns[,3],main="Means of samples of size 16",xlab="Size 16 means", las=1,col="darkred")'
18 ⎕RNET'Evaluate' 'lines(density(mns[,3]),col="blue")'
19 ⎕t Define the R-plot type, axis labels, colors, etc.
20 ⎕RNET'Evaluate' 'dev.off()'
21 ⎕t Write the chart data to the target bitmap file
22 F←'F'⎕wi 'Create' 'Form' ('size' 40 80) ('caption' 'R-plot in APL64')
23 P←'F.P'⎕wi 'Create' 'Picture' ('where' 0 0 40 80) ('size' 30 60) ('style' 0)
24 P⎕wi 'bitmap' 'chartFnm.bmp'
25 sink←F⎕wi 'Wait'
26 ⎕t Display the chart in a modal APL form
27 :CATCH
28 'CentralLimitChart failed: ',⎕dm
29 :ENDTRY
30

```

[0;17]

Commit Changes Commit & Close

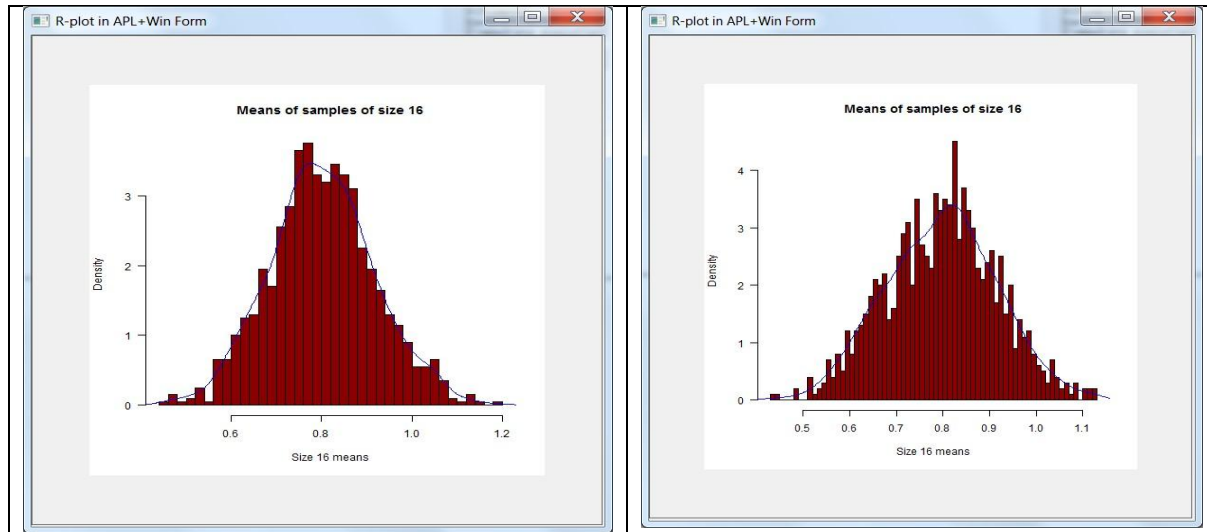
```

0  CentralLimitChart
1  0.3 0.32 0.34 0.36 0.38 0.4 0.42 0.44 0.46 0.48 0.5 0.52 0.54 0.56 0.58 0.6 0.62 0.64 0.66 0.68 0.7 0.72 0.7
2  4 0.76 0.78 0.8 0.82 0.84 0.86 0.88 0.9 0.92 0.94 0.96 0.98 1 1.02 1.04 1.06 1.08 1.1 1.12 1.14 1.16 1
3  0 0 0 0 0 0 1 4 2 2 7 14 15 18 19 31 35 52 50 48 67 60 70 73 63 55 65 50 44 29 27 36 12 18 12 7 4 4
4  1 2 2 0.05 0 0 0 0 0 0 0.05 0.2 0.1 0.1 0.35 0.7 0.75 0.9 0.95 1.55 1.75 2.6 2.5 2.4 3.35 3 3.5 3.65
5  3.15 2.75 3.25 2.5 2.2 1.45 1.35 1.8 0.6 0.9 0.6 0.35 0.2 0.2 0.05 0.1 0.1 0.31 0.33 0.35 0.37 0.39 0
6  .41 0.43 0.45 0.47 0.49 0.51 0.53 0.55 0.57 0.59 0.61 0.63 0.65 0.67 0.69 0.71 0.73 0.75 0.77 0.79 0.81
7  0.83 0.85 0.87 0.89 0.91 0.93 0.95 0.97 0.99 1.01 1.03 1.05 1.07 1.09 1.11 1.13 1.15 mns[, 3] 1
8  1
9

```

Running

Hist: Ln: 9 Col: 6 Ins Classic Num EN_US



R-Plot of Beta Distribution

The beta distribution has two parameters α and β which provide flexibility to specify an asymmetrical probability distribution with mean $1+1\div\beta/\alpha$ in APL operation order. The 'BetaDistChart' `rbeta(n,  $\alpha$ ,  $\beta$ )` function to generate n sample values of a beta-distributed random variable and displays the plot in APL64.

BetaDistChart X

⌕ APLNext 20240422

⌕ n: #vals

⌕ a: alpha

⌕ b: beta

(n a b)←X

:TRY *

chartFnm←«chartFnm.bmp»

☐ NFE 'Delete' chartFnm

☐ RNET 'SetSymbol' 'chartFnm' chartFnm

☐ RNET 'Evaluate' 'bmp(chartFnm)'

☐ RNET 'SetSymbol' 'n' n

☐ RNET 'SetSymbol' 'a' a

☐ RNET 'SetSymbol' 'b' b

☐ RNET 'Evaluate' 'b = `rbeta(n,a,b)`'

☐ RNET 'Evaluate' 'hist(b, prob = TRUE, breaks=50, col = "darkred")'

☐ RNET 'Evaluate' 'dev.off()'

F←'F'☐wi 'Create' 'Form' ('size' 40 80) ('caption' 'R-plot in APL64')

P←'F.P'☐wi 'Create' 'Picture' ('where' 0 0 40 80) ('size' 30 60) ('style' 0)

P☐wi 'bitmap' 'chartFnm.bmp'

sink←F☐wi 'Wait'

```
:CATCH ' BetaDistChart failed: ',,dm
:ENDTRY
```

The screenshot shows the APL64: CLEAR WS window. The script in the editor is as follows:

```

0 BetaDistChart X
1 A APLNext 20240422
2 A n: #vals
3 A a: alpha
4 A b: beta
5 (n a b)→X
6 ⍎TRY *
7 chartFnm←«chartFnm.bmp»
8 ⍎NFE 'Delete' chartFnm
9 ⍎RNET 'SetSymbol' 'chartFnm' chartFnm
10 ⍎RNET 'Evaluate' 'bmp(chartFnm)'
11 ⍎RNET 'SetSymbol' 'n' n
12 ⍎RNET 'SetSymbol' 'a' a
13 ⍎RNET 'SetSymbol' 'b' b
14 ⍎RNET 'Evaluate' 'b = rbeta(n,a,b)'
15 ⍎RNET 'Evaluate' 'hist(b, prob = TRUE, breaks=50, col = "darkred")'
16 ⍎RNET 'Evaluate' 'dev.off()'
17 F←'F'⍎wi 'Create' 'Form' ('size' 40 80) ('caption' 'R-plot in APL64')
18 P←'F.P'⍎wi 'Create' 'Picture' ('where' 0 0 40 80) ('size' 30 60) ('style' 0)
19 P⍎wi 'bitmap' 'chartFnm.bmp'
20 sink←F⍎wi 'Wait'
21 ⍎CATCH ' BetaDistChart failed: ',,dm
22 ⍎ENDTRY
23

```

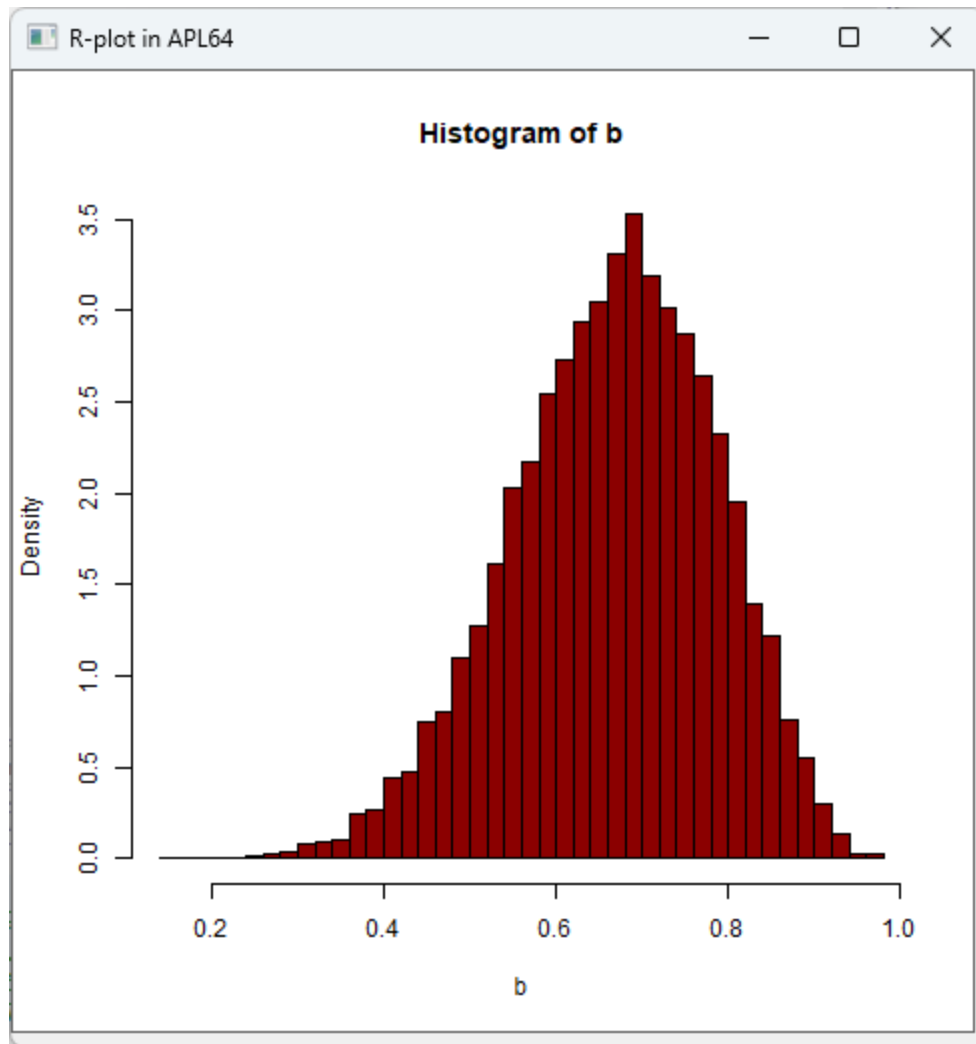
The output window shows the following data:

```

0 BetaDistChart 10000 10 5
1 0.14 0.16 0.18 0.2 0.22 0.24 0.26 0.28 0.3 0.32 0.34 0.36 0.38 0.4 0.42 0.44 0.46 0.48
2 6 0.88 0.9 0.92 0.94 0.96 0.98 1 1 1 0 1 3 5 8 15 19 20 49 52 87 95 149 160 219
3 0.005 0.005 0.005 0 0.005 0.015 0.025 0.04 0.075 0.095 0.1 0.245 0.26 0.435 0.47
4 .645 2.32 1.95 1.39 1.22 0.76 0.555 0.3 0.13 0.025 0.02 0.15 0.17 0.19 0.21 0.23
5 0.61 0.63 0.65 0.67 0.69 0.71 0.73 0.75 0.77 0.79 0.81 0.83 0.85 0.87 0.89 0.91
6 1
7

```

The status bar at the bottom indicates 'Ready' and 'Cmd: Ln: 0 Col: 0 Ins Classic Num EN_US'.



Statistical Values

- ☐ RNET 'Set' 'dV' (11 12 13 14 15 11.2 11 13)
- ☐ RNET 'Eval' 'v1=mean(dV)'
- ☐ RNET 'Get' 'v1'
- ☐ RNET 'Eval' 'v2=sd(dV)'
- ☐ RNET 'Get' 'v2'
- ☐ RNET 'Eval' 'v3=sum(dV)'
- ☐ RNET 'Get' 'v3'
- ☐ RNET 'Eval' 'v4=length(dV)'
- ☐ RNET 'Get' 'v4'
- ☐ RNET 'Eval' 'v5=sort(dV)'
- ☐ RNET 'Get' 'v5'
- ☐ RNET 'Eval' 'v6=unique(dV)'
- ☐ RNET 'Get' 'v6'
- ☐ RNET 'Eval' 'v7=max(dV)'

- ☐ RNET 'Get' 'v7'
- ☐ RNET 'Eval' 'v8=min(dV)'
- ☐ RNET 'Get' 'v8'
- ☐ RNET 'Eval' 'v9=median(dV)'
- ☐ RNET 'Get' 'v9'
- ☐ RNET 'Eval' 'v10=var(dV)'
- ☐ RNET 'Get' 'v10'
- ☐ RNET 'Eval' 'v11=quantile(dV)'
- ☐ RNET 'Get' 'v11'

- ☐ RNET 'Eval' 'anovaM = anova(model)'
- ☐ RNET 'Eval' 'v1=coef(model)'
- ☐ RNET 'Get' 'v1'
- ☐ RNET 'Eval' 'v2=confint(model)'
- ☐ RNET 'Get' 'v2'
- ☐ RNET 'Eval' 'v3=deviance(model)'
- ☐ RNET 'Get' 'v3'
- ☐ RNET 'Eval' 'v4=vcov(model)'
- ☐ RNET 'Get' 'v4'

Results may vary because the example uses normal distribution samples.

```

0  ☐ RNET 'Set' 'x' (1100)
1  ☐ RNET 'Eval' 'y = (x*2)+rnorm(x)/10'
2  ☐ RNET 'Eval' 'model <- lm(y~x)'
3  ☐ RNET 'Eval' 'anovaM = anova(model)'
4  ☐ RNET 'Eval' 'v1=coef(model)'
5  ☐ RNET 'Get' 'v1'
6  0.01475583972 1.99996412
7  ☐ RNET 'Eval' 'v2=confint(model)'
8  ☐ RNET 'Get' 'v2'
9  -0.02498354947 0.05449522892
10 1.999280935 2.000647305
11 ☐ RNET 'Eval' 'v3=deviance(model)'
12 ☐ RNET 'Get' 'v3'
13 0.9678089428
14 ☐ RNET 'Eval' 'v4=vcov(model)'
15 ☐ RNET 'Get' 'v4'
16 0.0004010092713 -5.985213004E-6
17 -0.000005985213004 1.185190694E-7
18

```

Getting Information about the R Installation under Program Control

Check the R version being used by ☐ RNET.

- ☐ RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin\x64'
- ☐ RNET 'Get' "R.version.string"

```

☐ RNET 'Get' 'R.home()'
☐ RNET 'Get' '.libPaths()'
☐ RNET 'Get' 'getwd()'
☒ RNET 'Get' 'Sys.info()'

```

```

0      RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin\x64'
1      RNET 'Get' "R.version.string"
2      R version 4.5.1 (2025-06-13 ucrt)
3      RNET 'Get' 'R.home()'
4      C:/PROGRA~1/R/R-45~1.1
5      RNET 'Get' '.libPaths()'
6      C:/Program Files/R/R-4.5.1/library
7      RNET 'Get' 'getwd()'
8      C:/Users/joebl/source/repos/APLNowLLC/APL64/NetCore/bin/Debug/net9.0
9      RNET 'Get' 'Sys.info()'
10     Windows
11     >= 8 x64
12     build 9200
13     5760-68TCHL3
14     x86-64
15     joebl
16     joebl
17     joebl
18     5760-68TCHL3
19     RNET 'Get' '.Platform$OS.type'
20     windows
21     |

```

```

☐ RNET 'InitR' 'C:\Program Files\R\R-4.5.1'
☐ RNET 'Eval' 'v1=Sys.getlocale()'
☐ RNET 'Get' 'v1'

```

```

0      RNET 'InitR' 'C:\Program Files\R\R-4.5.1'
1      RNET 'Eval' 'v1=Sys.getlocale()'
2      RNET 'Get' 'v1'
3      LC_COLLATE=English_United States.1252;LC_CTYPE=English_United States.1252;LC_MONETARY=English_United States.1252;LC_NUMERIC=C;LC_TIME=English_United States.1252
4      |

```

R Packages Installed on the Workstation

☐ RNET 'Eval' "v1=installed.packages()[,c('Package','Version')]"
☐ RNET 'Get' 'v1'

APL64: CLEAR WS

File Edit Session Objects Tools Options Help

0 ☐RNET 'Eval' "v1=installed.packages()[,c('Package','Version')]"

1 ☐RNET 'Get' 'v1'

2	base	4.5.1
3	boot	1.3-31
4	class	7.3-23
5	cli	3.6.5
6	cluster	2.1.8.1
7	codetools	0.2-20
8	compiler	4.5.1
9	curl	7.0.0
10	datasets	4.5.1
11	farver	2.1.2
12	foreign	0.8-90
13	ggplot2	4.0.0
14	glue	1.8.0
15	graphics	4.5.1
16	grDevices	4.5.1
17	grid	4.5.1
18	gtable	0.3.6
19	isoband	0.2.7
20	KernSmooth	2.23-26
21	labeling	0.4.3
22	lattice	0.22-7
23	lifecycle	1.0.4
24	magick	2.9.0
25	magrittr	2.0.4
26	MASS	7.3-65
27	Matrix	1.7-3
28	methods	4.5.1
29	mgcv	1.9-3
30	nlme	3.1-168
31	nnet	7.3-20
32	parallel	4.5.1
33	R6	2.6.1
34	RColorBrewer	1.1-3
35	Rcpp	1.1.0
36	rlang	1.1.6
37	rpart	4.1.24
38	S7	0.2.0
39	scales	1.4.0
40	spatial	7.3-18
41	splines	4.5.1
42	stats	4.5.1
43	stats4	4.5.1
44	survival	3.8-3
45	tcltk	4.5.1
46	tools	4.5.1
47	translations	4.5.1
48	utils	4.5.1
49	vctrs	0.6.5
50	viridisLite	0.4.2
51	withr	3.0.2
52		

Ready | Hist: Ln: 52 Col: 6 | Ins | Classic | Num | EN_US

R may not throw an Exception

R may not throw an exception for results which are apparently inappropriate. It is up to the APL64 programmer to validate APL64 input sent to the R environment for processing.

Example: Create this function and run it:

```
Z←Sqrt X
⎕RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin\x64'
:if X<0
  'Warning: Input is negative, check results!'
:endif
Z←⎕RNET 'Get' '(-1)^.5'
```

The screenshot shows the APL64: CLEAR WS application window. The top menu bar includes File, Edit, Session, Objects, Tools, Options, and Help. Below the menu is a toolbar with various icons. The main editing area displays the following code:

```
0 Z←Sqrt X
1 ⎕RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin\x64'
2 :if X<0
3   'Warning: Input is negative, check results!'
4 :endif
5 Z←⎕RNET 'Get' '(-1)^.5'
```

Below the code editor is a status bar with the text [0:0] and buttons for Commit Changes and Commit & Close. The bottom panel shows the execution results:

```
0 )ed ▾
1 Sqrt -1
2 Warning: Input is negative, check results!
3 0.0E298
4
```

The bottom status bar indicates the application is Ready and shows the history: Hist: Ln: 4 Col: 6 Ins Classic Num EN_US.

Resources

What is R?

[R is a statistics and graphics toolkit](#). Its origin is the S language developed at Bell Laboratories. R provides a wide variety of statistics and graphics tools. R is available as no-cost, open-source software under the terms of the Free Software Foundation's GNU General Public License. R is in wide use by many enterprises both public and private for scientific, research and commercial purposes on a worldwide basis, so it incorporates tested tools for statistics and graphics. R is not implemented in .Net, so it is 'unmanaged' code.

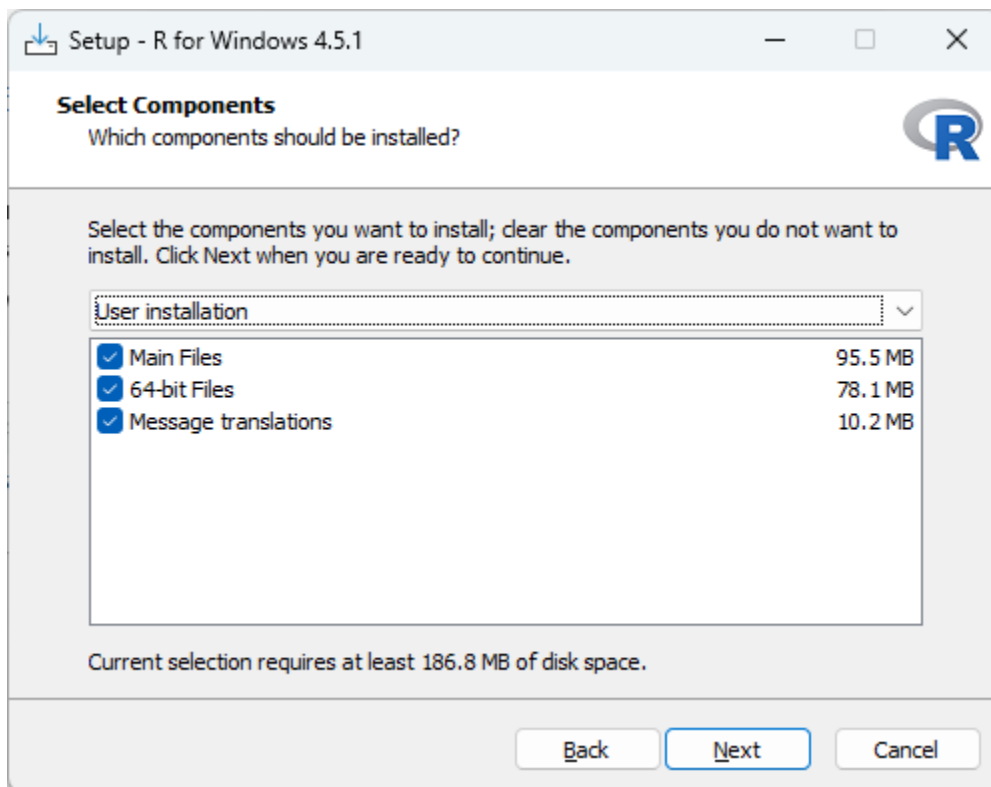
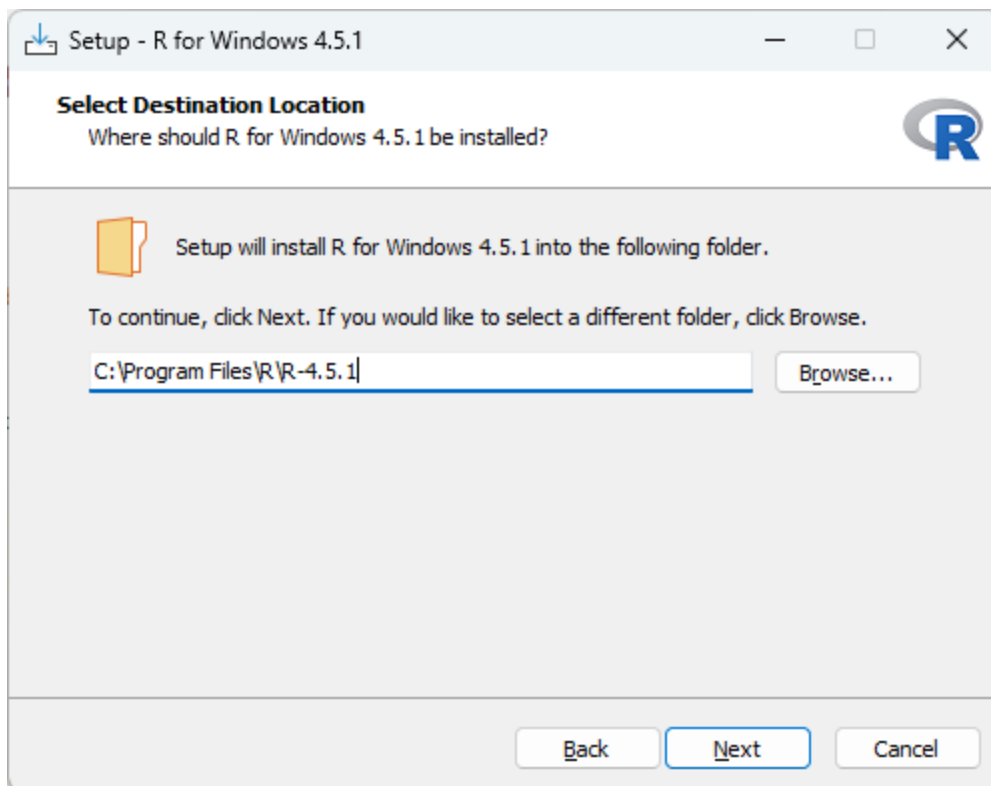
The R-Project

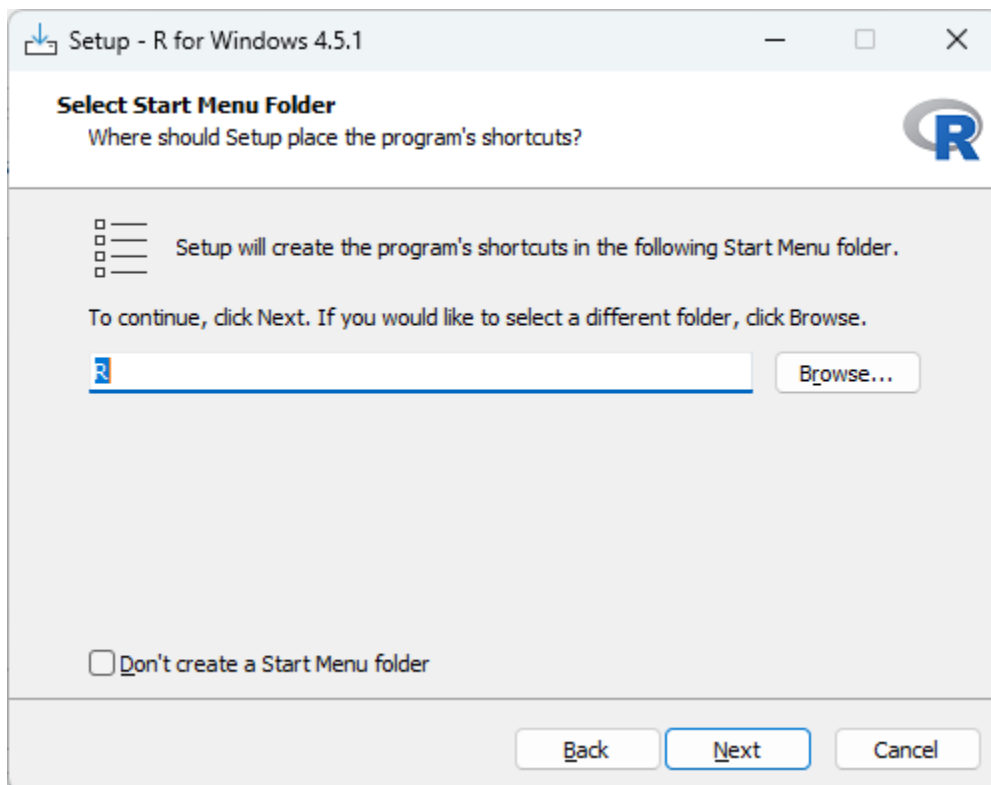
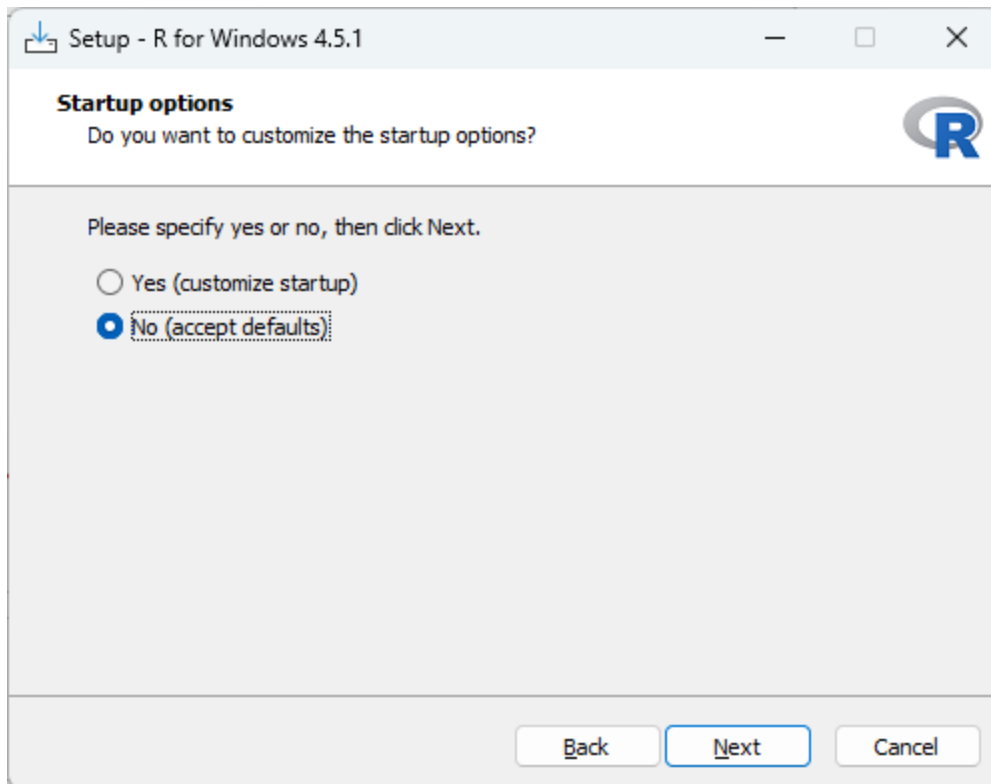
The [R-project](#) provides many useful resources.

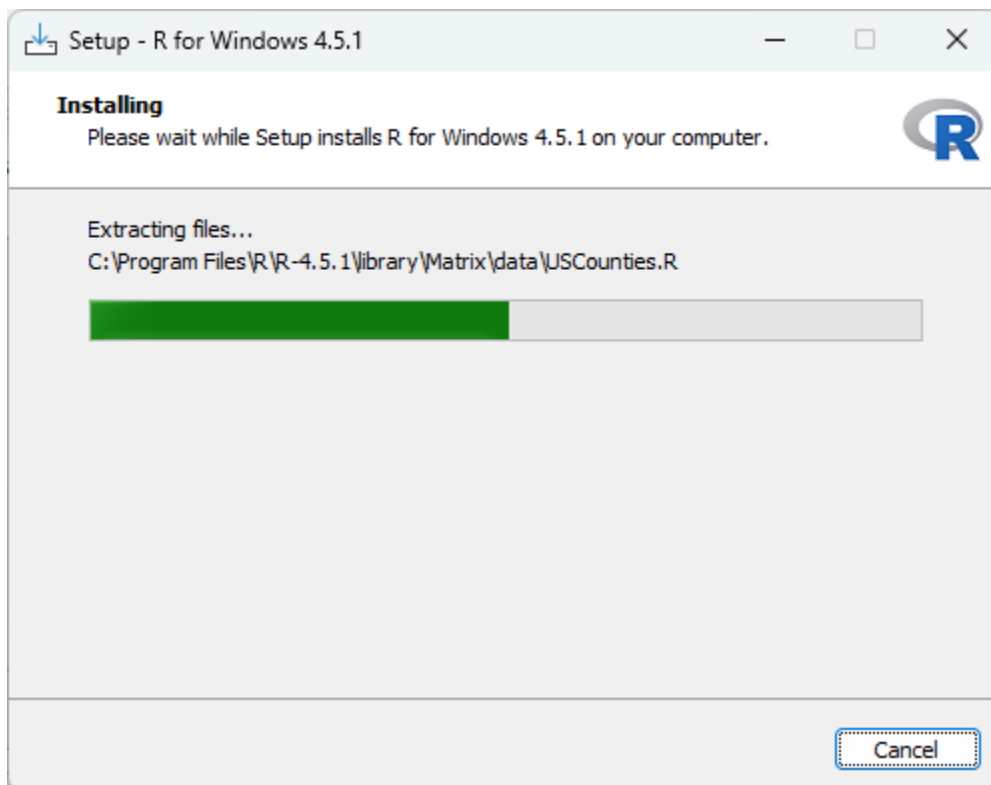
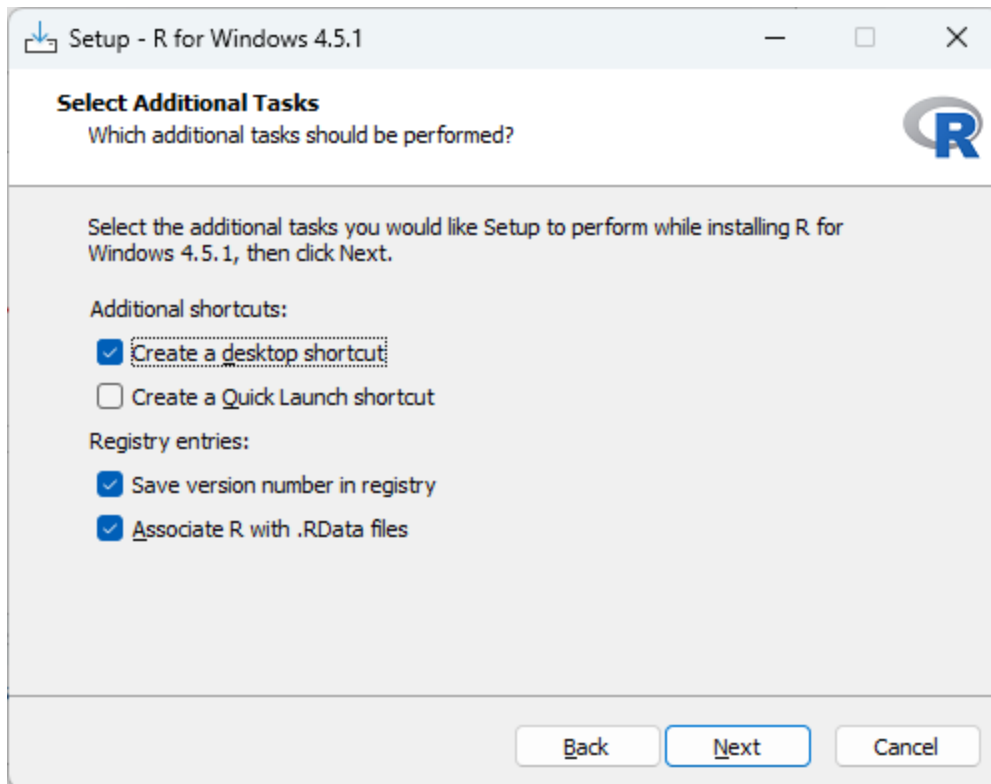
Installing R

The R license requires that end users install the product, so it is not included in APL64. The R toolkit should be installed to the target workstation. The R toolkit is cross-platform, however the appropriate version of R for the target operating system must be installed. The R installer defaults are sufficient. It may be necessary to run R-installer with elevated credentials, for example, in Windows 'run as administrator'.

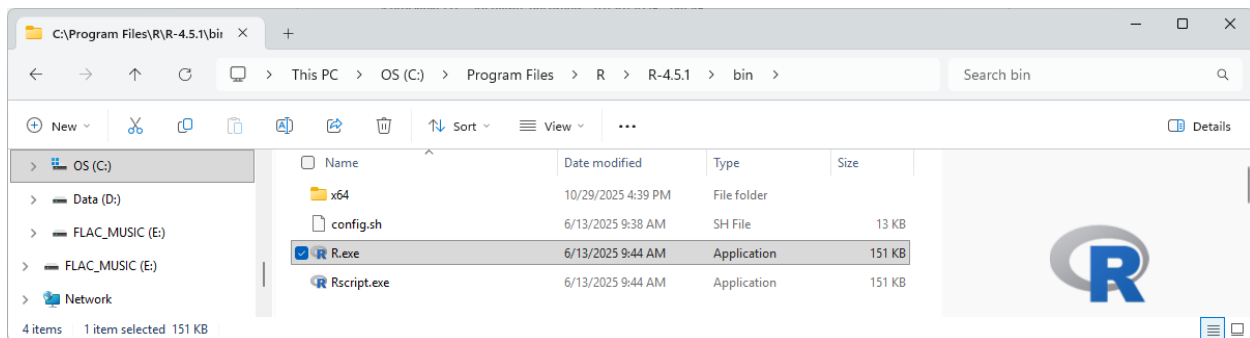
For Windows get the installer [here](#).



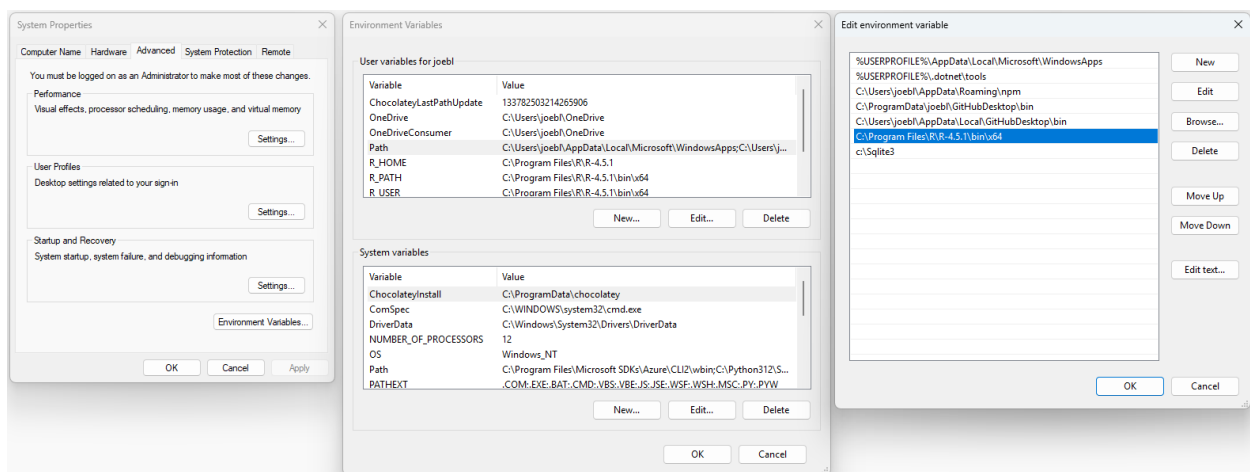




Using Windows Explorer, find the path of R.exe on the target workstation. This path will be needed when using the ☐ RNET 'InitR' installationPathOfR action:



In Windows it may be necessary to create or modify the environment variables to provide the path to the R components. The Windows Path environment variable should contain the path to the R components:



Compatibility with APL+Win

The RNET implementation in APL64 is simplified, is cross-platform, has improved performance, and has new features not available in APL+Win. RNET does not require the use of the APL64 CSE system function, or deprecated ActiveX technology. Because RNET is designed to be cross-platform, images created by RNET are necessarily file-based as there is no universal platform to display images in an application. R-images, such as those create by Plot or GGPlot2 can be included within a WI form.

R-Tutorials

This document is not an R-tutorial. [Here](#) is a good R-tutorial as well as this [one](#).

R.Net

[R.NET](#) is a .Net interface to R used to implement RNET.