PROC IML #1
SAS Interactive Matrix Language, An Introduction

JC Wang

Western Michigan University
Department of Statistics
Outline

1. Introduction to PROC IML
   - Interactive Matrix Language

2. Creating Matrices and Matrix Operations
   - Defining/Creating Matrices
   - Operators and Matrix Operations
   - Subscripting

3. Selected Statements, Functions, and Call Routines
   - Utility Statements
What is IML

- SAS/IML is an interactive programming language. I=Interactive, M=Matrix, L=Language. Matrices are the fundamental objects.
- The programming language is dynamic, interactive because necessary activities such as memory allocation and dimensioning of matrices are performed automatically.
What is IML

- SAS/IML is an interactive programming language. I=Interactive, M=Matrix, L=Language. Matrices are the fundamental objects.
- The programming language is dynamic, interactive because necessary activities such as memory allocation and dimensioning of matrices are performed automatically.
What is IML

- SAS/IML is an interactive programming language. I=Interactive, M=Matrix, L=Language. Matrices are the fundamental objects.
- The programming language is dynamic, interactive because necessary activities such as memory allocation and dimensioning of matrices are performed automatically.
Characteristics of IML

IML

- operates on matrices (numeric or character);
- possesses powerful operators, functions, and call routines;
- is interactive;
- allows dynamic allocation of storage for matrices, modules;
- allows access to multiple files all at once;
- provides full access to SAS data files;
- produces graphics.
Outline

1. Introduction to PROC IML
   - Interactive Matrix Language

2. Creating Matrices and Matrix Operations
   - Defining/Creating Matrices
   - Operators and Matrix Operations
   - Subscripting

3. Selected Statements, Functions, and Call Routines
   - Utility Statements
Defining Matrix

- Matrices can be either numeric (stored as double precision) or character (of same length up to 32,767 bytes).
- Matrices are stored in named objects (SAS names) by using assignment operator =.
- Matrices are defined row-wise (elements separated by spaces, and rows separated by commas) and are enclosed in braces {}.
- Character elements, when enclosed in matching quotes, are case sensitive. Otherwise, all characters are converted to upper case.
Defining Matrix

- Matrices can be either numeric (stored as double precision) or character (of same length up to 32,767 bytes).
- Matrices are stored in named objects (SAS names) by using assignment operator =.
- Matrices are defined row-wise (elements separated by spaces, and rows separated by commas) and are enclosed in braces {}.
- Character elements, when enclosed in matching quotes, are case sensitive. Otherwise, all characters are converted to upper case.
Defining Matrix

- Matrices can be either numeric (stored as double precision) or character (of same length up to 32,767 bytes).
- Matrices are stored in named objects (SAS names) by using assignment operator `=`.
- Matrices are defined row-wise (elements separated by spaces, and rows separated by commas) and are enclosed in braces `{}`.
- Character elements, when enclosed in matching quotes, are case sensitive. Otherwise, all characters are converted to upper case.
Defining Matrix

- Matrices can be either numeric (stored as double precision) or character (of same length up to 32,767 bytes).
- Matrices are stored in named objects (SAS names) by using assignment operator =.
- Matrices are defined row-wise (elements separated by spaces, and rows separated by commas) and are enclosed in braces {}.
- Character elements, when enclosed in matching quotes, are case sensitive. Otherwise, all characters are converted to upper case.
Defining Matrix

- Matrices can be either numeric (stored as double precision) or character (of same length up to 32,767 bytes).
- Matrices are stored in named objects (SAS names) by using assignment operator =.
- Matrices are defined row-wise (elements separated by spaces, and rows separated by commas) and are enclosed in braces {}.
- Character elements, when enclosed in matching quotes, are case sensitive. Otherwise, all characters are converted to upper case.
Matrix Operations

Element-wise Operations

- **arithmetics**  
  - (prefix, reversing sign)  
  - +  
  - -  
  - #  
  - ##  
  - /  
  Note that + operator also allows character operands for string concatenation operation which works exactly as CONCAT function. Missing value results when at least one operand has missing value.

- **comparisons**  
  - >  
  - <  
  - =  
  - >=  
  - <=  
  - ^=  
  Missing element is the smallest in comparison.

- **logical**  
  - ^ (prefix)  
  - &  
  - |  

- **element-wise extrema**  
  - <> (element-wise maxima)  
  - >< (element-wise minima)
Matrix Operations

Element-wise Operations

- **arithmetics**  
  \( - \) (prefix, reversing sign)  
  \(+\)  
  \(-\)  
  \(#\)  
  \(#\#\)  
  \(/\)  
  Note that the + operator also allows character operands for string concatenation operation which works exactly as CONCAT function. Missing value results when at least one operand has missing value.

- **comparisons**  
  \( > \)  
  \(< \)  
  \( = \)  
  \( >= \)  
  \( <= \)  
  \( ^= \)  
  Missing element is the smallest in comparison.

- **logical**  
  \( ^\) (prefix)  
  \&  
  \( |\)  

- **element-wise extrema**  
  \( <> \) (element-wise maxima)  
  \( <\rangle \) (element-wise minima)
Matrix Operations

Element-wise Operations

- **arithmetics**  
  - (prefix, reversing sign)  
  - +  
  - −  
  - #  
  - ##  
  - /  
  
  Note that + operator also allows character operands for string concatenation operation which works exactly as CONCAT function. Missing value results when at least one operand has missing value.

- **comparisons**  
  - >  
  - <  
  - =  
  - >=  
  - <=  
  - ^=  
  
  Missing element is the smallest in comparison.

- **logical**  
  - ^ (prefix)  
  - &  
  - |  

- **element-wise extrema**  
  - <> (element-wise maxima)  
  - <<< (element-wise minima)
Matrix Operations

Element-wise Operations

**arithmetics**  
− (prefix, reversing sign)  
+  
−  
#  
###  
/  

Note that + operator also allows character operands for string concatenation operation which works exactly as CONCAT function. Missing value results when at least one operand has missing value.

**comparisons**  
>  
<  
=  
>=  
<=  
^=  

Missing element is the smallest in comparison.

**logical**  
^ (prefix)  
&  
|  

**element-wise extrema**  
<> (element-wise maxima)  
<<  
(element-wise minima)
Matrix Operations

Element-wise Operations

- **arithmetics**  
  - (prefix, reversing sign)  
  +  -  #  ##  /  
  Note that + operator also allows character operands for string concatenation operation which works exactly as CONCAT function. Missing value results when at least one operand has missing value.

- **comparisons**  
  >  <  =  >=  <=  ^=  
  Missing element is the smallest in comparison.

- **logical**  
  ^ (prefix)  &  |  

- **element-wise extrema**  
  <> (element-wise maxima)  >> (element-wise minima)
Matrix Operations

- **algebraic** * (matrix product)  ** (matrix power)  ' (back quote, transpose)  @ (direct product or Kronecker product)

- **concatenation** || (binds columns)  // (binds rows)

- **special operator** : (index creation operator) creates row vector
  value1:value2 (numeric) or ‘prefix1’:'prefixn' (character)

Use DO function for numeric vectors with increments other than 1 or −1.
Matrix Operations

Matrix Operations

- **algebraic**  
  - `*` (matrix product)  
  - `**` (matrix power)  
  - `'` (back quote, transpose)  
  - `@` (direct product or Kronecker product)

- **concatenation**  
  - `||` (binds columns)  
  - `//` (binds rows)

- **special operator**  
  - `:` (index creation operator) creates row vector
  - `value1:value2` (numeric) or `’prefix1’:’prefixn’` (character)

Use **DO** function for numeric vectors with increments other than 1 or −1.
Matrix Operations

Matrix Operations

- **algebraic**  * (matrix product)  ** (matrix power)  ‘ (back quote, transpose)  @ (direct product or Kronecker product)
- **concatenation**  || (binds columns)  // (binds rows)
- **special operator**  : (index creation operator) creates row vector
  value1:value2 (numeric) or ’prefix1’:'prefixn’ (character)
  Use **DO** function for numeric vectors with increments other than 1 or −1.
Matrix Operations

Matrix Operations

- **algebraic**  * (matrix product)  ** (matrix power)  ‘ (back quote, transpose)  @ (direct product or Kronecker product)

- **concatenation**  || (binds columns)  // (binds rows)

- **special operator**  : (index creation operator) creates row vector

  value1:value2 (numeric) or ‘prefix1’:’prefixn’ (character)

Use **DO** function for numeric vectors with increments other than 1 or −1.
Subscripting

Subscript

\[ \text{operand}[\text{rows}, \text{cols}] \quad \text{operand}[\text{elements}] \quad \text{where} \]

- **operand**: usually a matrix name, but it can also be an expression or literal (i.e., constant).
- **rows**: an expression, either scalar or vector, for selecting one or more rows from the operand.
- **cols**: an expression, either scalar or vector, for selecting one or more columns from the operand.

**Note:**

- rows and cols can be numeric or character (names).
- subscripted matrix can appear on the left of an assignment statement.
Subscripting

Operand

\[ \text{operand}[\text{rows}, \text{cols}] \quad \text{operand}[\text{elements}] \quad \text{where} \]

- \text{operand}: usually a matrix name, but it can also be an expression or literal (i.e., constant).
- \text{rows}: an expression, either scalar or vector, for selecting one or more rows from the operand.
- \text{cols}: an expression, either scalar or vector, for selecting one or more columns from the operand.

Note:

- \text{rows} and \text{cols} can be numeric or character (names).
- subscripted matrix can appear on the left of an assignment statement.
Subscripting

Subscript

\( \text{operand}[\text{rows}, \text{cols}] \text{ operand}[\text{elements}] \text{ where} \)

- \( \text{operand} \): usually a matrix name, but it can also be an expression or literal (i.e., constant).

- \( \text{rows} \): an expression, either scalar or vector, for selecting one or more rows from the operand.

- \( \text{cols} \): an expression, either scalar or vector, for selecting one or more columns from the operand.

Note:

- \( \text{rows} \) and \( \text{cols} \) can be numeric or character (names)
- subscripted matrix can appear on the left of an assignment statement
Subscripting

Subscript

\[
\text{operand}[\text{rows}, \text{cols}] \quad \text{operand}[\text{elements}] \quad \text{where}
\]

- **operand**: usually a matrix name, but it can also be an expression or literal (i.e., constant).
- **rows**: an expression, either scalar or vector, for selecting one or more rows from the operand.
- **cols**: an expression, either scalar or vector, for selecting one or more columns from the operand.

Note:

- rows and cols can be numeric or character (names)
- subscripted matrix can appear on the left of an assignment statement
Subscripting

Subscript

\[ \text{operand}[\text{rows, cols}] \quad \text{operand}[\text{elements}] \quad \text{where} \]

- **operand**: usually a matrix name, but it can also be an expression or literal (i.e., constant).
- **rows**: an expression, either scalar or vector, for selecting one or more rows from the operand.
- **cols**: an expression, either scalar or vector, for selecting one or more columns from the operand.

**Note:**

1. **rows and cols** can be numeric or character (names)
2. subscripted matrix can appear on the left of an assignment statement
Subscripting
Subscript

\texttt{operand[rows, cols]} \texttt{operand[elements]} \texttt{where}

- \texttt{operand}: usually a matrix name, but it can also be an expression or literal (i.e., constant).
- \texttt{rows}: an expression, either scalar or vector, for selecting one or more rows from the operand.
- \texttt{cols}: an expression, either scalar or vector, for selecting one or more columns from the operand.

Note:

1. \texttt{rows} and \texttt{cols} can be numeric or character (names)
2. subscripted matrix can appear on the left of an assignment statement
Subscripting

Subscript reduction

- row reduction: `operand[operator, ]`
- column reduction: `operand[, operator]`
- row (first) and column (second) reduction: `operand[operator, operator]`
- entire matrix reduction: `operand[operator]`

Available reduction operators are `+ # ## : <> >:< <::> >:<`. Can combine subscripting with subscript reduction. Unless all missing, missing values are excluded from the reduction operation.

See examples `ProcIML01.sas–ProcIML08.sas`. 
Subscripting

Subscript reduction

- **row reduction**: `operand[operator,]`
- **column reduction**: `operand[,operator]`
- **row (first) and column (second) reduction**: `operand[operator,operator]`
- **entire matrix reduction**: `operand[operator]`

Available reduction operators are `+`, `#`, `##`, `:<`, `><`, `<>`, `:<:`, `:<<`. Can combine subscripting with subscript reduction. Unless all missing, missing values are excluded from the reduction operation. See examples `ProcIML01.sas–ProcIML08.sas`.
Subscripting
Subscript reduction

- **row reduction**: `operand[operator, ]`
- **column reduction**: `operand[, operator]`
- **row (first) and column (second) reduction**: `operand[operator, operator]`
- **entire matrix reduction**: `operand[operator]`

Available reduction operators are `+ # ## : <> >< <::> :<`. Can combine subscripting with subscript reduction. Unless all missing, missing values are excluded from the reduction operation. See examples `ProcIML01.sas–ProcIML08.sas`.
Subscripting

Subscript reduction

- **row reduction**: `operand[operator,]`
- **column reduction**: `operand[,operator]`
- **row (first) and column (second) reduction**: `operand[operator,operator]`
- **entire matrix reduction**: `operand[operator]`

Available reduction operators are `+ # ## : <> >< <::> >:<`. Can combine subscripting with subscript reduction. Unless all missing, missing values are excluded from the reduction operation.

See examples `ProcIML01.sas–ProcIML08.sas`.
Subscripting
Subscript reduction

- row reduction: \(\text{operand}[\text{operator},]\)
- column reduction: \(\text{operand}[\text{operator}]\)
- row (first) and column (second) reduction: \(\text{operand}[, \text{operator}, \text{operator}]\)
- entire matrix reduction: \(\text{operand}[\text{operator}]\)

Available reduction operators are \(+\) \(#\) \(##\) : \(<<>\) \(><<\) \(<:<\) \(>:<\). Can combine subscripting with subscript reduction. Unless all missing, missing values are excluded from the reduction operation.

See examples ProcIML01.sas–ProcIML08.sas.
Subscripting

Subscript reduction

- **row reduction**: `operand[operator,]`
- **column reduction**: `operand[,operator]`
- **row (first) and column (second) reduction**:
  
  `operand[operator,operator]`
- **entire matrix reduction**: `operand[operator]`

Available reduction operators are `+ # ## : <> >> <= :<<`. Can combine subscripting with subscript reduction. Unless all missing, missing values are excluded from the reduction operation.

See examples `ProcIML01.sas–ProcIML08.sas`. 
Outline

1. Introduction to PROC IML
   - Interactive Matrix Language

2. Creating Matrices and Matrix Operations
   - Defining/Creating Matrices
   - Operators and Matrix Operations
   - Subscripting

3. Selected Statements, Functions, and Call Routines
   - Utility Statements
RESET Statement

to set processing options

RESET <options>;

Some options (defaults first)

- **overall printing**: CENTER|NOCENTER, NOPRINT|PRINT, NOPRINTALL|PRINTALL, LINESIZE=n(default=78), PAGESIZE=n(default=21), FW=n(default=9), SPACES=n(default=4)

- **printing matrices**: NAME|NONAME, NOAUTONAME|AUTONAME, NOFUZZ|FUZZ,

- **printing destination**: NOLOG|LOG

Use `SHOW OPTIONS;` statement to show options currently in effect
RESET Statement

to set processing options

RESET <options>;

Some options (defaults first)

- **overall printing**: CENTER|NOCENTER, NOPRINT|PRINT, NOPRINTALL|PRINTALL, LINESIZE=n (default=78), PAGESIZE=n (default=21), FW=n (default=9), SPACES=n (default=4)

- **printing matrices**: NAME|NONAME, NOAUTONAME|AUTONAME, NOFUZZ|FUZZ,

- **printing destination**: NOLOG|LOG

Use SHOW OPTIONS; statement to show options currently in effect.
RESET Statement

to set processing options

RESET <options>;

Some options (defaults first)

- **overall printing**: CENTER|NOCENTER, NOPRINT|PRINT, NOPRINTALL|PRINTALL, LINESIZE=n(default=78), PAGESIZE=n(default=21), FW=n(default=9), SPACES=n(default=4)

- **printing matrices**: NAME|NONAME, NOAUTONAME|AUTONAME, NOFUZZ|FUZZ,

- **printing destination**: NOLOG|LOG

Use `SHOW OPTIONS;` statement to show options currently in effect
RESET Statement

to set processing options

RESET <options>;

Some options (defaults first)

- **overall printing**: CENTER|NOCENTER, NOPRINT|PRINT, NORPRINTALL|PRINTALL, LINESIZE=n(default=78), PAGESIZE=n(default=21), FW=n(default=9), SPACES=n(default=4)

- **printing matrices**: NAME|NONAME, NOAUTONAME|AUTONAME, NOFUZZ|FUZZ,

- **printing destination**: NOLOG|LOG

Use SHOW OPTIONS; statement to show options currently in effect
RESET Statement

**to set processing options**

```latex
RESET <options>;
```

Some *options* (defaults first)

- **overall printing**: CENTER|NOCENTER, NOPRINT|PRINT, 
  NOPRINTALL|PRINTALL, LINESIZE=\(n\) (default=78), 
  PAGESIZE=\(n\) (default=21), FW=\(n\) (default=9), 
  SPACES=\(n\) (default=4)

- **printing matrices**: NAME|NONAME, 
  NOAUTONAME|AUTONAME, NOFUZZ|FUZZ,

- **printing destination**: NOLOG|LOG

Use **SHOW OPTIONS**; statement to show options currently in effect
PRINT Statement

PRINT <matrices> <(expression)> <"message"> <pointer-controls> <[options]>;

where (in any sensible order)

- **matrices**: names of matrices
- **(expression)**: expression in parentheses that is evaluated
- **"message"**: message in matching quotes
- **pointer-controls**: control the pointer for printing—using a comma (,) to skip a single line and using a slash (/) to skip to a new page.
- **[options]**: options enclosed in square brackets and separated by blanks—COLNAME=matrix, ROWNAME=matrix, FORMAT=format, LABEL=label
PRINT Statement

PRINT <matrices> <(expression)> <"message"> <pointer-controls> <[options]>;

where (in any sensible order)

- **matrices**: names of matrices
- **(expression)**: expression in parentheses that is evaluated
- **"message"**: message in matching quotes
- **pointer-controls**: control the pointer for printing—using a comma (,) to skip a single line and using a slash (/) to skip to a new page.
- **[options]**: options enclosed in square brackets and separated by blanks—COLNAME=matrix, ROWNAME=matrix, FORMAT=format, LABEL=label
PRINT Statement

\[
\text{PRINT} \ <\text{matrices}> \ <{(expression)}> \ <"message">\ \\
<\text{pointer-controls}> \ <[\text{options}]>; \\
\]

where (in any sensible order)

- **matrices**: names of matrices
- **(expression)**: expression in parentheses that is evaluated
- **"message"**: message in matching quotes
- **pointer-controls**: control the pointer for printing—using a comma (,) to skip a single line and using a slash (/) to skip to a new page.
- **[options]**: options enclosed in square brackets and separated by blanks— COLNAME=matrix, ROWNAME=matrix, FORMAT=format, LABEL=label
**PRINT Statement**

```
PRINT <matrices> <(expression)> <"message"> <pointer-controls> <[options]>;
```

where (in any sensible order)

- **matrices**: names of matrices
- **(expression)**: expression in parentheses that is evaluated
- **"message"**: message in matching quotes
- **pointer-controls**: control the pointer for printing—using a comma (,) to skip a single line and using a slash (/) to skip to a new page.
- **[options]**: options enclosed in square brackets and separated by blanks—`COLNAME=matrix`, `ROWNAME=matrix`, `FORMAT=format`, `LABEL=label`
PRINT Statement

\[
\text{PRINT } <\text{matrices}> <(\text{expression})> "\text{message}"
\]

\[
<\text{pointer-controls}> <[\text{options}]>;
\]

where (in any sensible order)

- **matrices**: names of matrices
- \((\text{expression})\): expression in parentheses that is evaluated
- "\text{message}"**: message in matching quotes
- **pointer-controls**: control the pointer for printing—using a comma (,) to skip a single line and using a slash (/) to skip to a new page.

- **[options]**: options enclosed in square brackets and separated by blanks— COLNAME=matrix, ROWNAME=matrix, FORMAT=format, LABEL=label
PRINT Statement

PRINT <matrices> <(expression)> <"message"> <pointer-controls> <[options]>;

where (in any sensible order)

- **matrices**: names of matrices
- **(expression)**: expression in parentheses that is evaluated
- **"message"**: message in matching quotes
- **pointer-controls**: control the pointer for printing—using a comma (,) to skip a single line and using a slash (/) to skip to a new page.
- **[options]**: options enclosed in square brackets and separated by blanks— COLNAME=matrix, ROWNAME=matrix, FORMAT=format, LABEL=label
Selected Functions

for creating matrices

- \( \mathbf{I}(\text{dimension}) \) & \( \mathbf{J}(\text{nrow}<, \text{ncol}<, \text{value}>>) \)
- \( \text{REPEAT}(\text{matrix, nrow, ncol}), \text{SHAPE}(\text{matrix}<, \text{nrow}<, \text{ncol}<, \text{pad-value}>>) \) and \( \text{CSHAPE}(\text{matrix, nrow, ncol, size}<, \text{padchar}>) \)
- \( \text{BLOCK}(\text{matrix1}<, \text{matrix2}, \ldots, \text{matrix15}>) \)
- \( \text{DO}(\text{start, stop, increment}) \)
- \( \text{DIAG}(\text{argument}) \) and \( \text{VECDIAG}(\text{square-matrix}) \)
- \( \text{DESIGN}(\text{column-vector}) \) and \( \text{DESIGNF}(\text{column-vector}) \)
Selected Functions
for creating matrices

- \( I(dimension) \) & \( J(nrow<, ncol<, value>) \)
- \( \text{REPEAT}(\text{matrix}, \ nrow, \ ncol), \ \text{SHAPE}(\text{matrix}<, \ nrow<, \ ncol<, \ \text{pad-value}>) \) and \( \text{CSHAPE}(\text{matrix}, \ nrow, \ ncol, \ \text{size}<, \ \text{padchar}) \)
- \( \text{BLOCK}(\text{matrix1}<, \ \text{matrix2}, \ ..., \ \text{matrix15}) \)
- \( \text{DO}(\text{start}, \ \text{stop}, \ \text{increment}) \)
- \( \text{DIAG}(\text{argument}) \) and \( \text{VECDIAG}(\text{square-matrix}) \)
- \( \text{DESIGN}(\text{column-vector}) \) and \( \text{DESIGNF}(\text{column-vector}) \)
Selected Functions
for creating matrices

- \( \mathbf{I}(\text{dimension}) \) & \( \mathbf{J}(\text{nrow}<, \text{ncol}<, \text{value}>>) \)
- \( \text{REPEAT}(\text{matrix}, \text{nrow}, \text{ncol}), \text{SHAPE}(\text{matrix}<, \text{nrow}<, \text{ncol}<, \text{pad-value}>>) \) and \( \text{CSHAPE}(\text{matrix}, \text{nrow}, \text{ncol}, \text{size}<, \text{padchar}>) \)
- \( \text{BLOCK}(\text{matrix}1<, \text{matrix}2, ..., \text{matrix}15>) \)
- \( \text{DO}(\text{start}, \text{stop}, \text{increment}) \)
- \( \text{DIAG}(\text{argument}) \) and \( \text{VECDIAG}(\text{square-matrix}) \)
- \( \text{DESIGN}(\text{column-vector}) \) and \( \text{DESIGNF}(\text{column-vector}) \)
Selected Functions
for creating matrices

- \( \mathbf{I}(\text{dimension}) \) & \( \mathbf{J}(\text{nrow}<, \text{ncol}<, \text{value}>>) \)
- \( \text{REPEAT}(\text{matrix}, \text{nrow}, \text{ncol}) \), \( \text{SHAPE}(\text{matrix}<, \text{nrow}<, \text{ncol}<, \text{pad-value}>>) \) and \( \text{CSHAPE}(\text{matrix}, \text{nrow}, \text{ncol}, \text{size}<, \text{padchar}>) \)
- \( \text{BLOCK}(\text{matrix}1<, \text{matrix}2, \ldots, \text{matrix}15>) \)
- \( \text{DO}(\text{start}, \text{stop}, \text{increment}) \)
- \( \text{DIAG}(\text{argument}) \) and \( \text{VECDIAG}(\text{square-matrix}) \)
- \( \text{DESIGN}(\text{column-vector}) \) and \( \text{DESIGNF}(\text{column-vector}) \)
Selected Functions for creating matrices

- \( \mathbf{I}(\text{dimension}) \) & \( \mathbf{J}(\text{nrow}<, \text{ncol}<, \text{value}>>) \)
- \( \text{REPEAT}(\text{matrix}, \text{nrow}, \text{ncol}), \text{SHAPE}(\text{matrix}<, \text{nrow}<, \text{ncol}<, \text{pad-value}>>) \) and \( \text{CSHAPE}(\text{matrix}, \text{nrow}, \text{ncol}, \text{size}<, \text{padchar}>) \)
- \( \text{BLOCK}(\text{matrix1}<, \text{matrix2}, \ldots, \text{matrix15}>) \)
- \( \text{DO}(\text{start}, \text{stop}, \text{increment}) \)
- \( \text{DIAG}(\text{argument}) \) and \( \text{VECDIAG}(\text{square-matrix}) \)
- \( \text{DESIGN}(\text{column-vector}) \) and \( \text{DESIGNF}(\text{column-vector}) \)
Selected Functions
for creating matrices

- $\mathbf{I}(\text{dimension}) \& \mathbf{J}(\text{nrow}<, \text{ncol}<, \text{value}>)$
- \text{REPEAT} (\text{matrix}, \text{nrow}, \text{ncol}), \text{SHAPE} (\text{matrix}<, \text{nrow}<, \text{ncol}<, \text{pad-value}>>) \text{ and } \text{CSHAPE} (\text{matrix}, \text{nrow}, \text{ncol}, \text{size}<, \text{padchar}>)$
- \text{BLOCK} (\text{matrix1}<, \text{matrix2}, \ldots, \text{matrix15}>)
- \text{DO} (\text{start}, \text{stop}, \text{increment})$
- \text{DIAG} (\text{argument}) \text{ and } \text{VECDIAG} (\text{square-matrix})$
- \text{DESIGN} (\text{column-vector}) \text{ and } \text{DESIGNF} (\text{column-vector})$
Selected Functions
for creating matrices

- \( \text{I}(\text{dimension}) \) & \( \text{J}(\text{nrow}<, \text{ncol}<, \text{value}>>) \)
- \( \text{REPEAT}(\text{matrix}, \text{nrow}, \text{ncol}), \text{SHAPE}(\text{matrix}<, \text{nrow}<, \text{ncol}<, \text{pad-value}>>) \) and \( \text{CSHAPE}(\text{matrix}, \text{nrow}, \text{ncol}, \text{size}<, \text{padchar}>) \)
- \( \text{BLOCK}(\text{matrix1}<, \text{matrix2}, \ldots, \text{matrix15}>) \)
- \( \text{DO}(\text{start}, \text{stop}, \text{increment}) \)
- \( \text{DIAG}(\text{argument}) \) and \( \text{VECDIAG}(\text{square-matrix}) \)
- \( \text{DESIGN}(\text{column-vector}) \) and \( \text{DESIGNF}(\text{column-vector}) \)
Selected Functions

for information about matrices

- \texttt{NCOL}(matrix) \text{ and } \texttt{NROW}(matrix)
- \texttt{LENGTH}(matrix) \text{ and } \texttt{NLENG}(matrix)
- \texttt{LOC}(matrix)
- \texttt{DET}(square-matrix), \texttt{EIGVAL}(square-matrix) \text{ and } \texttt{EIGVEC}(square-matrix)
Selected Functions
for information about matrices

- **NCOL** (*matrix*) and **NROW** (*matrix*)
- **LENGTH** (*matrix*) and **NLENG** (*matrix*)
- **LOC** (*matrix*)
- **DET** (*square-matrix*), **EIGVAL** (*square-matrix*) and **EIGVEC** (*square-matrix*)
Selected Functions
for information about matrices

- **NCOL** *(matrix)* and **NROW** *(matrix)*
- **LENGTH** *(matrix)* and **NLENG** *(matrix)*
- **LOC** *(matrix)*
- **DET** *(square-matrix)*, **EIGVAL** *(square-matrix)* and **EIGVEC** *(square-matrix)*
Selected Functions
for information about matrices

- `NCOL(matrix)` and `NROW(matrix)`
- `LENGTH(matrix)` and `NLENG(matrix)`
- `LOC(matrix)`
- `DET(square-matrix)`, `EIGVAL(square-matrix)` and `EIGVEC(square-matrix)`
Selected Functions
for information about matrices

- `NCOL(matrix)` and `NROW(matrix)`
- `LENGTH(matrix)` and `NLENG(matrix)`
- `LOC(matrix)`
- `DET(square-matrix)`, `EIGVAL(square-matrix)` and `EIGVEC(square-matrix)`