Introduction to SNA with R and statnet

Ryan M. Acton, Ph.D.
Department of Sociology
University of Massachusetts Amherst

Sunbelt XXXII
Redondo Beach, CA
March 13, 2012

University of Massachusetts Amherst

Roadmap for today

1. Introduce the R software and language
   - Demonstrate basic syntax
   - Overview of data types and structures
   - Demonstrate importing/writing data
2. SNA in R
   - Importing network data
   - Manipulating network data
   - Visualization of network data
   - Calculating network descriptives
   - Handling metadata for complex networks
   - Some classical network analytic techniques

Starting with the basics...

- `R` is both
  1. a language
  2. a software platform

Starting with the basics...

- R is both
  1. a language
  2. a software platform
- The R software is open-source, cross-platform, and FREE

Starting with the basics...

- `R` is both
  1. a language
  2. a software platform
- The `R` software is open-source, cross-platform, and FREE
- Its home on the web: `http://www.r-project.org/`

Starting with the basics...

- Some other features:
  - `R` uses a command-line environment, like Stata and SAS

Starting with the basics...

- Some other features:
  - R uses a command-line environment, like Stata and SAS
  - R is highly extensible:
    - you can write your own custom functions
    - there are $\sim 3,700$ free add-on packages

Starting with the basics...

- Some other features:
  - R uses a command-line environment, like Stata and SAS
  - R is highly extensible:
    - you can write your own custom functions
    - there are $\sim 3,700$ free add-on packages
  - Generally good at reading in/writing out other file formats

Starting with the basics...

- Some other features:
  - R uses a command-line environment, like Stata and SAS
  - R is highly extensible:
    - you can write your own custom functions
    - there are $\sim 3,700$ free add-on packages
  - Generally good at reading in/writing out other file formats
  - Everything in R is an object—data, functions, everything

Fundamentals of using R

- When you type in commands at the prompt and hit Enter
  1. R tries to interpret what you've asked it to do (evaluation)
  2. If it understands what you've told it to do, no problem
  3. If it does not understand, it will likely give you an error or warning message

Fundamentals of using R

- When you type in commands at the prompt and hit 

  1. R tries to interpret what you've asked it to do (evaluation)
  2. If it understands what you've told it to do, no problem
  3. If it does not understand, it will likely give you an error or warning message
- Some commands trigger R to print something to the screen, others don't

Fundamentals of using R

- When you type in commands at the prompt and hit 

  1. R tries to interpret what you've asked it to do (evaluation)
  2. If it understands what you've told it to do, no problem
  3. If it does not understand, it will likely give you an error or warning message
- Some commands trigger R to print something to the screen, others don't
- If you type in an incomplete command, R will usually respond by changing the command prompt to the + character

Fundamentals of using R

- When you type in commands at the prompt and hit Enter
  1. R tries to interpret what you've asked it to do (evaluation)
  2. If it understands what you've told it to do, no problem
  3. If it does not understand, it will likely give you an error or warning message
- Some commands trigger R to print something to the screen, others don't
- If you type in an incomplete command, R will usually respond by changing the command prompt to the + character
  - This means you must either:
    - Type in the remainder of the command
    - Hit Esc (on a Mac) to cancel
    - Type in Ctrl + C (on Windows and Linux) to cancel

Common data types and structures in R

- R has several built-in data types and structures

Common data types and structures in R

- R has several built-in data types and structures
- Common data types:

Common data types and structures in R

- R has several built-in data types and structures
- Common data types:
  - Numeric types (like integers, real numbers, etc.)
    - `12`
    - `3.14`

Common data types and structures in R

- R has several built-in data types and structures
- Common data types:
  - Numeric types (like integers, real numbers, etc.)
    - `12`
    - `3.14`
  - Strings (alphanumeric characters in quotation marks)
    - `"hello"`
    - `"Redondo Beach, CA"`
    - `"3.14"`

Common data types and structures in R

- R has several built-in data types and structures
- Common data types:
  - Numeric types (like integers, real numbers, etc.)
    - `12`
    - `3.14`
  - Strings (alphanumeric characters in quotation marks)
    - `"hello"`
    - `"Redondo Beach, CA"`
    - `"3.14"`
- Common data structures:

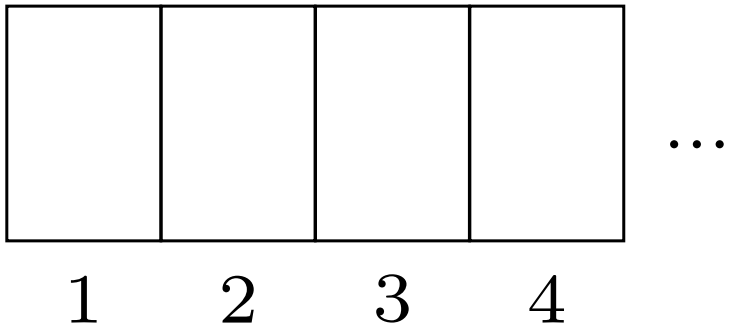
Common data types and structures in R

- R has several built-in data types and structures
- Common data types:
  - Numeric types (like integers, real numbers, etc.)
    - `12`
    - `3.14`
  - Strings (alphanumeric characters in quotation marks)
    - `"hello"`
    - `"Redondo Beach, CA"`
    - `"3.14"`
- Common data structures:
  1. One-dimensional:
     - Vectors

Common data types and structures in R

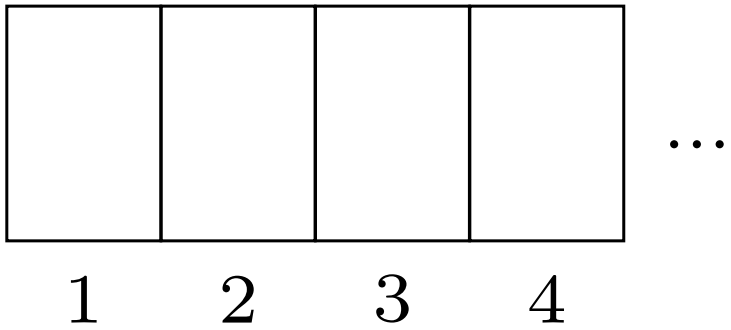
- R has several built-in data types and structures
- Common data types:
  - Numeric types (like integers, real numbers, etc.)
    - `12`
    - `3.14`
  - Strings (alphanumeric characters in quotation marks)
    - `"hello"`
    - `"Redondo Beach, CA"`
    - `"3.14"`
- Common data structures:
  1. One-dimensional:
     - Vectors
  2. Multi-dimensional:
     - Matrices
     - Data frames
     - `network` objects (demonstrated today)

Vectors in R



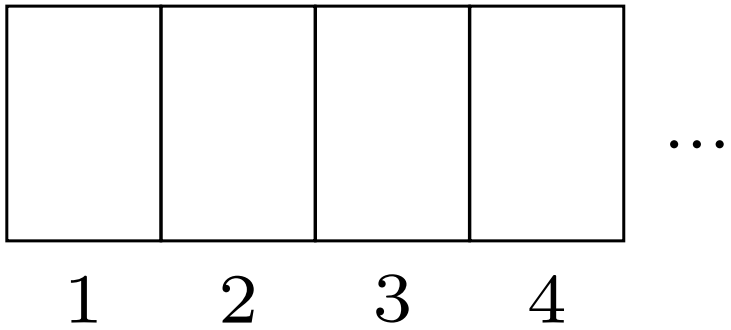
- A vector is a one-dimensional data structure
  - Its single dimension is its "length"
  - Generally, vectors can be as long as your computer has sufficient memory

Vectors in R



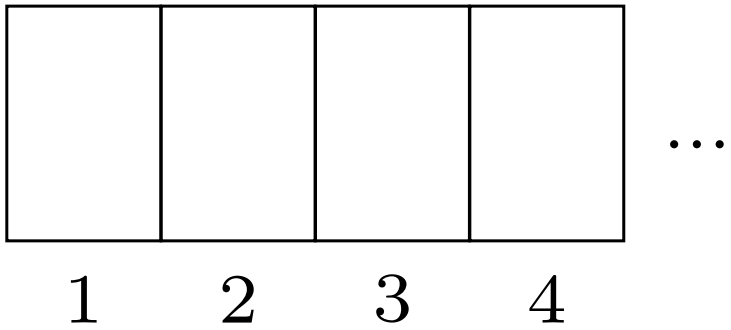
- A vector is a one-dimensional data structure
  - Its single dimension is its "length"
  - Generally, vectors can be as long as your computer has sufficient memory
- Vectors are indexed starting with 1

Vectors in R



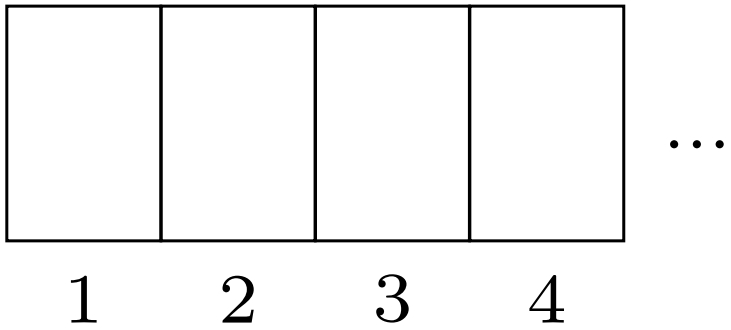
- A vector is a one-dimensional data structure
  - Its single dimension is its "length"
  - Generally, vectors can be as long as your computer has sufficient memory
- Vectors are indexed starting with 1
  - A vector of length *n* has *n* cells

Vectors in R



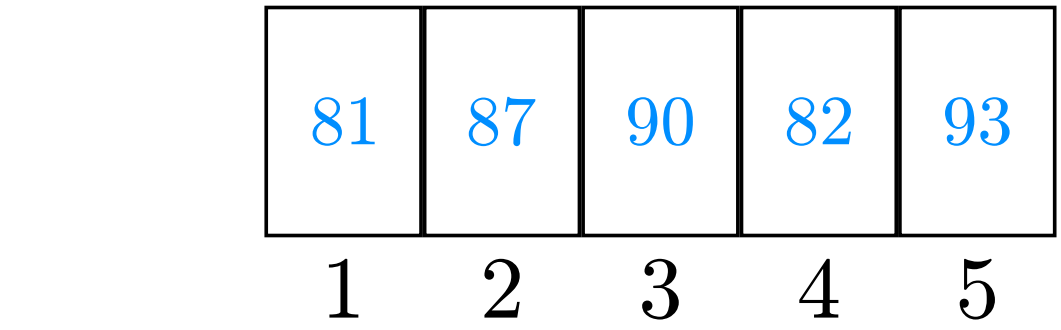
- A vector is a one-dimensional data structure
  - Its single dimension is its "length"
  - Generally, vectors can be as long as your computer has sufficient memory
- Vectors are indexed starting with 1
  - A vector of length *n* has *n* cells
  - Each cell can hold a single value, like a numeric or string value

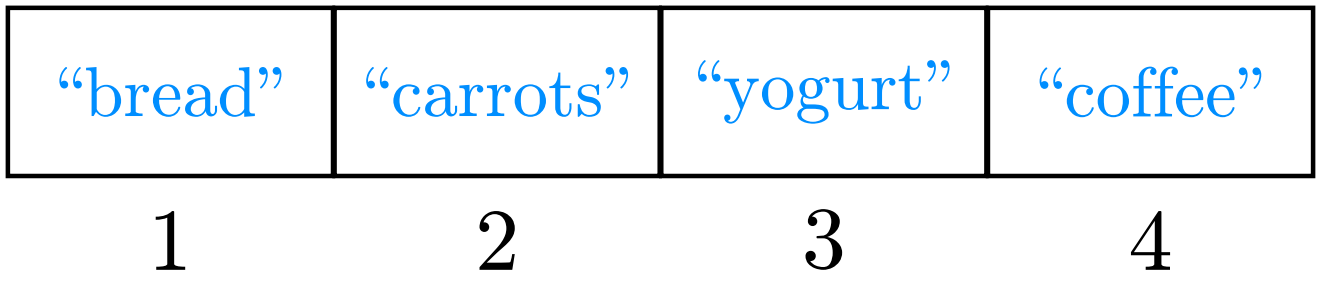
Vectors in R



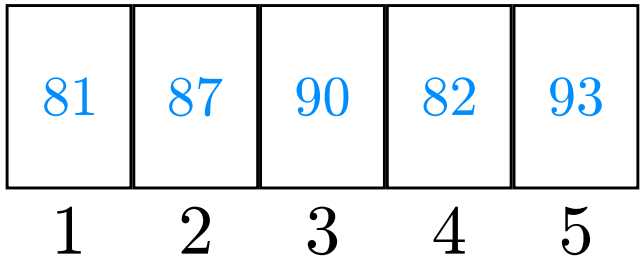
- A vector is a one-dimensional data structure
  - Its single dimension is its "length"
  - Generally, vectors can be as long as your computer has sufficient memory
- Vectors are indexed starting with 1
  - A vector of length *n* has *n* cells
  - Each cell can hold a single value, like a numeric or string value
  - Vectors can only store data of the same type
    - Either all strings, or all numerics, but not both!

Some examples of vectors in R



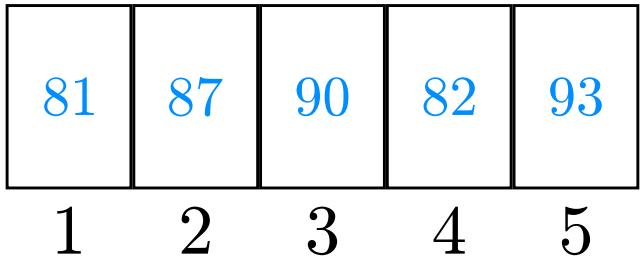


Working with vectors in R



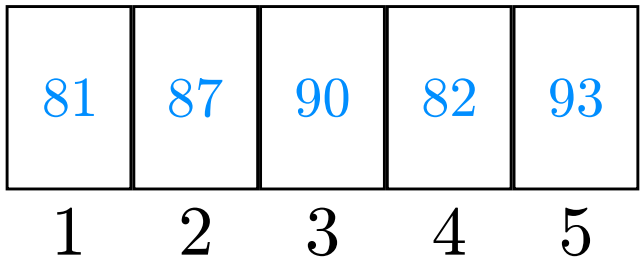
- We index vectors in R using so-called "square bracket notation"

Working with vectors in R



- We index vectors in R using so-called "square bracket notation"
- Examples:
  - Assume you have a vector of numeric values called `testScores`

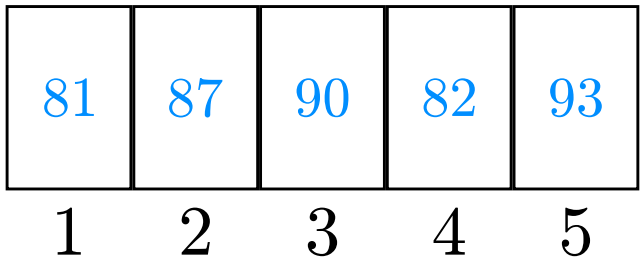
Working with vectors in R



- We index vectors in R using so-called "square bracket notation"
- Examples:
  - Assume you have a vector of numeric values called `testScores`
  - To retrieve the value in the third cell:

```
testScores[3]
```

Working with vectors in R



- We index vectors in R using so-called "square bracket notation"
- Examples:
  - Assume you have a vector of numeric values called `testScores`
  - To retrieve the value in the third cell:

```
testScores[3]
```

  - To retrieve all BUT the third value:

```
testScores[-3]
```

Why use vectors?

- They're good for storing one-dimensional data
  - Collections of values, like test scores or peoples' names

Why use vectors?

- They're good for storing one-dimensional data
  - Collections of values, like test scores or peoples' names
- They're NOT intended for higher-dimensional data
  - like data from spreadsheets, which are organized into rows and columns

Why use vectors?

- They're good for storing one-dimensional data
  - Collections of values, like test scores or peoples' names
- They're NOT intended for higher-dimensional data
  - like data from spreadsheets, which are organized into rows and columns
- However, matrix-like data (spreadsheet data) can be thought of as collections of vectors
  - Each row of a matrix is a vector
  - Each column of a matrix is a vector

Why use vectors?

- They're good for storing one-dimensional data
  - Collections of values, like test scores or peoples' names
- They're NOT intended for higher-dimensional data
  - like data from spreadsheets, which are organized into rows and columns
- However, matrix-like data (spreadsheet data) can be thought of as collections of vectors
  - Each row of a matrix is a vector
  - Each column of a matrix is a vector
- More about vectors in `R` in a few minutes...

Two-dimensional data in R

- Most (all?) of us are familiar with two-dimensional data
  - Spreadsheet-style data
- The dimensions are the *rows* and *columns*
- Here, 9 rows and 6 columns: a 9×6 data matrix

id	name	age	sex	handed	lastDocVisit
5012	Danielle	44	F	R	2012
2331	Josh	44	M	R	2008
1989	Mark	40	M	R	2010
2217	Emma	32	F	L	2012
2912	Sarah	33	F	R	2011
6604	Dave	36	M	L	2007
7201	Theresa	38	F	L	2009
4929	Carolyn	42	F	R	2009
8167	Gil	30	M	L	2010

Two-dimensional data in R

- R has two built-in data structures for storing two-dimensional data:
  1. Matrices
  2. Data frames

Two-dimensional data in R

- R has two built-in data structures for storing two-dimensional data:
  1. Matrices
  2. Data frames
- In R, these two structures look (nearly) identical

Two-dimensional data in R

- R has two built-in data structures for storing two-dimensional data:
  1. Matrices
  2. Data frames
- In R, these two structures look (nearly) identical
- In many instances, they behave the same

Two-dimensional data in R

- R has two built-in data structures for storing two-dimensional data:
  1. Matrices
  2. Data frames
- In R, these two structures look (nearly) identical
- In many instances, they behave the same
- Some R functions expect either a matrix or a data frame

Two-dimensional data in R

- R has two built-in data structures for storing two-dimensional data:
  1. Matrices
  2. Data frames
- In R, these two structures look (nearly) identical
- In many instances, they behave the same
- Some R functions expect either a matrix or a data frame
- What's the difference between the two?

Matrices versus data frames in R

- Matrices can only store data of one type
  - either all strings or all numerics, but not both!
  - if you try to give it multipe types, it converts everything to string format

Matrices versus data frames in R

- Matrices can only store data of one type
  - either all strings or all numerics, but not both!
  - if you try to give it multipe types, it converts everything to string format
- Data frames can store data of multiple types
  - Ideal for classical data analysis, where you might have a mix of numeric and string values
  - E.g., a mix of peoples' ages, favorite colors, BMI scores, names

Working with matrices and data frames in R

id	name	age	sex	handed	lastDocVisit
5012	Danielle	44	F	R	2012
2331	Josh	44	M	R	2008
1989	Mark	40	M	R	2010
2217	Emma	32	F	L	2012
2912	Sarah	33	F	R	2011
6604	Dave	36	M	L	2007
7201	Theresa	38	F	L	2009
4929	Carolyn	42	F	R	2009
8167	Gil	30	M	L	2010

- We also index matrices and data frames using "square bracket notation," indexing the rows and columns separately

Working with matrices and data frames in R

id	name	age	sex	handed	lastDocVisit
5012	Danielle	44	F	R	2012
2331	Josh	44	M	R	2008
1989	Mark	40	M	R	2010
2217	Emma	32	F	L	2012
2912	Sarah	33	F	R	2011
6604	Dave	36	M	L	2007
7201	Theresa	38	F	L	2009
4929	Carolyn	42	F	R	2009
8167	Gil	30	M	L	2010

- We also index matrices and data frames using "square bracket notation," indexing the rows and columns separately
  - In the square brackets, the first position refers to the row(s), and the second position refers to the column(s)

Working with matrices and data frames in R

id	name	age	sex	handed	lastDocVisit
5012	Danielle	44	F	R	2012
2331	Josh	44	M	R	2008
1989	Mark	40	M	R	2010
2217	Emma	32	F	L	2012
2912	Sarah	33	F	R	2011
6604	Dave	36	M	L	2007
7201	Theresa	38	F	L	2009
4929	Carolyn	42	F	R	2009
8167	Gil	30	M	L	2010

- We also index matrices and data frames using "square bracket notation," indexing the rows and columns separately
  - In the square brackets, the first position refers to the row(s), and the second position refers to the column(s)
- Examples:
  - Assume the above data are stored in R as `friendSurvey`

Working with matrices and data frames in R

id	name	age	sex	handed	lastDocVisit
5012	Danielle	44	F	R	2012
2331	Josh	44	M	R	2008
1989	Mark	40	M	R	2010
2217	Emma	32	F	L	2012
2912	Sarah	33	F	R	2011
6604	Dave	36	M	L	2007
7201	Theresa	38	F	L	2009
4929	Carolyn	42	F	R	2009
8167	Gil	30	M	L	2010

- We also index matrices and data frames using "square bracket notation," indexing the rows and columns separately
  - In the square brackets, the first position refers to the row(s), and the second position refers to the column(s)
- Examples:
  - Assume the above data are stored in R as `friendSurvey`
  - To retrieve Sarah's age (6th row, 3rd column):

```
friendSurvey[6,3]
```

Working with matrices and data frames in R

id	name	age	sex	handed	lastDocVisit
5012	Danielle	44	F	R	2012
2331	Josh	44	M	R	2008
1989	Mark	40	M	R	2010
2217	Emma	32	F	L	2012
2912	Sarah	33	F	R	2011
6604	Dave	36	M	L	2007
7201	Theresa	38	F	L	2009
4929	Carolyn	42	F	R	2009
8167	Gil	30	M	L	2010

- To retrieve all of the data from the 2nd row:

```
friendSurvey[2,]
```

Working with matrices and data frames in R

id	name	age	sex	handed	lastDocVisit
5012	Danielle	44	F	R	2012
2331	Josh	44	M	R	2008
1989	Mark	40	M	R	2010
2217	Emma	32	F	L	2012
2912	Sarah	33	F	R	2011
6604	Dave	36	M	L	2007
7201	Theresa	38	F	L	2009
4929	Carolyn	42	F	R	2009
8167	Gil	30	M	L	2010

- To retrieve all of the data from the 2nd row:

```
friendSurvey[2,]
```

- To retrieve all of the data from the 4th column:

```
friendSurvey[,4]
```

Working with matrices and data frames in R

- Square bracket notation works for both matrix and data frame objects in R

Working with matrices and data frames in R

- Square bracket notation works for both matrix and data frame objects in R
- Data frames provide you another option: "dollar sign notation"
  - Only works for retrieving specific columns

Working with matrices and data frames in R

id	name	age	sex	handed	lastDocVisit
5012	Danielle	44	F	R	2012
2331	Josh	44	M	R	2008
1989	Mark	40	M	R	2010
2217	Emma	32	F	L	2012
2912	Sarah	33	F	R	2011
6604	Dave	36	M	L	2007
7201	Theresa	38	F	L	2009
4929	Carolyn	42	F	R	2009
8167	Gil	30	M	L	2010

- If these data were stored as a data frame in R, we could use dollar sign notation to retrieve individual column vectors:

Working with matrices and data frames in R

id	name	age	sex	handed	lastDocVisit
5012	Danielle	44	F	R	2012
2331	Josh	44	M	R	2008
1989	Mark	40	M	R	2010
2217	Emma	32	F	L	2012
2912	Sarah	33	F	R	2011
6604	Dave	36	M	L	2007
7201	Theresa	38	F	L	2009
4929	Carolyn	42	F	R	2009
8167	Gil	30	M	L	2010

- If these data were stored as a data frame in R, we could use dollar sign notation to retrieve individual column vectors:
  - To retrieve the "sex" column vector:

```
friendSurvey$sex
```

Working with matrices and data frames in R

id	name	age	sex	handed	lastDocVisit
5012	Danielle	44	F	R	2012
2331	Josh	44	M	R	2008
1989	Mark	40	M	R	2010
2217	Emma	32	F	L	2012
2912	Sarah	33	F	R	2011
6604	Dave	36	M	L	2007
7201	Theresa	38	F	L	2009
4929	Carolyn	42	F	R	2009
8167	Gil	30	M	L	2010

- If these data were stored as a data frame in R, we could use dollar sign notation to retrieve individual column vectors:
  - To retrieve the "sex" column vector:

```
friendSurvey$sex
```

  - To retrieve the the 4th person's age:

```
friendSurvey$age[4]
```