Lecture 3: Control Structures + Functions

Intro to Data Science for Public Policy
Spring 2017

Jeff Chen + Dan Hammer

Roadmap

- **Motivating Story**
- Control Structures
- Functions
- Code-along: Collaborative Filtering

Motivation

Oh, SNAP!

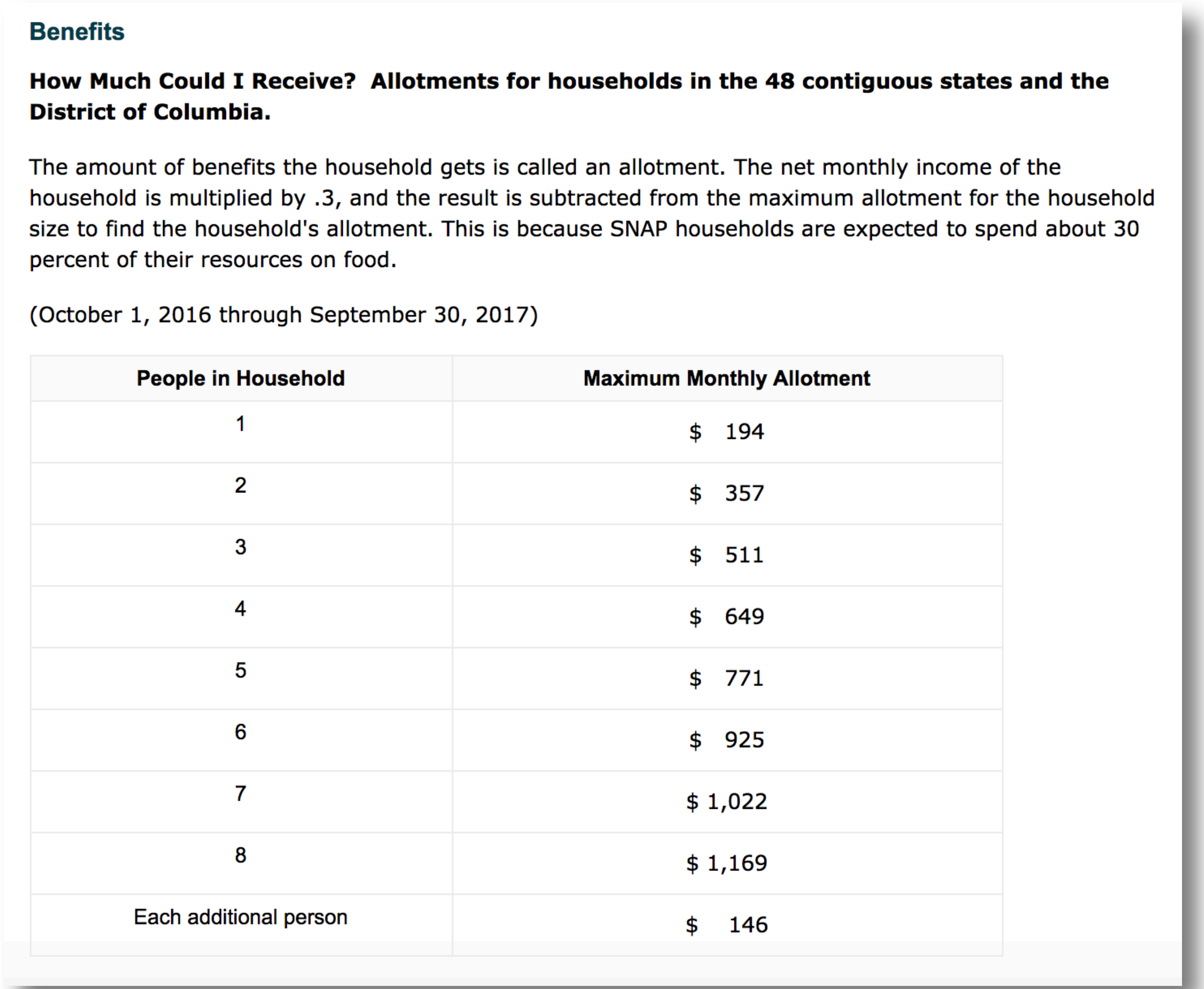
(Supplemental Nutrition Assistance Program
Or "Food Stamps")

https://www.fns.usda.gov/snap/eligibility

Background

- US Department of Agriculture (USDA) program
- Original 'Food Stamps Program' established in 1939 and has evolved ever since. Peak participation at one time was 4 million.
- Now mostly 'Electronic Benefits Transfer' (EBT)
- FY2016
  - Benefits cost = $70.9 billion
  - User base = 44.2 million Americans

SNAP Benefits for FY2017



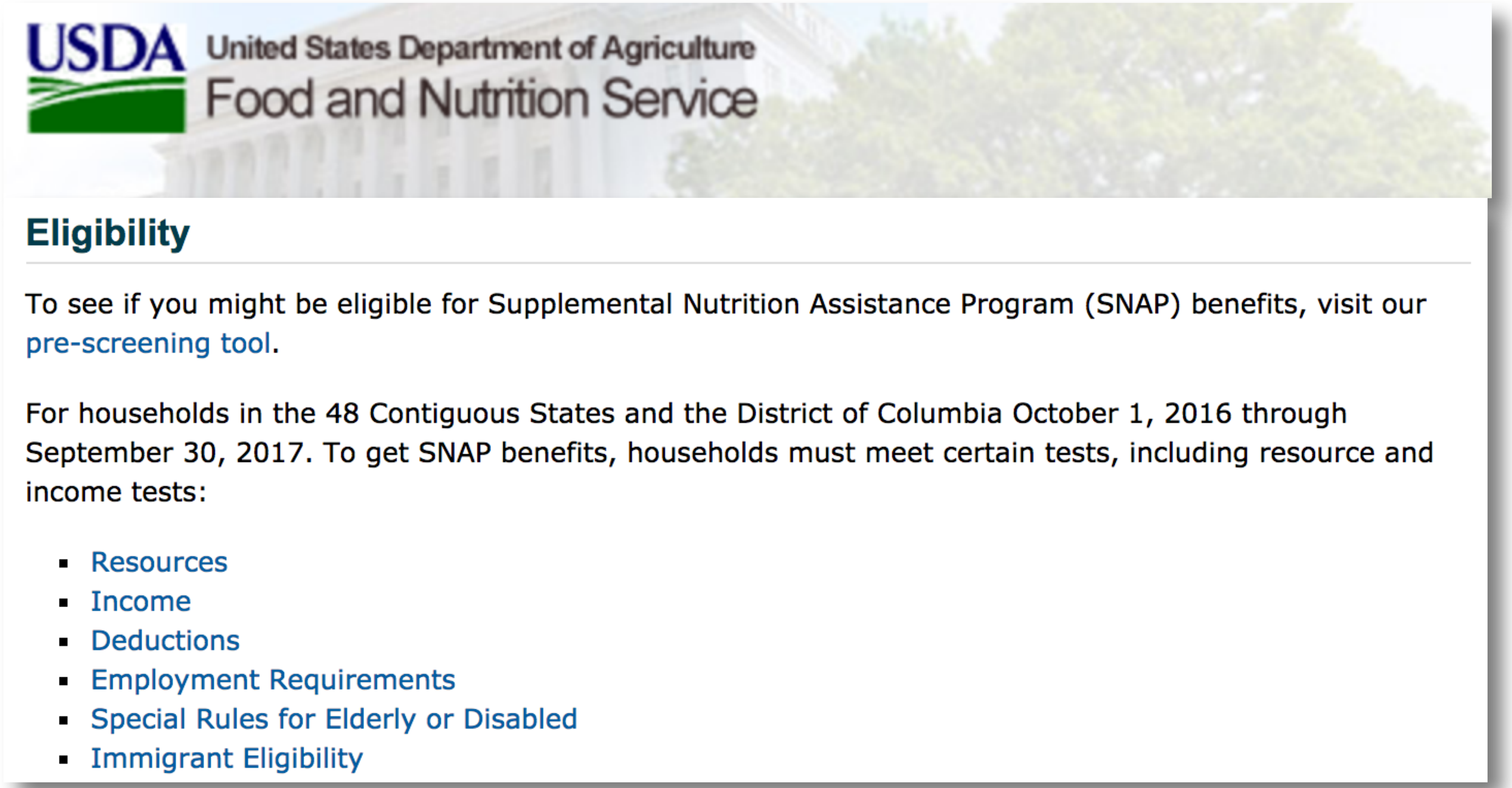
Benefits

How Much Could I Receive? Allotments for households in the 48 contiguous states and the District of Columbia.

The amount of benefits the household gets is called an allotment. The net monthly income of the household is multiplied by .3, and the result is subtracted from the maximum allotment for the household size to find the household's allotment. This is because SNAP households are expected to spend about 30 percent of their resources on food.

(October 1, 2016 through September 30, 2017)

People in Household	Maximum Monthly Allotment
1	$ 194
2	$ 357
3	$ 511
4	$ 649
5	$ 771
6	$ 925
7	$ 1,022
8	$ 1,169
Each additional person	$ 146


United States Department of Agriculture
Food and Nutrition Service

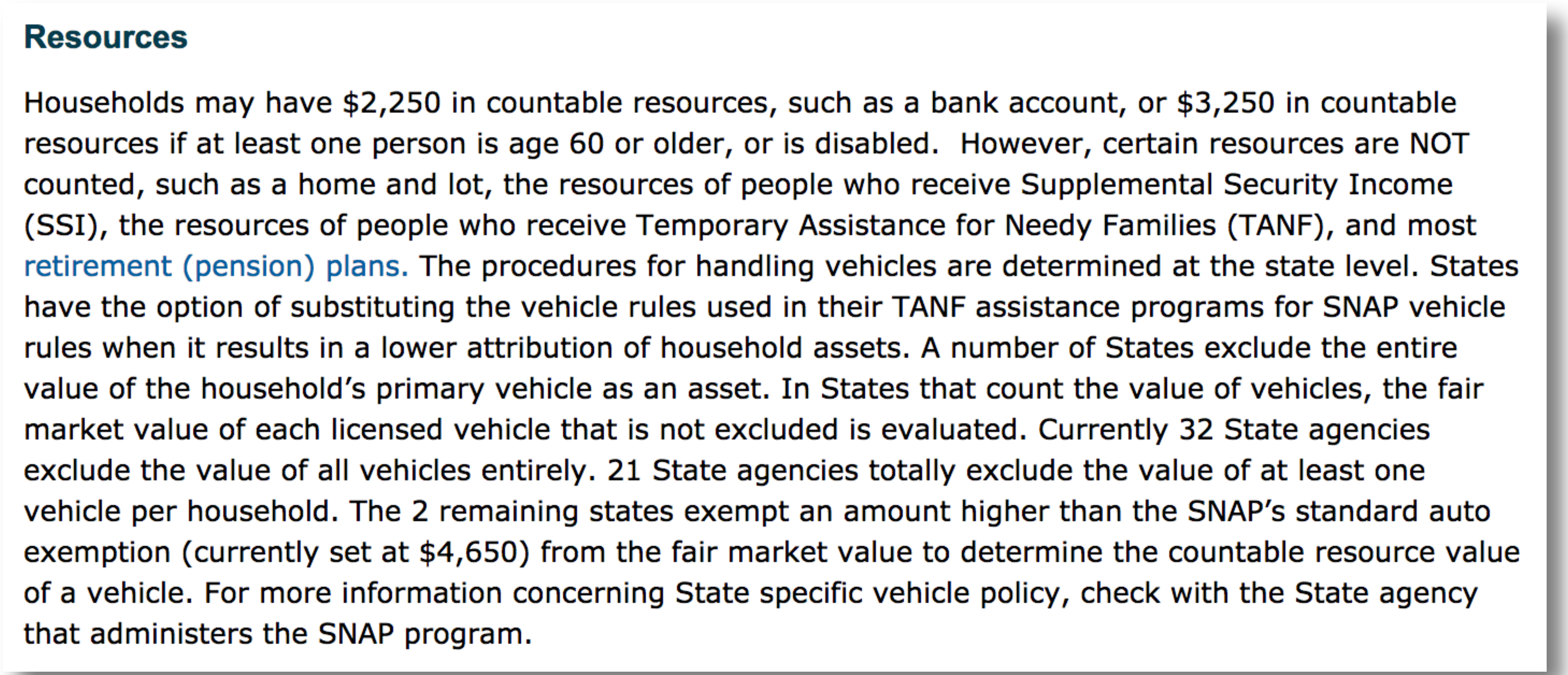
Eligibility

To see if you might be eligible for Supplemental Nutrition Assistance Program (SNAP) benefits, visit our pre-screening tool.

For households in the 48 Contiguous States and the District of Columbia October 1, 2016 through September 30, 2017. To get SNAP benefits, households must meet certain tests, including resource and income tests:

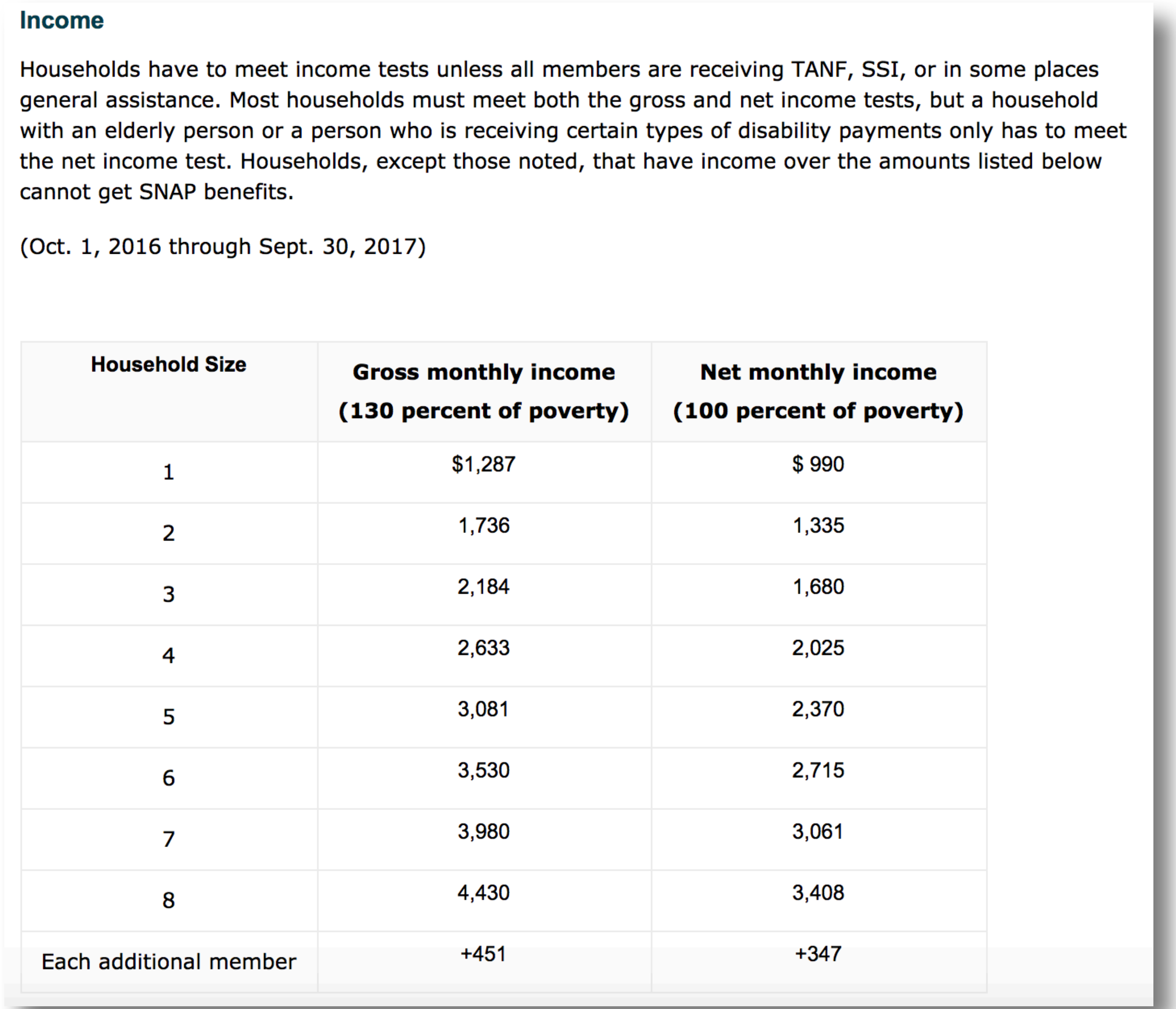
- Resources
- Income
- Deductions
- Employment Requirements
- Special Rules for Elderly or Disabled
- Immigrant Eligibility

Motivation



Resources

Households may have $2,250 in countable resources, such as a bank account, or $3,250 in countable resources if at least one person is age 60 or older, or is disabled. However, certain resources are NOT counted, such as a home and lot, the resources of people who receive Supplemental Security Income (SSI), the resources of people who receive Temporary Assistance for Needy Families (TANF), and most retirement (pension) plans. The procedures for handling vehicles are determined at the state level. States have the option of substituting the vehicle rules used in their TANF assistance programs for SNAP vehicle rules when it results in a lower attribution of household assets. A number of States exclude the entire value of the household's primary vehicle as an asset. In States that count the value of vehicles, the fair market value of each licensed vehicle that is not excluded is evaluated. Currently 32 State agencies exclude the value of all vehicles entirely. 21 State agencies totally exclude the value of at least one vehicle per household. The 2 remaining states exempt an amount higher than the SNAP's standard auto exemption (currently set at $4,650) from the fair market value to determine the countable resource value of a vehicle. For more information concerning State specific vehicle policy, check with the State agency that administers the SNAP program.

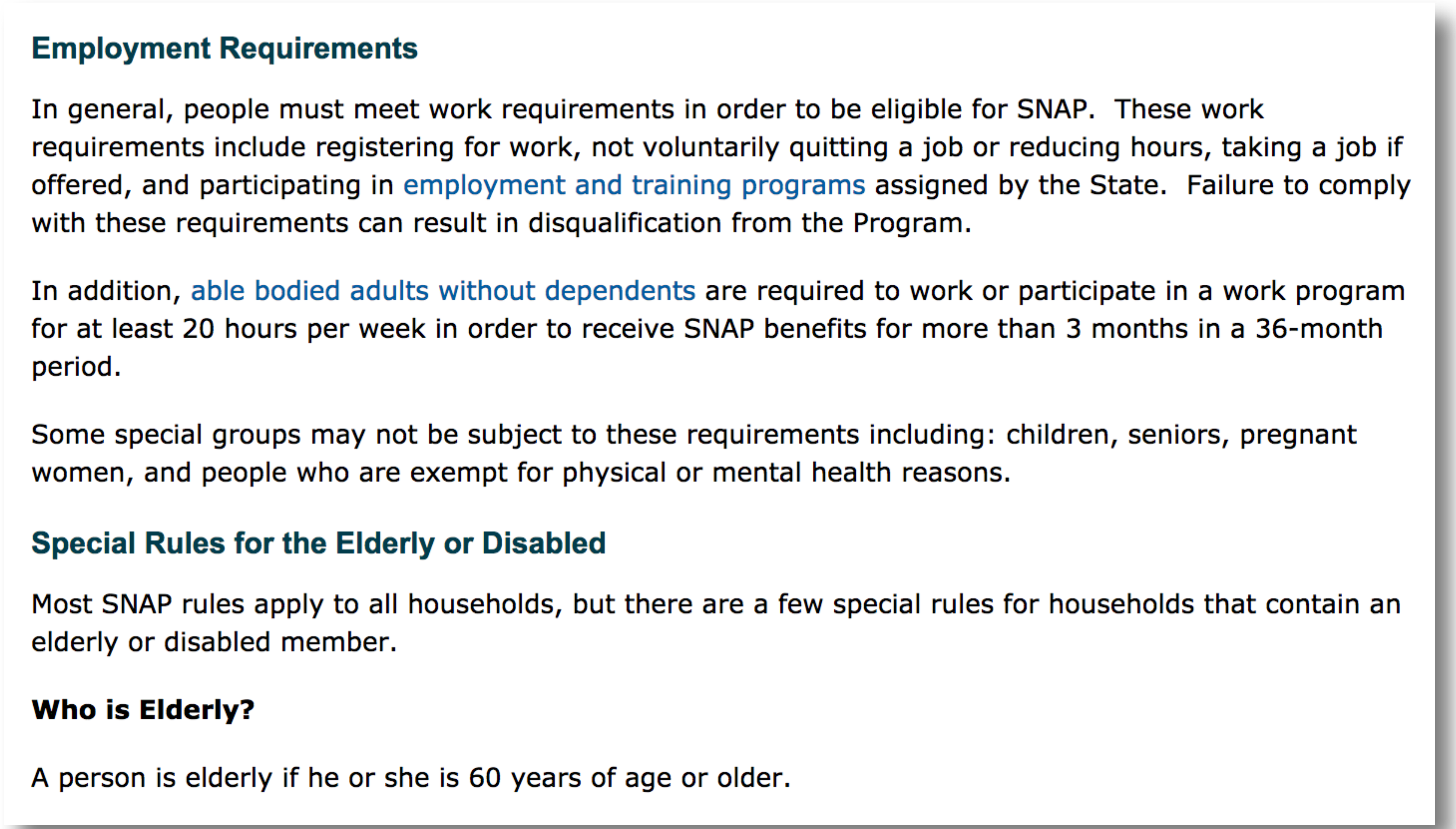

Income

Households have to meet income tests unless all members are receiving TANF, SSI, or in some places general assistance. Most households must meet both the gross and net income tests, but a household with an elderly person or a person who is receiving certain types of disability payments only has to meet the net income test. Households, except those noted, that have income over the amounts listed below cannot get SNAP benefits.

(Oct. 1, 2016 through Sept. 30, 2017)

Household Size	Gross monthly income (130 percent of poverty)	Net monthly income (100 percent of poverty)
1	$1,287	$ 990
2	1,736	1,335
3	2,184	1,680
4	2,633	2,025
5	3,081	2,370
6	3,530	2,715
7	3,980	3,061
8	4,430	3,408
Each additional member	+451	+347

Motivation


Employment Requirements

In general, people must meet work requirements in order to be eligible for SNAP. These work requirements include registering for work, not voluntarily quitting a job or reducing hours, taking a job if offered, and participating in employment and training programs assigned by the State. Failure to comply with these requirements can result in disqualification from the Program.

In addition, able bodied adults without dependents are required to work or participate in a work program for at least 20 hours per week in order to receive SNAP benefits for more than 3 months in a 36-month period.

Some special groups may not be subject to these requirements including: children, seniors, pregnant women, and people who are exempt for physical or mental health reasons.

Special Rules for the Elderly or Disabled

Most SNAP rules apply to all households, but there are a few special rules for households that contain an elderly or disabled member.

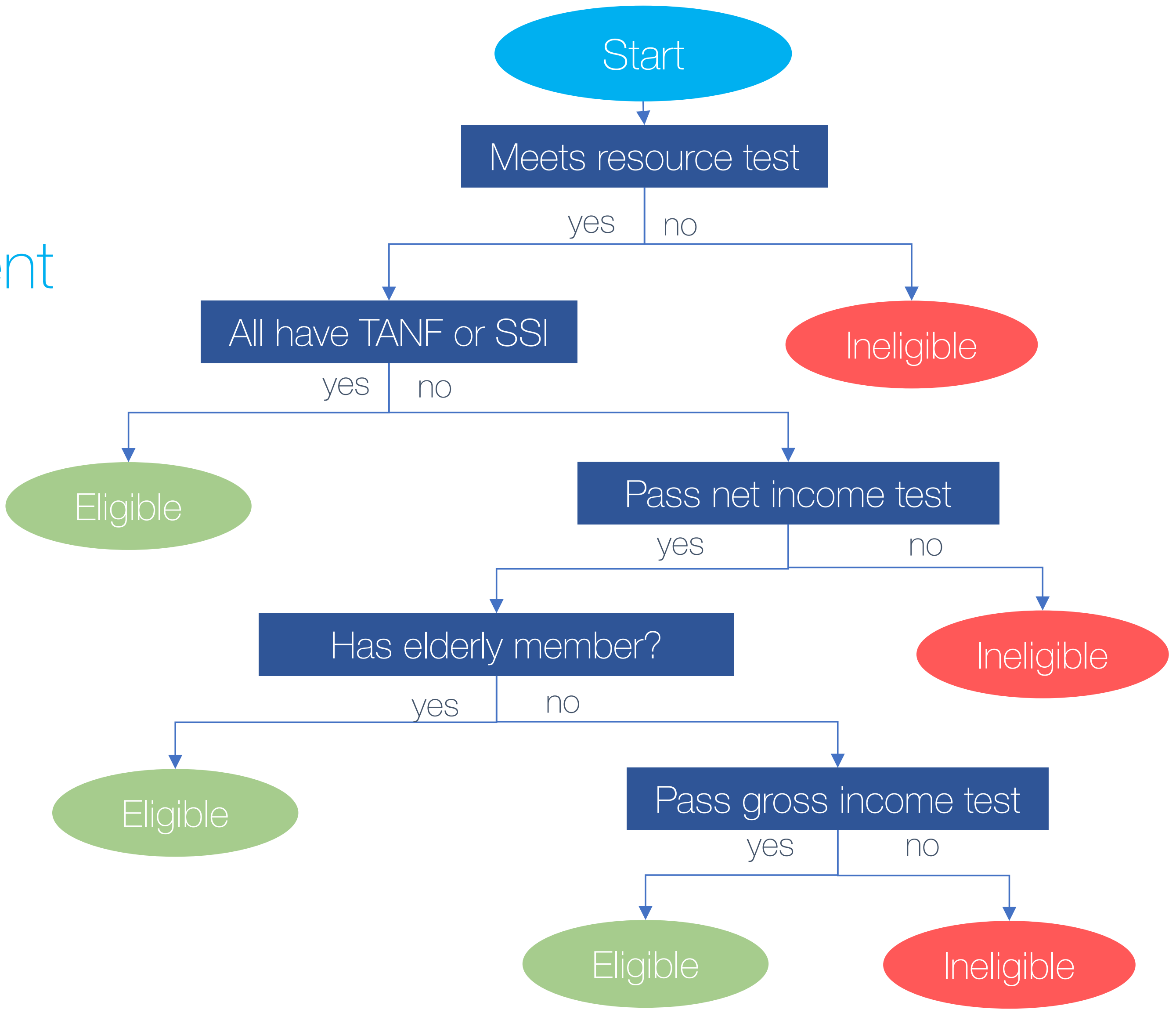
Who is Elderly?

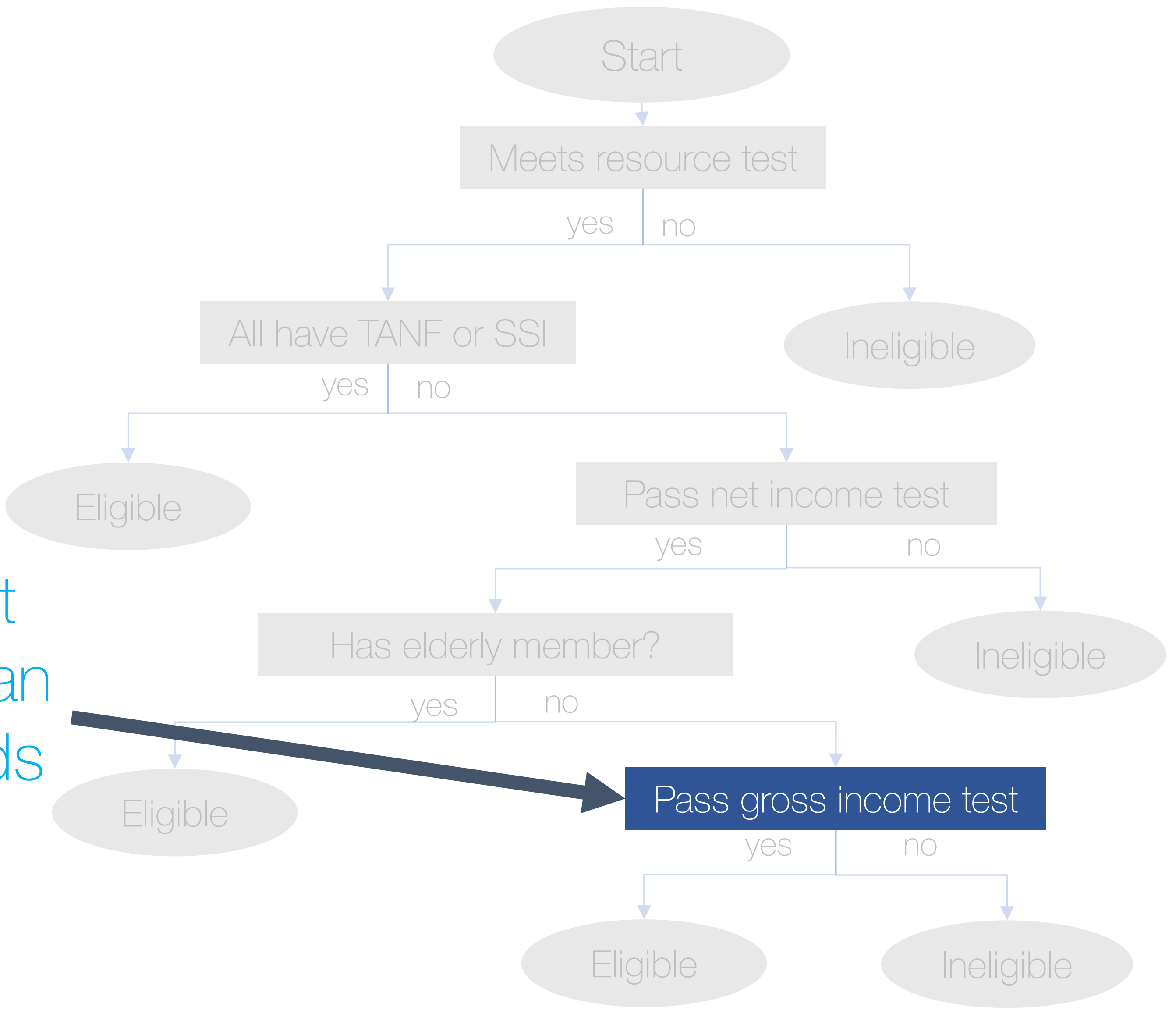
A person is elderly if he or she is 60 years of age or older.

Imagine if you had to figure that out manually for every applicant?!

A Screening Process.

A decision tree, excluding employment requirement.





Each decision point or "node" involves an argument that needs to be evaluated

Motivation

The procedures associated with the node can be codified into a series of pre-specified actions that converts an input into an output, or a **function**.

Function. Pass gross income test (Pseudo code)

```
INPUTS gross.income, family.size
TABLES ref
OUTPUT TRUE or FALSE

IF family.size <= 8,
    a = FIND gross.income where ref.family.size == family.size
    RETURN gross.income < a
ELSE
    a = FIND gross.income where ref.family.size == 8
    b = +347 x (family.size – 8)

    RETURN gross.income < a + b
```

What to do to screen a list of households?

(1) Identify the minimum parameters

Resource = Amount in bank account, etc.

Welfare = List of other program participation

Net Income, Gross Income, Family Size

Elderly family member?

Employment status

(2) Write discrete functions that handle each node if it requires any data manipulation.

Func. Meets resource test

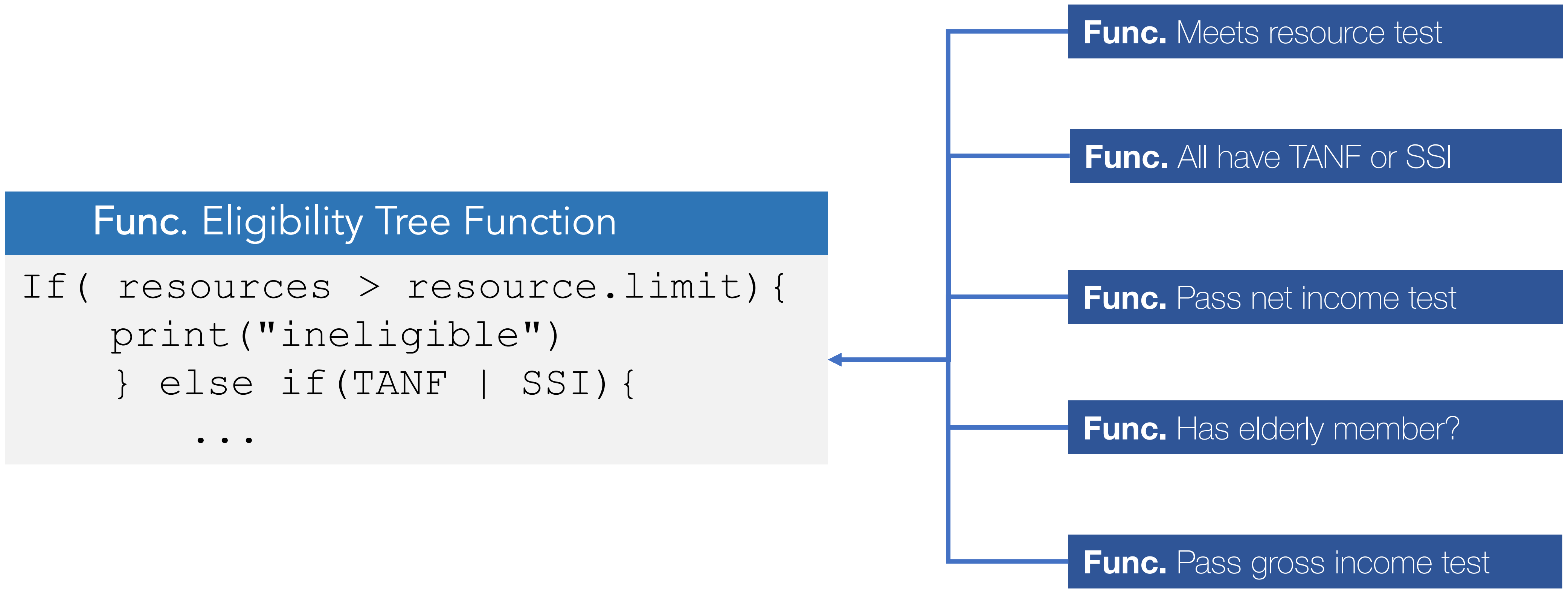
Func. All have TANF or SSI

Func. Pass net income test

Func. Has elderly member?

Func. Pass gross income test

(2) Write a master function that uses the short well-defined, discrete functions

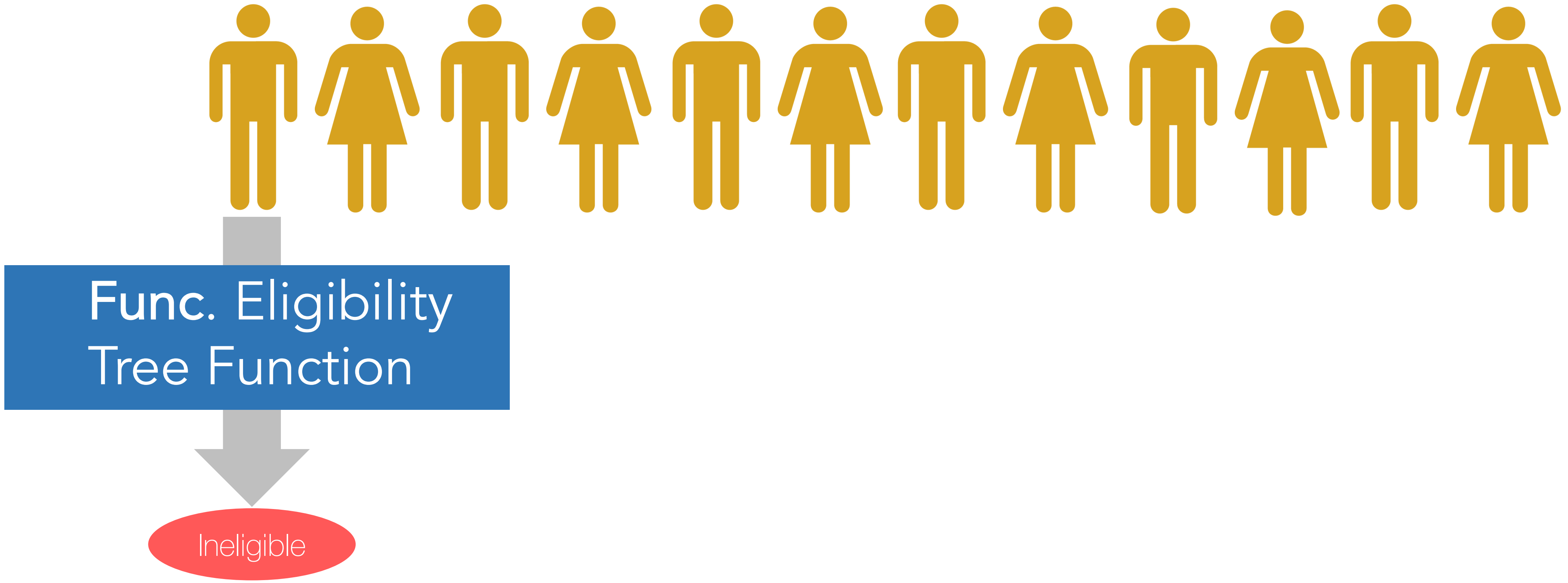

Func. Eligibility Tree Function
Func. Meets resource test
Func. All have TANF or SSI
Func. Pass net income test
Func. Has elderly member?
Func. Pass gross income test

```
If( resources > resource.limit){
    print("ineligible")
    } else if(TANF | SSI){
        ...
```

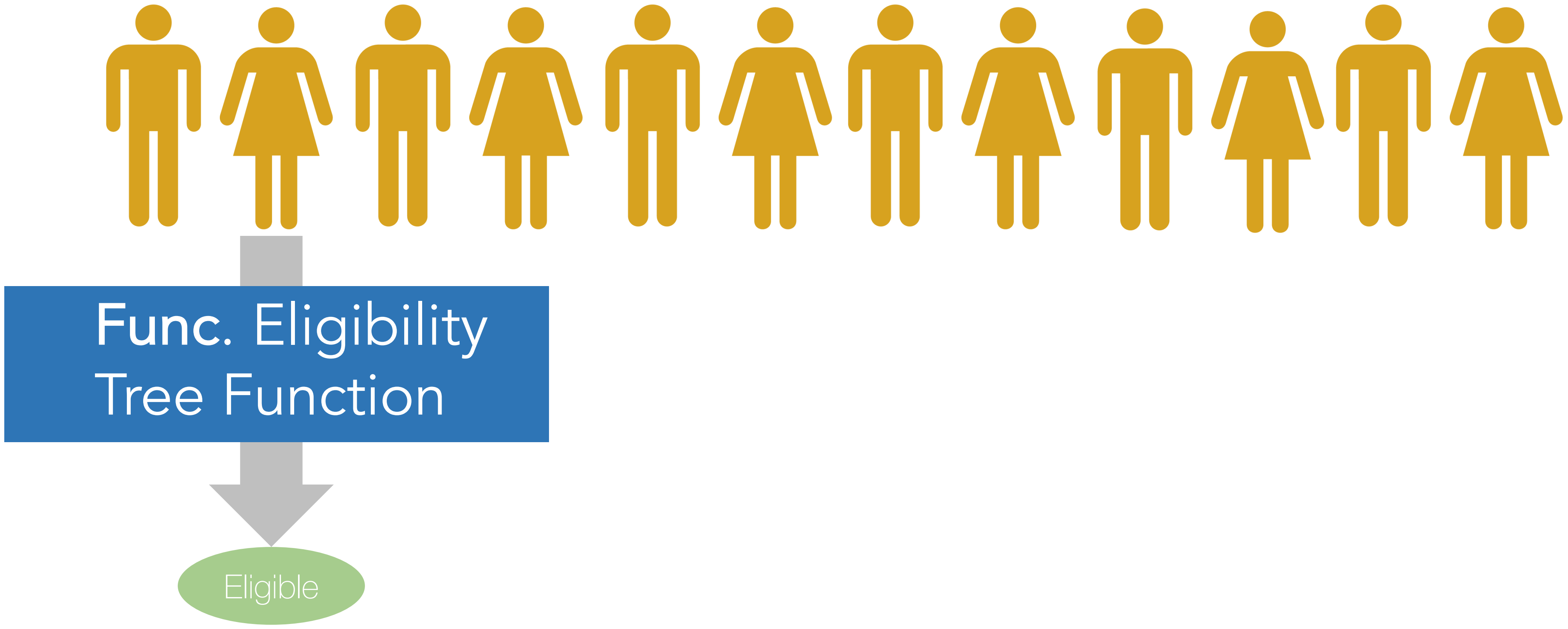
(3) Given a list of households and their attributes, apply function to each household, write out result.

Func. Eligibility Tree Function

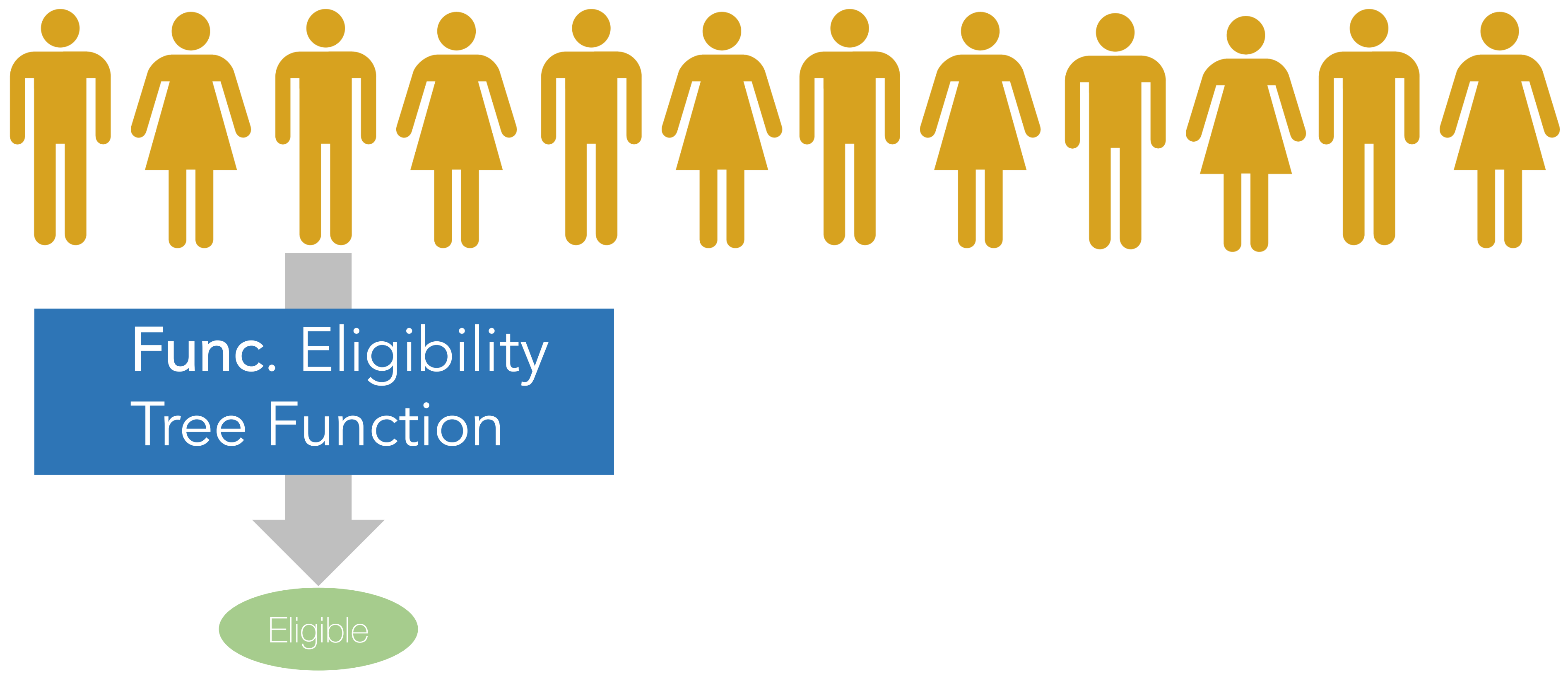
(3) Given a list of households and their attributes, apply function to each household, write out result.



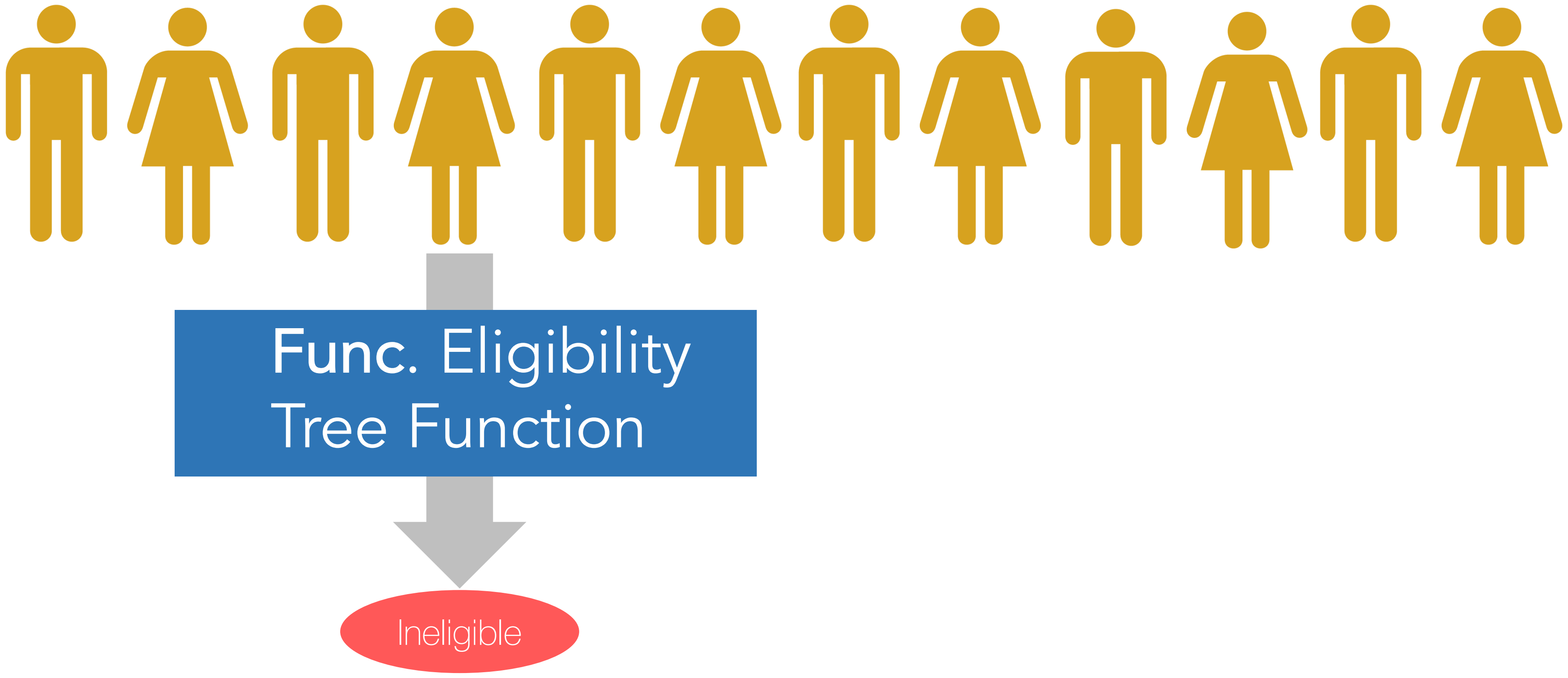
(3) Given a list of households and their attributes, apply function to each household, write out result.



(3) Given a list of households and their attributes, apply function to each household, write out result.



(3) Given a list of households and their attributes, apply function to each household, write out result.



Roadmap

- Motivating Story
- **Control Structures**
- Functions
- Code-along

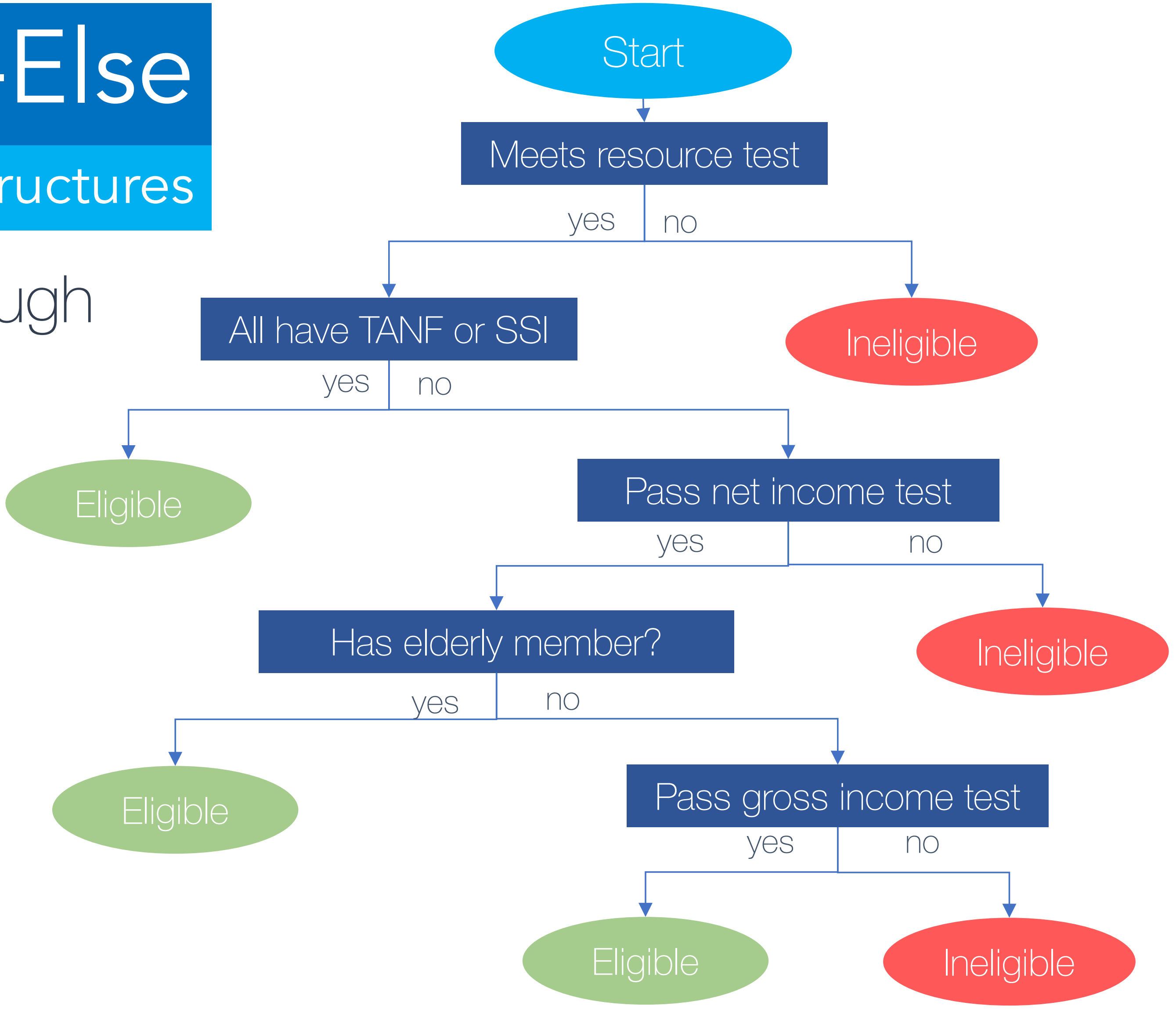
Control Structures

- A set of code that analyzes input variables and controls the direction in which the code will flow.
- Not all data should be treated the same. At times the idiosyncrasies dictate what actions should be taken.
- Control structures operate on individual instances – “one at a time”

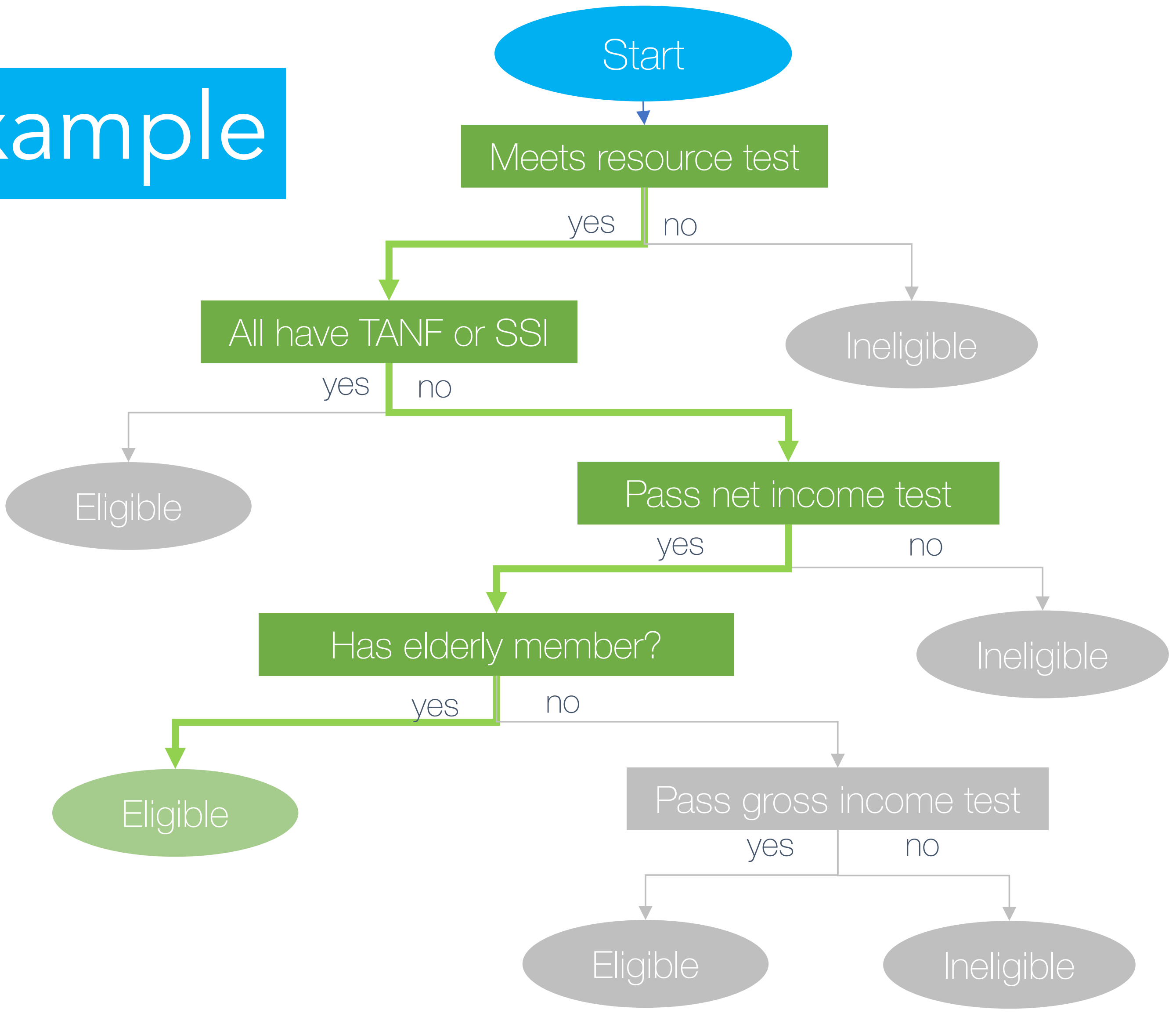
If-Else

Control Structures

A record moves through a series of logical arguments



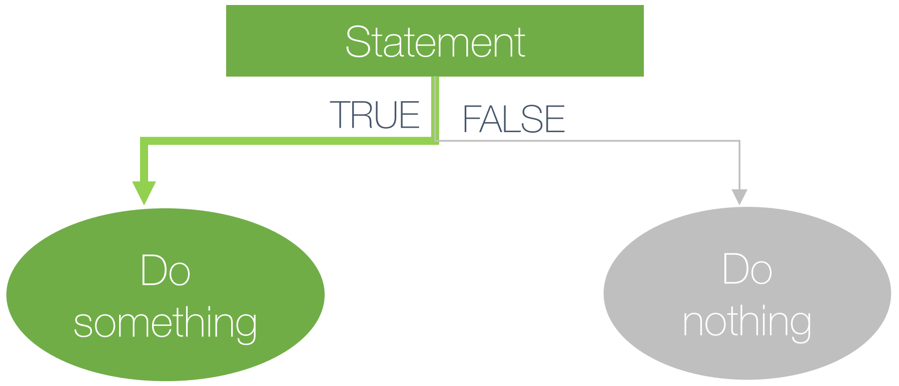
If-else : example



If structure

- One of the core control structures that evaluates a logical argument, then directs the 'flow' of the code.
- The most basic formulation evaluates an argument. If true, then executes specific code.

```
if([statement]){
    #do whatever is in here
}
```



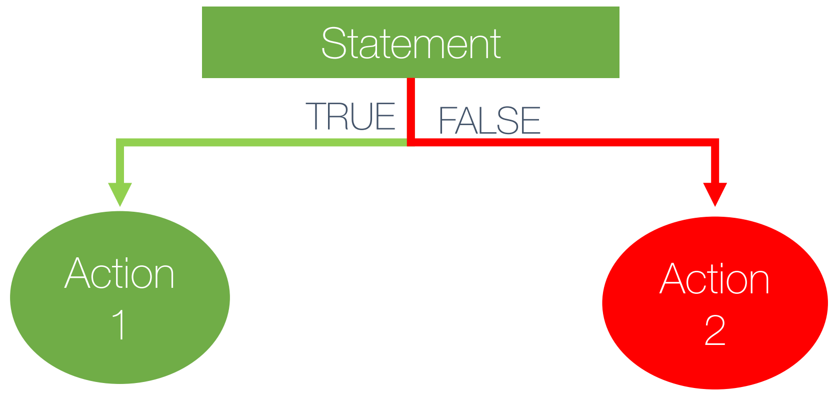
If structure

```
climate<- "round"
if(earth == "round"){
   print("The circumference is 24,901 miles!")
}
```

If-else structure

If-else adds an additional action for when the evaluated argument is FALSE.

```
if([statement]){
    #Action 1
} else {
    #Action 2
}
```



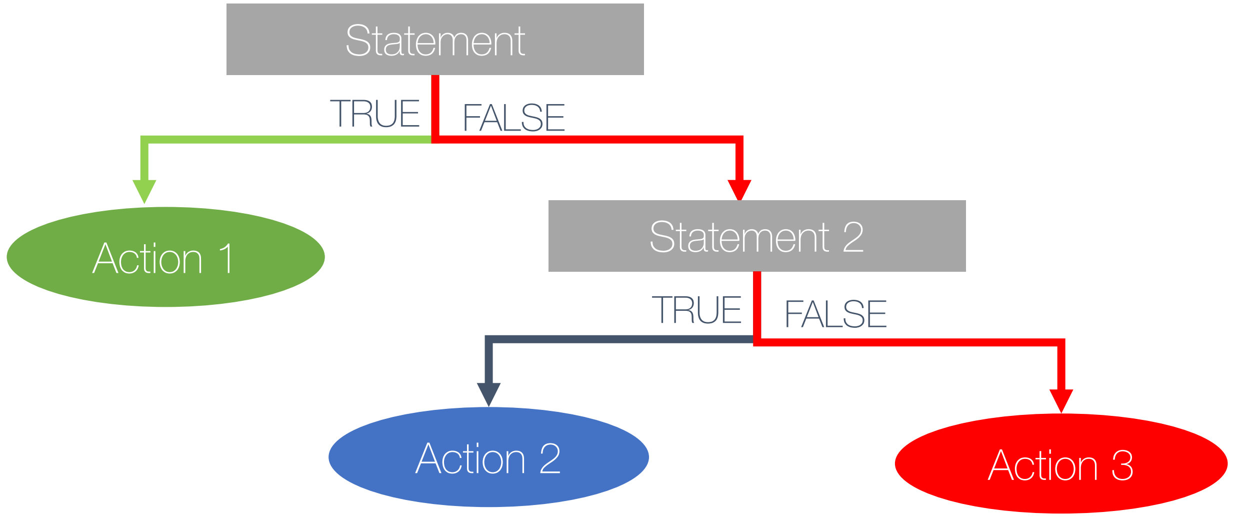
If-else structure

```r
earth <- "cubed"
if(earth == "round"){
   print("The Earth is a sphere!")
} else {
   print("Check again, buddy.")
}
```

If-else structure

If-else can be greatly expanded to handle more complex scenarios

```
if([statement]){
    #Action 1
} else if([another statement]){
    #Action 2
} else {
    #Action 3
}
```

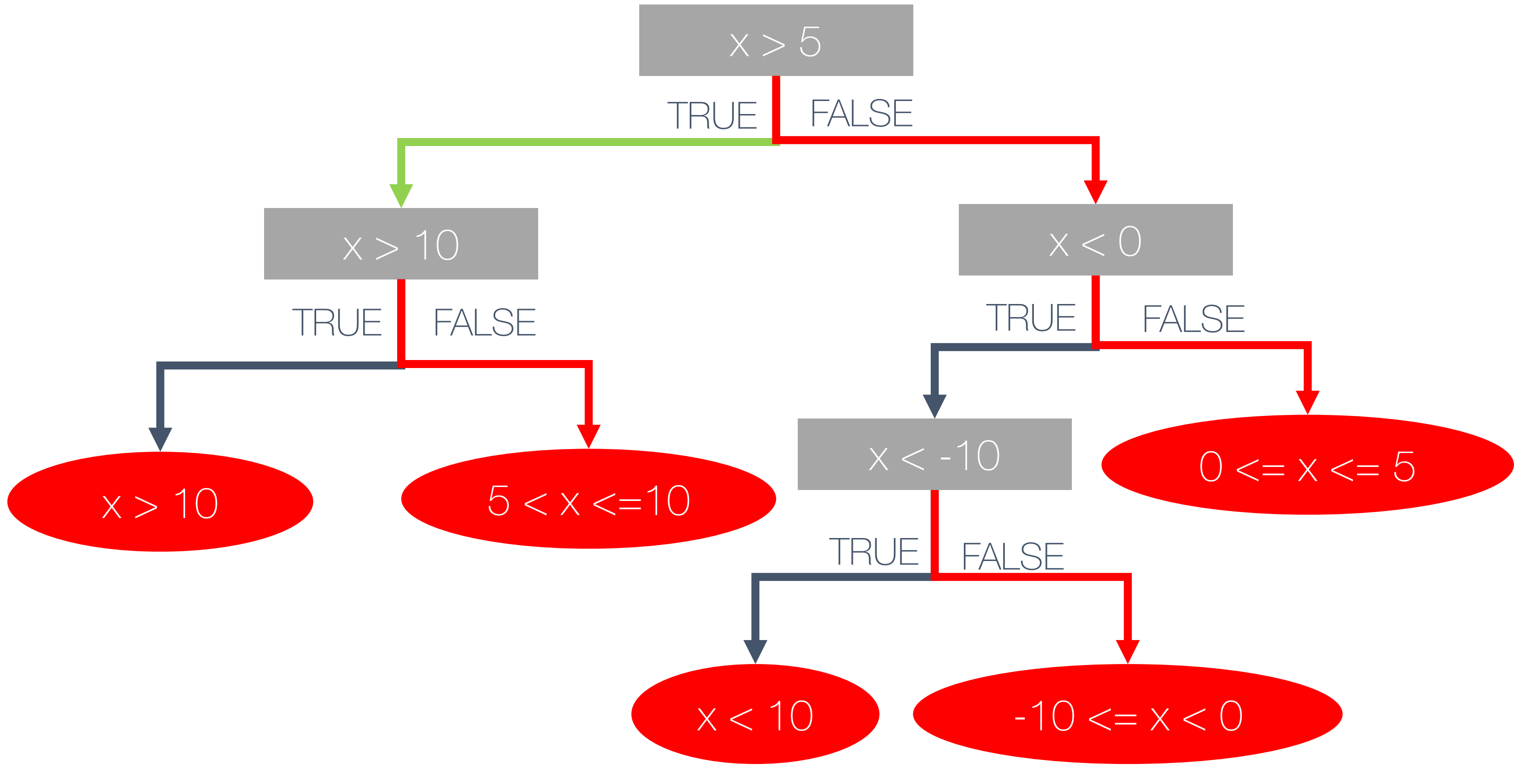


If-else structure

```
earth <- "ball"
if(earth == "sphere"){
   print("Indeed, the earth is a sphere!")
} else if(earth == "ball"){
   print("There’s a better word for that. ")
} else {
   print("Nope.")
}
```

Nested if-else exercise

Write the if-else code to print the values in red



If-else structure: exercise

```
if(x > 5){
     if(x > 10){
         print("x > 10")
    } else {
          print("5 < x <= 10")
    }
} else {
     if(x < 0 ){
         if(x < - 10){
                 print("x < -10")}
         else {
                 print("-10 <= x <= 0")                    }
    } else {
          print("0 <= x <= 5")
         }
}
```

For Loops

Control Structures

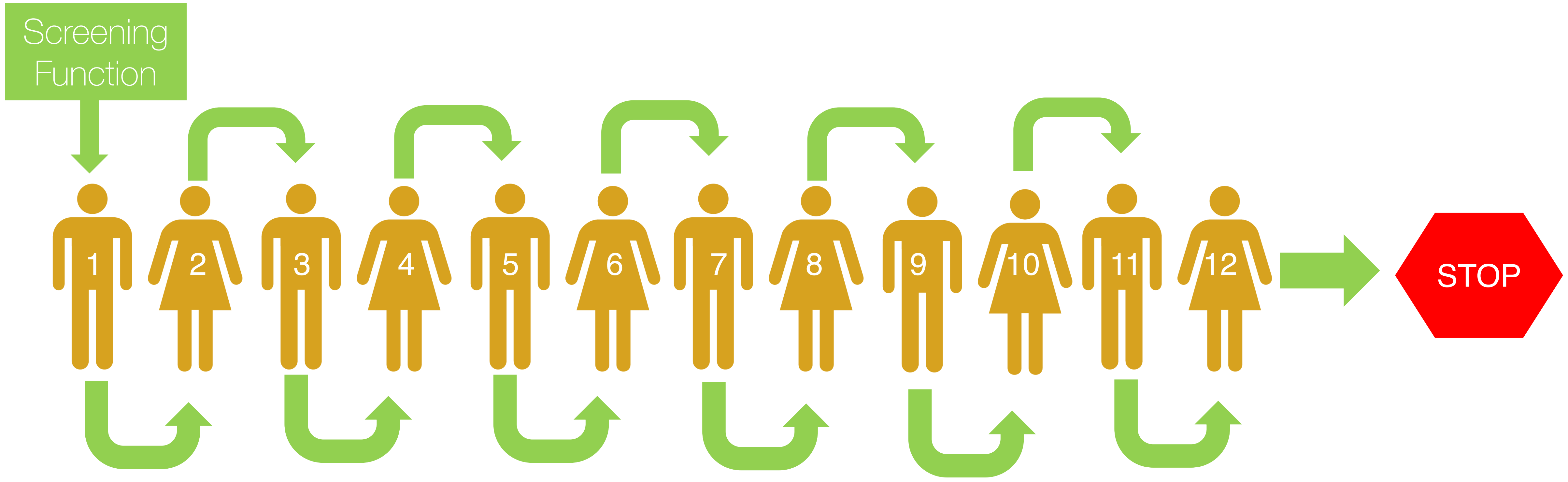
The function is applied a specified number of times (i) and returns a result for each of the i iterations

Screening Function

Apply to all 12 people

For Loops: example

The function is applied a specified number of times (i) and returns a result for each of the i iterations



For-loop construction

- For a given range of values, for-loops will iterate through each value in the range. In each iteration, the index variable “i” is replaced with the next sequential value in the range.

```
for(i in 1:10){
    print(i)
}
```

```
vec <- c("CA", "VA", "NY")
for(i in vec){
    print(i)
}
```

For-loops: Under the hood

(2) i is the "index". When initiated, the first value of **i = 1**, then after print(**i**) is run, the next value will be 2.

(1) The loop will iterate through all numbers from 1 through 10

```
for(i in 1:10){
    print(i)
}
```

(3) The print statement will execute whenever the value of **i** is changed.

For-loops: Loop the loop

What if there are multiple dimensions that need to be looped? Nested loops allow for more complex operations.

```
#Nested loops
for(i in 1:10){
    for(j in 10:1){
    print(paste(j, i))
}
}
```

Notice the change in the indices. Why is this important?

For-loops: programming paradigms

For-loops are often used to compute and transform data into new results for each record of a dataset.

- **Create dummy, then append.** The data needs to be stored. Thus, often times, a ”dummy” object needs to be created before entering into the loop. Then values are added to that object.
- **Create dummy, then overwrite.** Similar to above, create a dummy object, but specifying the dimensions of the object, then rewrite individual cells. This is useful for large datasets.

For-loops: programming paradigm

```
#Dummy --> append
#add data to empty vector
x <- c()
for(i in 1:10){
    x <- c(x, runif(i))
}
x
length(x)
```

```
#Dummy → Overwrite
#set matrix size, over
x <- matrix(NA, nrow = 100,
            ncol = 100, byrow= T)

for(i in 1:nrow(x)){
    for(j in 1:ncol(x)){
    x[i,j] <- runif(1)
    print(paste(i, j))
    }
}
x
length(x)
```

For-loops: tips

```
for(i in 1:10){
    print(i)
}
```

```
a <- proc.time()[3]
for(i in 1:10){
    print(i)
}
proc.time()[3] - b
```

- Use **print()** to help log where you are in your loop. It’s standard to place the index value **i** in the statement
- Consider using **proc.time()** to time your loops to get a sense of performance.
- Write the contents of your loop first before trying to put into the loop. Test it on a few cases to see if it works.

For-loops: exercise

Fibonacci numbers are defined as

$$F_n = F_{n-1} + F_{n-2}$$

or numbers that are defined as the sum of the preceding two numbers. For example, given an initial sequence of "0, 1", the next five numbers are "1, 2, 3, 5". Write a forloop to get the 100th Fibonacci number.

For-loops: exercise

```
n0 <- 0
n1 <- 1
f <- 0
for(i in 1:99){
    f <- n0 + n1
    n0 <- n1
    n1 <- f
}
option(scipen=999)
f
```

`Answer: 354224848179261915075`

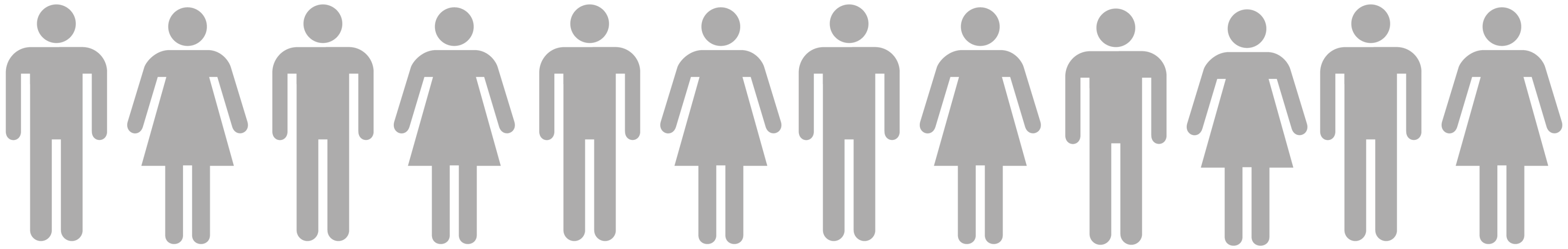
While Loops

Control Structures

A function is applied until a criteria is met.

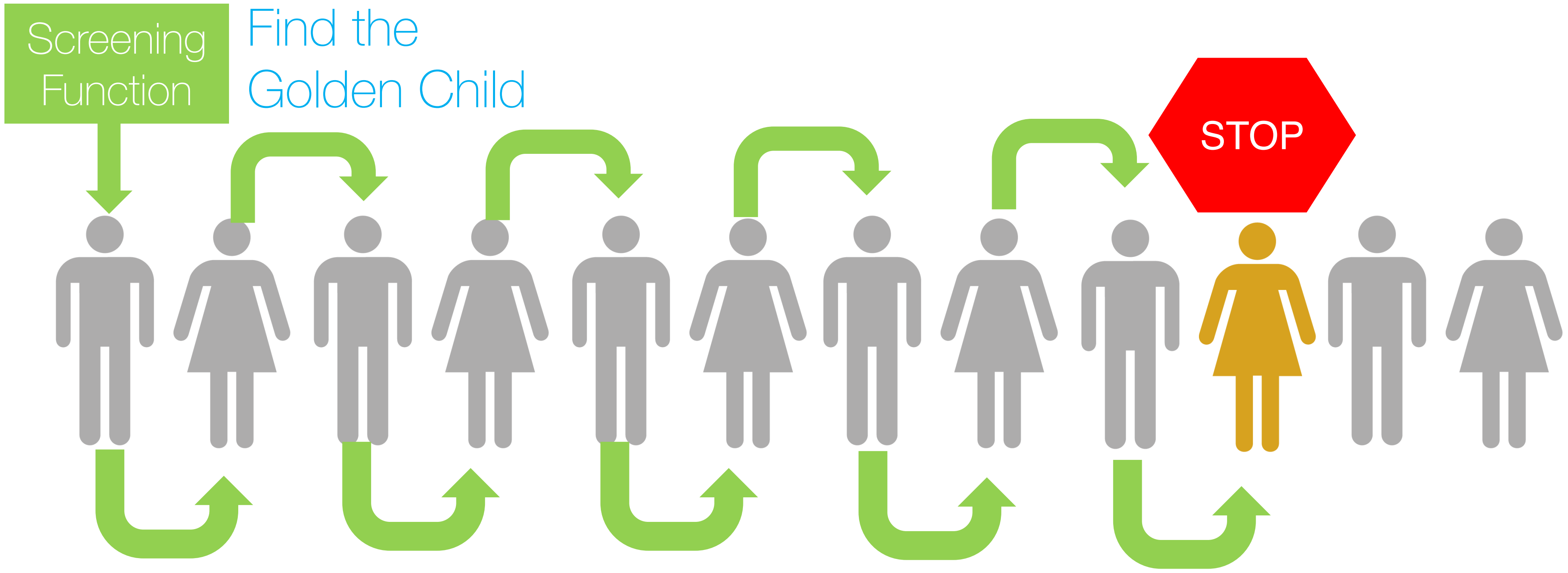
Screening Function

Find the Golden Child



While loops: example

A function is applied until a criteria is met.



While loop construction

- Given a logical statement, while loops continue to iterate so as long as the logical statement is TRUE
- Often times, an index value needs to be advanced within the loop

```
temp <- 0
while(temp < 100){
    print(paste("It’s ", temp,"F, still cool"))
    temp <- temp + 2
}
print("Too hot now.")
```

While loop construction: exercise

Given the function $f(x) = 1/x$, at what value of x does the percent change between x_i and x_{i-1} fall below 0.03%? Assume the function's lower bound is $x=2$.

While loop construction

```
x <- 2
delta <- 1
while(delta > 0.002){
    f0 <- 1/(x-1)
    f1 <- 1/x
    delta <- abs(f1/f0 - 1)
    x <- x + 1
}
x-1
```

Roadmap

- Motivating Story
- Control Structures
- **Functions**
- Code-along

Functions

Intro

User-defined functions are a set of code that accept an input, work in sequence to produce a result. Functions operate with specific parameters, much like the **mean()** and **gsub()** and methods.

```
func <- function(param1, param2,...){
    statements
    return(obj)
}
```

Functions

Intro

Why write your own function?

- More efficiently process and handle data
- Create new algorithms for prediction and research
- Write new libraries to do higher level tasks (example: ggplot2 came from somewhere!)

Function example

Example: Sample Covariance = $cov(X,Y) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})$

```
#Create data
   set.seed(123)
   x <- rnorm(100,10,3)
   y <- rnorm(100,50,4)
```

Function example

Example: Sample Covariance = $cov(X,Y) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})$

```
#The real COV
    cov(x, y)

#Function
    cov2 <- function(x, y){
             z <- sum((x - mean(x)) * (y - mean(y))) / (length(x)-1)
             return(z)
    }

#Check
    cov2(x, y)
```

Function: exercise

Convert the Fibonacci sequence into a function fibseq() with parameters n0, n1, and high (nth number)

```
n0 <- 0
n1 <- 1
f <- 0
for(i in 1:99){
    f <- n0 + n1
    n0 <- n1
    n1 <- f
}
```

Function: exercise

```
fibseq <- function(n0, n1, high){
f <- 0
for(i in 1:(high - 1)){
    f <- n0 + n1
    n0 <- n1
    n1 <- f
}
return(f)
}
```

Roadmap

- Motivating Story
- Control Structures
- Functions
- Code-along: Collaborative Filtering

Collaborative Filtering

Recommendation engines are one of the most used tools to upsell products. Orgs that use rec engines:

- Netflix
- Amazon
- Ebay
- Google
- Dating services
- Pandora/Spotify

Any organization with many products and repeat could benefit from a rec engine

Collaborative Filtering

Recommendation engines often times are driven by preferences as revealed by users on websites:

- Views
- Likes
- Purchases
- Ratings

Collaborative Filtering

People with similar purchasing behaviors are more likely to accept product recommendations.

Person	Syringes	Alcohol pads	Bandaids	Insulin	Neosporin
Jake	1			1	
Amy	1	1		1	
Oliver		1	1		
Sally		1	1		1

Collaborative Filtering

Jake and Amy both purchase **insulin** and **syringes**. Since Amy purchased **alcohol pads**, Jake might also need **alcohol pads.**

Person	Syringes	Alcohol pads	Bandaids	Insulin	Neosporin
Jake	1			1	
Amy	1	1		1	
Oliver		1	1		
Sally		1	1		1

Collaborative Filtering

If someone buys **Neosporin**, what item should be recommended? The result is intuitive and can be quantified

Person	Syringes	Alcohol pads	Bandaids	Insulin	Neosporin
Jake	1			1	
Amy	1	1		1	
Oliver		1	1		
Sally		1	1		1

Collaborative Filtering

Cosine similarity measures the angle between two non-zero vectors. Pearson's Correlation is the mean-centered version.

$$\cos(\theta) = \frac{\sum_{i=1}^{n} X_i Y_i}{\sqrt{\sum_{i=1}^{n} X_i^2}\sqrt{\sum_{i=1}^{n} Y_i^2}}$$

Where X and Y are binary vectors. When binary data, cosine similarity is bound between 0 and 1, where 0 is no correlation

Collaborative Filtering

Item-based filtering requires a **n x n cosine similarity matrix** that allows for ranking similarly purchased items.

	Syringe	Alcohol pad	Bandaid	insulin	neo
syringe	1.00	0.41	0.00	1.00	0.00
alc	0.41	1.00	0.82	0.41	0.58
band	0.00	0.82	1.00	0.00	0.71
insul	1.00	0.41	0.00	1.00	0.00
neo	0.00	0.58	0.71	0.00	1.00

Collaborative Filtering

If someone buys a Bandaid, which item(s) should be paired?

	Syringe	Alcohol pad	Bandaid	insulin	neo
syringe	1.00	0.41	0.00	1.00	0.00
alc	0.41	1.00	0.82	0.41	0.58
band	0.00	0.82	1.00	0.00	0.71
insul	1.00	0.41	0.00	1.00	0.00
neo	0.00	0.58	0.71	0.00	1.00

Collaborative Filtering: Public Policy Use

- Call centers can be automatically fed types of products and alerts to callers when certain services are requested
- Medical centers and social services can provide recommendations of other services

Collaborative Filtering: Considerations

- Types of measures: Ratings, views, purchases
- Dependent on user base: wisdom of the crowds good if people have similar behavior, but pockets of atypical users skew results
- Sparsity: More items higher chance of many blanks.
- Computation: For large number of items, computation will take a while.
    - Codealong full example: 1900 x 17900 matrix requires 4 hours.
    - Codealong 1900 x 500 item example: < 1 min.

For next time

- Read Lecture 4 – Exploratory Data Analysis
- Homework #1 due before Lecture 4 starts