Graficando un histograma de masa invariante en R

R

R es un lenguaje de programación utilizado para estadística y análisis de datos. En este tutorial, introduciremos el análisis de datos con R utilizando los datos de acceso libre del CMS del 2011. Al final de este tutorial, sabrás como crear histogramas de masa invariante e identificar las partículas que aparecen en los histogramas.

Tipos de Datos en R

Hay diferentes tipos de datos en R. Tipos de datos básicos incluyen los siguientes:

- Lógico
- Numérico
- Entero
- Complejo
- Carácter

A las variables se les asigna un tipo sin tener que especificarlo. Las variables numéricas son simplemente números. Las variables enteras se especifican añadiendo una “L” al final del número:

> a = 5L

Las variables complejas son números complejos, se pueden declarar utilizando el siguiente formato:

> b = 5 + 3i

> d = 8 + 0i

Las variables lógicas pueden tener los valores VERDADERO o FALSO. Puedes asignar una condición a una variable lógica:

> c = 3 > 5

Las variables de caracteres son letras o frases, aunque también pueden añadirse números y símbolos entre comillas.

> cr = “3!”

Vectores

Las variables pueden ser tanto escalares como vectores. Un vector escalar es una variable que contiene un único número. Los vectores pueden contener más de un valor, y se crean de la manera siguiente:

> a = c(2, 3, 5)

Los vectores pueden ser cualquiera de los tipos de datos básicos mencionados anteriormente. También podemos aplicar condiciones a los vectores para crear un vector lógico, utilizando la siguiente sintaxis:

```
> a = c(2, 5, 8, 3, 9)

> b = a > 3
```

podemos crear un vector con los siguientes valores:

```
[1] FALSE TRUE TRUE FALSE TRUE
```

Para accede a un elemento particular de un vector, podemos utilizar el nombre del vector y el número de elemento entre corchetes, contando desde el 1. El comando:

```
> a[1]
```

produce el siguiente resultado:

```
[1]  2
```

Que es el primer elemento del vector "a".

Podemos también seleccionar los valores de un vector aplicando una condición lógica, de modo que sólo los elementos del vector que cumplen con la condición se muestren:

```
> a[a>3]

[1] 5 8 9
```

Matrices

En R, Podemos crear una matriz a partir de un vector. Las matrices son estructuras de datos de dos dimensiones. La manera de crear una matriz es especificando los valores que contendrá, el número de filas y columnas y la manera en que se llenará la matriz (en la dirección de las filas o columnas).

```
> a = c(1:9)
```

Crea una matriz con valores del 1 al 9.

```
> A = matrix(y, nrow=3, ncol=3, byrow=TRUE)

> print(A)

     [,1] [,2] [,3]

[1,]    1    2    3

[2,]    4    5    6

[3,]    7    8    9
```

Para accesar un elemento de la matriz tenemos que especificar los valores de la fila y la columna después del nombre de la matriz:

> A[2,3]

Devuelve el valor 6, que está en la segunda fila y la tercera columna.

Podemos accesar a los valores de una columna entera al mismo tiempo, dejando un espacio en blanco donde el número de la fila estaría, y viceversa. A[2,] nos da el valor de la segunda fila.

Las matrices también pueden ser creadas utilizando condiciones para otras matrices, y también pueden accesarse utilizando condiciones lógicas. Accesar valores mediante condiciones lógicas da como resultado un vector con los valores de la matriz para los cuales la condición es verdadera.

> a = c(1:25)

Creando un vector con valores del 1 al 25

> A = matrix(a, nrow=5, ncol=5, byrow=TRUE)

Creando una matriz a partir del vector a, con 5 filas y 5 columnas, y rellenándola en el sentido de las filas

> C = A > 12

Creando una matriz lógica a partir de una condición

> C

[,1] [,2] [,3] [,4] [,5]

[1,] FALSE FALSE FALSE FALSE FALSE

[2,] FALSE FALSE FALSE FALSE FALSE

[3,] FALSE FALSE TRUE TRUE TRUE

[4,] TRUE TRUE TRUE TRUE TRUE

[5,] TRUE TRUE TRUE TRUE TRUE

Mostrando la matriz lógica, los valores que se muestran son "TRUE" (verdaderos) cuando los elementos correspondientes de la matriz A cumplen la condición A>12.

> A[C]

[1] 16 21 17 22 13 18 23 14 19 24 15 20 25

Seleccionar valores de la matriz A utilizando la matriz C, esto nos da los valores de A para los cuales los valores correspondientes de C son verdaderos. Esto es, los valores para los que A > 12, en la forma de un vector.

Arrays

Las series (“Arrays”) son estructuras de datos similares a las matrices, pero pueden tener más de dos dimensiones. Puedes crearlas a partir de un vector y especificar el número de dimensiones de la serie de datos.

```
> a = c(1:27)
```

Creando un vector con valores del 1 al 27

```
> A = array(a, dim=c(3,3,3))
```

Creando un array a partir del vector a, que tiene 3 matrices de dimensiones 3x3.

```
> A
```

Mostrando el array

```
, , 1

     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9

, , 2

     [,1] [,2] [,3]
[1,]   10   13   16
[2,]   11   14   17
[3,]   12   15   18

, , 3

     [,1] [,2] [,3]
[1,]   19   22   25
[2,]   20   23   26
[3,]   21   24   27
```

Listas

Las listas son como los vectores, pero pueden tener diferentes tipos de datos al mismo tiempo, y también pueden contener vectores.

```
> l = list(c(1,2,3),'a', 1, 1+5i)
```

Marcos de datos

Los marcos de datos (“Data frames”) son como listas de vectores que tienen la misma longitud. Se utilizan para registrar datos de forma tabular. Puedes crear un marco de datos utilizando el siguiente código:

```
> datos = data.frame(
+ Nombre = c("James", "David"),
+ Genero= c('M','M'),
+ Edad = c(20, 23))
```

Mostrar el marco de datos – print(datos) – da como resultado lo siguiente:

```
  Nombre Genero Edad
1 James      M    20
2 David      M    23
```

Si quieres accesar una columna en particular en un marco de datos, puedes utilizar el nombre del marco de datos seguido del signo de dólar $ seguido del nombre de la columna. En el ejemplo anterior, si queremos acceder a los nombres, podemos escribir:

```
> datos$Nombre
```

Lo cual dará como resultado un vector con los nombres escritos en él. Si queremos, por otro lado, acceder sólo a una fila en particular, podemos utilizar lo siguiente:

```
> datos[1,]
```

Lo cual dará la primera fila del marco de datos.

También es posible importar datos desde otros archivos, como archivos CSV.

Importando datos desde archivos CSV

Puedes importar archivos CSV (comma separated values: valores separados por coma) en R, y analizar los datos contenidos en estos archivos. En este tutorial vamos a utilizar como ejemplo los datos de acceso libre publicados por la colaboración CMS del CERN. Estos datos pueden ser encontrados en el siguiente link: http://opendata.cern.ch/record/545, en el sitio web de datos de acceso libre del CERN. Puedes descargar los diferentes archivos CSV y guardarlos en tu computadora.

Para importar los archivos de datos en R utilizamos el siguiente comando:

> data = read.csv("C:/Data/Jpsimumu.csv")

Aquí estamos guardando los datos contenidos en el archivo Jpsimumu.csv en una variable llamada "data". Asegurate de utilizar la dirección correcta en tu computadora, dependiendo de en dónde has guardado los archivos.

Los archivos de datos contienen miles de entradas, así que en lugar de mostrar todo el archivo, podemos echar un vistazo utilizando la siguiente función:

> head(data)

Esto mostrará las primeras 6 filas del marco de datos.

Calculando la masa invariante

Vamos a utilizar el archivo de datos Jpsimumu. Este archivo contiene eventos del detector CMS donde dos muones fueron detectados.

Primero, lee los datos y escribelos en la variable jpsi,

> jpsi = read.csv("Jpsimumu.csv")

Puedes escribir

> head(jpsi)

Para ver las primeras 6 filas de datos en la variable, para asegurarte de que todo está bien. Puedes ver los valores para la energía (E), el momento (p), la pseudo-raidez eta (η) y el ángulo phi(φ). El valor para la masa no se muestra, pero podemos calcularlo a partir de los valores para la energía y el momento.

Masa invariante

En física relativista, la masa invariante es la masa del Sistema que es equivalente a la masa en reposo de este. Si calculamos la masa invariante de un sistema de un Sistema de partículas que son producto del decaimiento de una partícula, entonces la masa invariante será lo mismo que la masa de la partícula original.

La masa invariante puede calcularse por medio de la siguiente ecuación:

$$M = \sqrt{(\Sigma E)^2 - \|\Sigma \boldsymbol{p}\|^2}$$

donde M es la masa invariante, ΣE es la energía total y $\Sigma \boldsymbol{p}$ es el momento lineal total.

Para calcular la masa invariante en el código, vamos a utilizar los valores de px, py y pz y los valores de la energía para las dos partículas. Primero, vamos a calcular la suma vectorial de los momentos de las partículas 1 y 2. Para calcular la suma vectorial, tenemos que sumar individualmente cada componente de los vectores:

```
> pxt = jpsi$px1+jpsi$px2

> pyt = jpsi$py1+jpsi$py2

> pzt = jpsi$pz1+jpsi$pz2
```

Después calculamos la magnitude del vector, con el Código siguiente:

```
> ptotal = sqrt( pxt^2 + pyt^2 + pzt^2 )
```

Aquí utilizamos la función sqrt( ) para obtener la raíz cuadrada de la suma de todos los componentes al cuadrado, y la guardamos en la variable ptotal.

También podemos definir una función para que haga este cálculo. Puedes definir tus propias funciones utilizando la siguiente sintaxis:

```
mifuncion = function(arg1, arg2…)

{

comandos

return(a)

}
```

Para definir una función para calcular la magnitud de un vector, necesitamos tomar tres argumentos y un valor de retorno. Podemos llamarle a esta función magnitude( ) -magnitud en inglés- y definirla como sigue

```
> magnitude = function(x, y, z) {

+ m = sqrt(x^2 + y^2 + z^2)

+ return(m) }
```

Ahora para hacer el cálculo utilizando la función simplemente pasamos los argumentos, que son los tres componentes del momento lineal en las diferentes direcciones espaciales:

```
> ptotal = magnitude(pxt, pyt, pzt)
```

También podemos definir una función para hacer el cálculo de la masa invariante de un sistema consistente en dos partículas.

```
> invmass = function(px1, px2, py1, py2, pz1, pz2, E1, E2){
+ px = px1+px2
+ py = py1+py2
+ pz = pz1+pz2
+ E=E1+E2
+ ptotal = magnitude(px, py, pz)
+ mass = sqrt(E^2 - ptotal^2)
+ return(mass)
+ }
```

Aquí primero indicamos el nombre de la función y sus argumentos, luego obtenemos la suma vectorial de p1 y p2 y luego la suma de las energías, luego obtenemos la magnitud del momento total y finalmente calculamos la masa invariante utilizando la ecuación mencionada anteriormente.

Ahora podemos calcular la masa invariante utilizando la función que acabamos de definir:

```
> jpsimass = invmass(jpsi$px1, jpsi$px2, jpsi$py1, jpsi$py2, jpsi$pz1, jpsi$pz2, jpsi$E1, jpsi$E2)
```

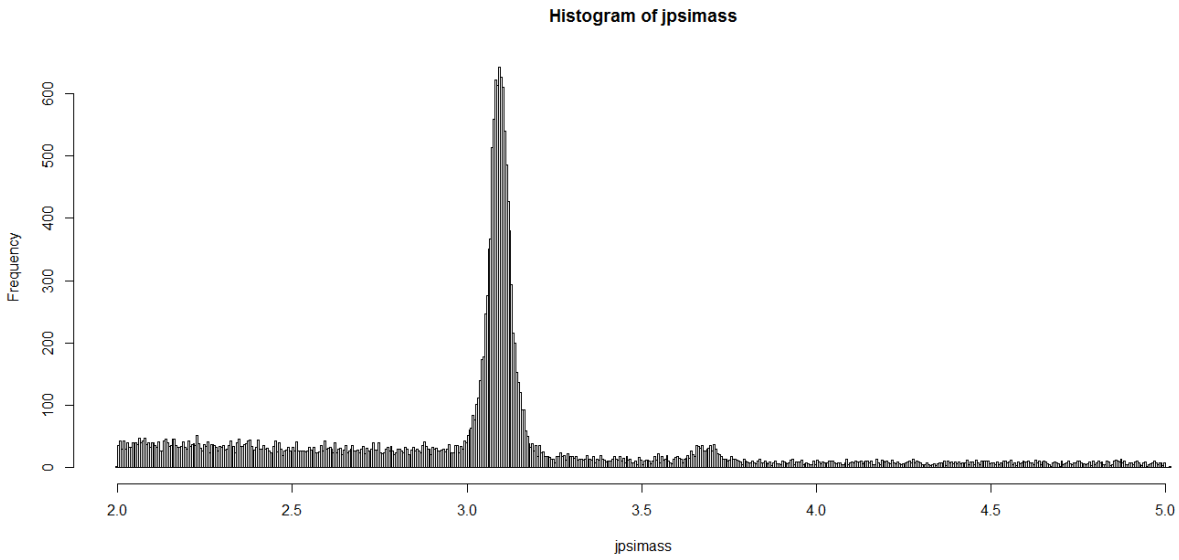
Y podemos mostrar los primeros valores de la masa utilizando la función head(jpsimass).

Graficando un histograma

Ahora que tenemos los valores para la masa invariante, podemos graficar un histograma para ver todos los datos. Podemos hacer esto con la función hist( ).

```
> hist(jpsimass, breaks = 500)
```

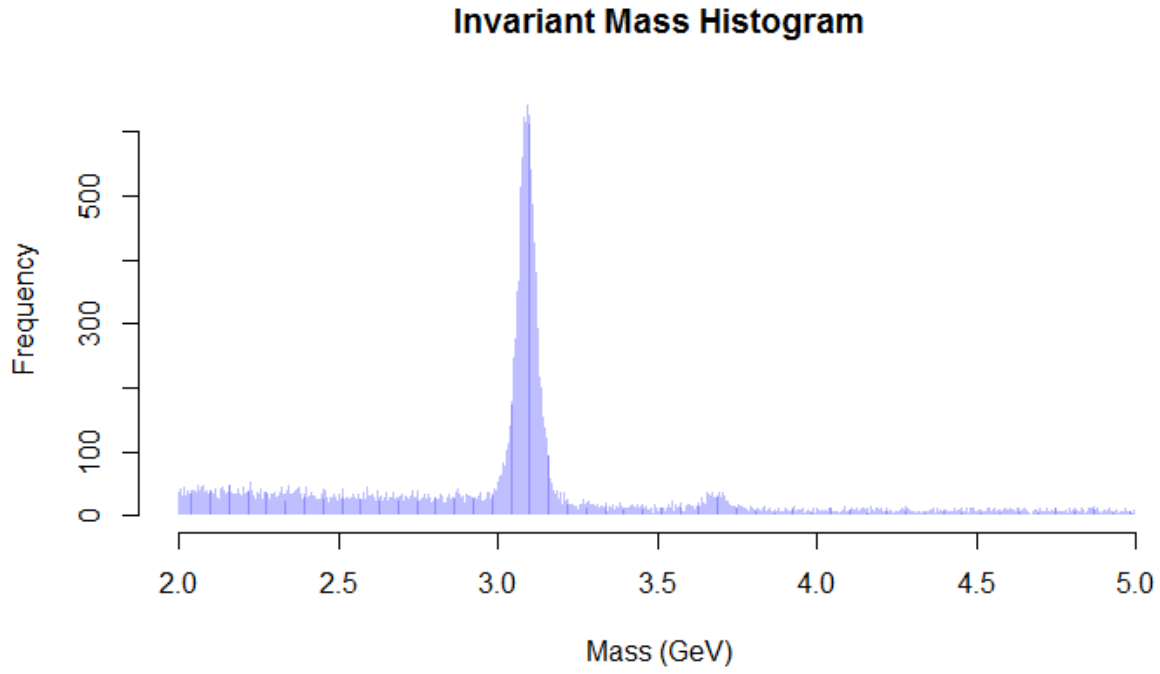
Aquí, “breaks” indica el número de Casillas que será creado. Puedes experimentar cambiando el número de casillas y ver como puedes visualizar mejor los datos. El histograma se verá más o menos como este:



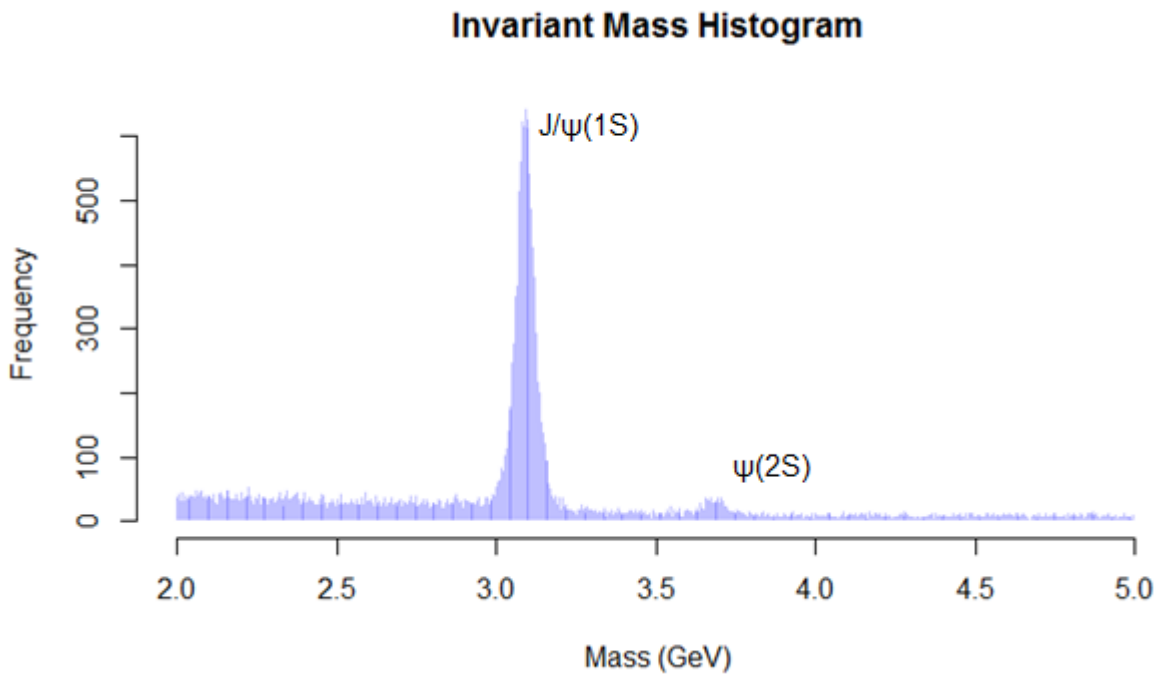
Podemos cambiar los colores y el título del histograma con diferentes comandos. Los siguientes comandos:

hist(jpsimass, breaks = 500, xlab = "Mass (GeV)", lty="blank", col=rgb(0,0,1,1/4),
main="Invariant Mass Histogram")

Aquí 'xlab' es el título para el eje x, lty="blank" indica que no hay bordes para las Casillas en el histograma, 'main' es el título del histograma y 'col' indica el color (los números son los valores para los canales rojo, verde, azul y alfa). Este código dará algo como esto:



Ahora podemos ver claramente un pico grande alrededor de 3.1 GeV y un pico más pequeño alrededor de 3.7 GeV. Estos dos picos corresponden a la masa de dos partículas que tienen decaimiento dimuónico (decaen en un muón y un anti-muón). Podemos buscar por una partícula como esa en el Particle Data Group (http://pdg.lbl.gov/), y encontramos que esas dos partículas son mesones: el mesón J/ψ(1S) y el mesón ψ(2S). Puedes ver las dos partículas en la siguiente gráfica:



Ahora puedes explorar el resto de los archivos y tratar de identificar las partículas que aparecen en cada histograma. Para identificar las partículas, toma en cuenta la masa y el modo de decaimiento y busca en la base de datos del Particle Data Group.