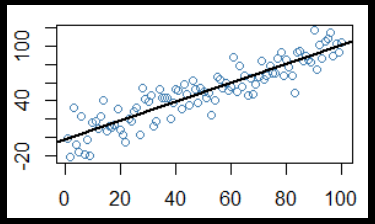
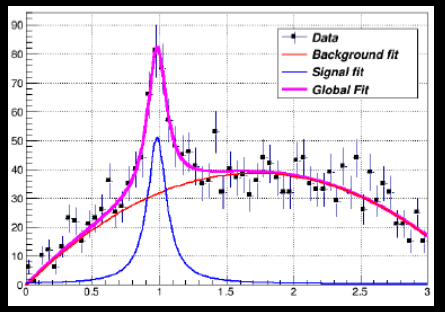
Temas avanzados en física computacional
Análisis de datos





Semestre

2016-I

Clase-4

José Bazo

jbazo@pucp.edu.pe

Contenidos del curso

- ✓ Introducción al análisis de datos y data science
- ✓ Lenguaje de programación R
- ✓ ROOT Data Analysis Framework

4. Manipulación y visualización de datos
5. Modelamiento estadístico
6. Machine Learning
7. TMVA (Toolkit for Multivariate Data Analysis)

4. Manipulación y visualización de datos

Bibliografía

Coursera: Getting and Cleaning data, Exploratory Data Analysis
at Johns Hopkins University:
https://www.coursera.org/learn/data-cleaning/

Procesamiento de datos

From raw to processed data

Filtro ➔ Selección ➔Cálculo de nuevas variables ➔Archivo en formato final

Procesamiento:

- Solo una vez (mundo ideal), usualmente muchas (preservar archivo original)
- Merging, subsetting, transforming

- Registrar todos los pasos (receta o único script), desde raw (input) hasta tidy dataset (output) sin parámetros libres
- Study design: descripción de cómo se recolectaron los datos
- Code book: describir cada variable, sus valores y unidades

Procesamiento de datos

Ej. **LHC** produce produces ~6×10^8 colisiones/s, generando ~1PB/s

Sistema electrónico rápido de pre-selección (**trigger**) reduce 10^{-4} (100 GB/s)
Luego 15000 cores seleccionan 1% de datos restantes para ser analizados.

Worldwide LHC Computing Grid (**WLCG**)

TIER 0 (Centro de datos @ CERN): 73 000 cores (<20% de capacidad total del Grid):

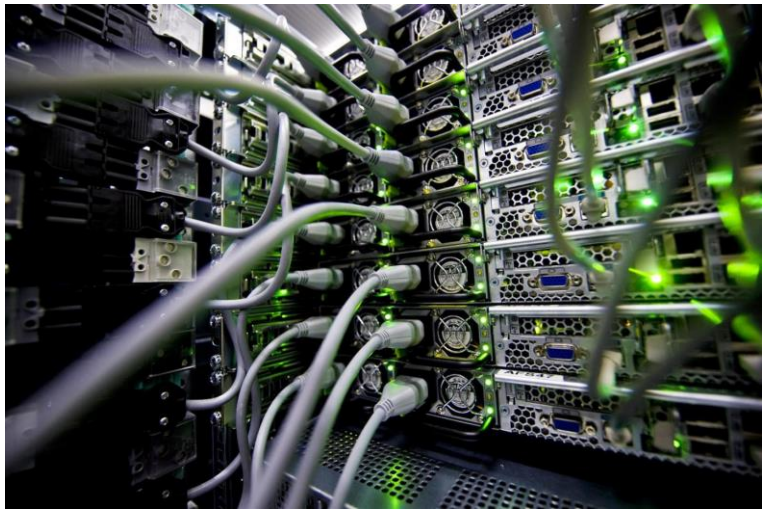
- Junta y almacena de largo plazo de datos raw: Tape
- Reconstrucción inicial de datos
- Distribuye (10 Gb/s) datos a:

TIER 1 (13 centros):

- Almacenamiento permanente
- Reprocesamiento de datos y análisis

TIER 2 (155 centros):

- Simulación
- Análisis final de los usuarios.

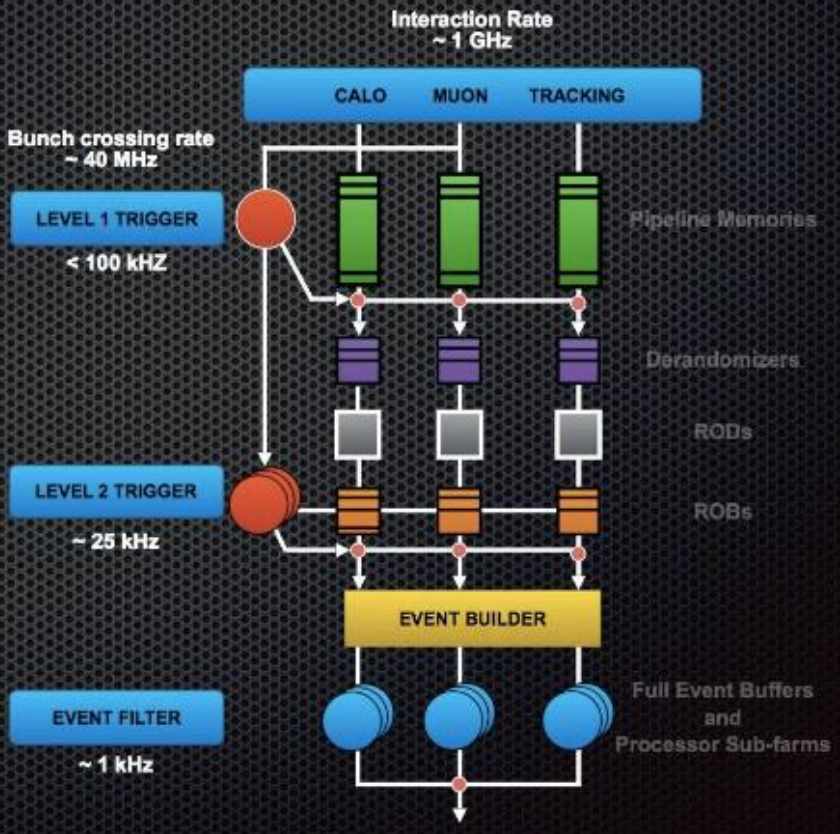


Procesamiento de datos

http://atlas.web.cern.ch/Atlas/GROUPS/DAQTRIG/TDR/V1REV1/L1TDR_all.pdf

The Trigger/DAQ System

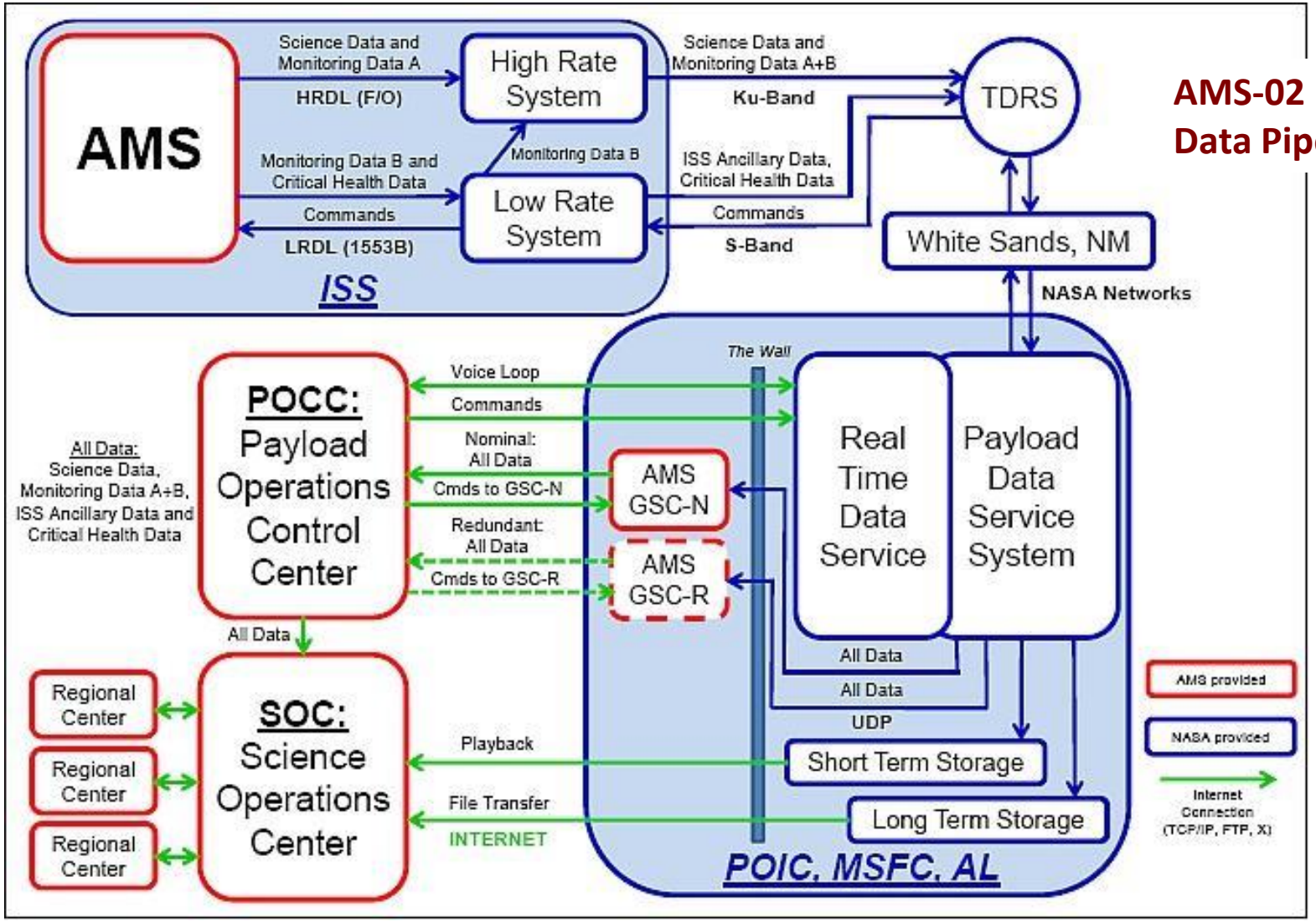
- 3 levels of online event selection
- Bunches of protons cross every 25 ns (40 MHz rate)
  - Reduce this to ~1 kHz for permanent storage
  - Rejection factor of 1e6
- Goal: retain efficiency of processes sought for in ATLAS



ROBs: Readout Buffers
RODs: Readout Drivers

3

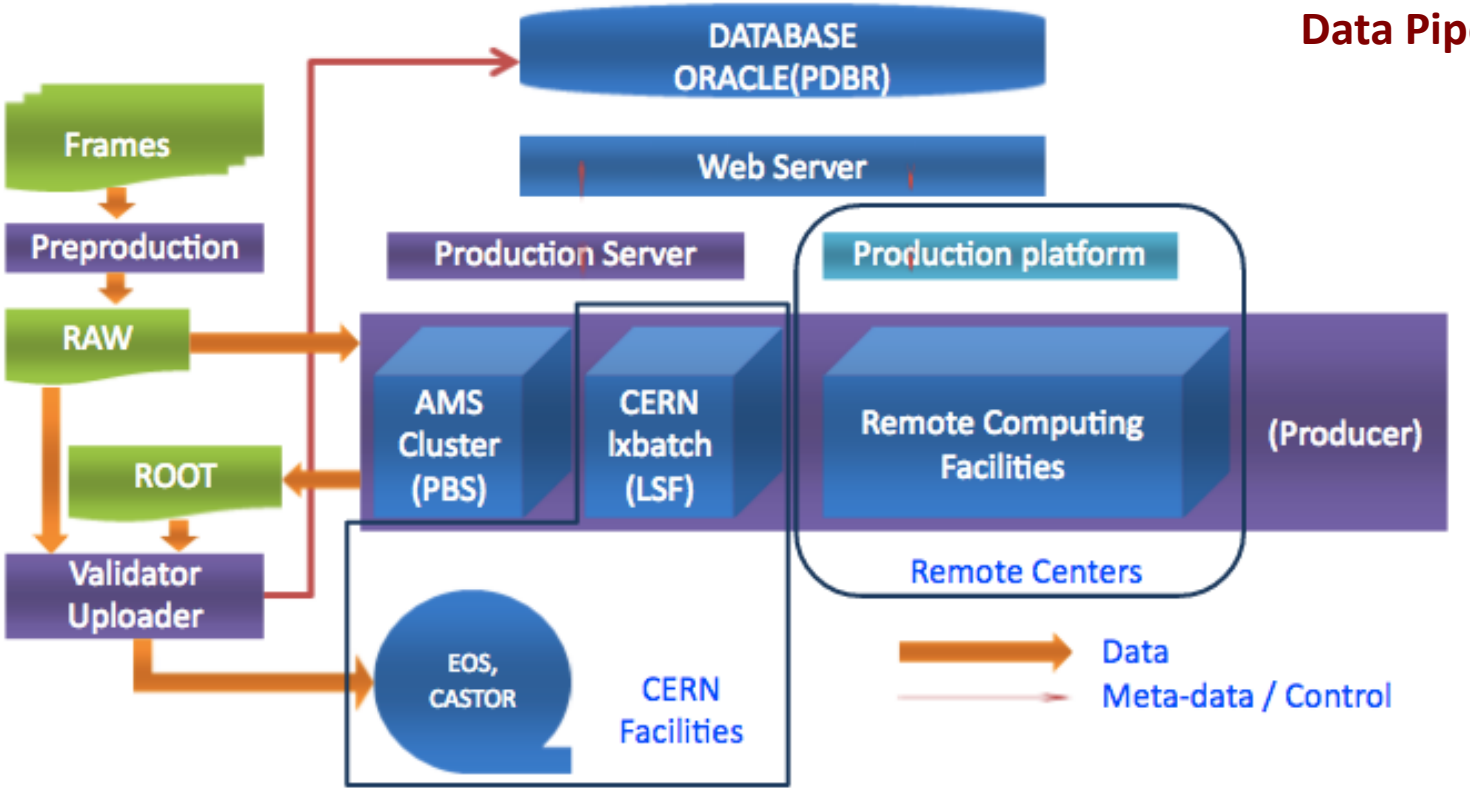
Procesamiento de datos



AMS-02 Data Pipeline

Procesamiento de datos

AMS-02
Data Pipeline



Leyendo datos

Directorio de trabajo:

getwd()

setwd() Path: Relativo: setwd("../Data")
Absoluto: setwd("/Users/Home/DataScience")
windows setwd("C:\\Users\\Home")

if(**file.exists**("dirName")) { **dir.create**("dirName") }

Leyendo datos

Leer de internet:

fileUrl<-"http://opendata.cern.ch/record/302/files/dielectron-Jpsi.csv"
download.file(fileUrl, destfile="./data/data.csv", method="curl")
list.files("./data")

para Mac

Registrar la fecha en que se bajó el archivo (podría cambiar)
dataDownloaded<-date()

Leer archivo local:

cerndata<-**read.table**("./data/data.csv", sep=",", header = TRUE)

Alternativa para csv:
cerndata<-**read.csv**("./data/data.csv")

head(cerndata)

Otros parámetros:
nrows=2 , lee 2 primeras filas
col.names
comment.char="#"
skip=2, salta 2 primeras líneas antes de comenzar lectura

Leyendo datos

Leer archivos Excel:

fileUrl<-" https://risweb.st-andrews.ac.uk/portal/files/241062447/Figure_1ab.xlsx"

download.file(fileUrl, destfile="./data/data2.xlsx", method="curl")

```
install.packages("xlsx")
library(xlsx)

data2<-read.xlsx("./data/data2.xlsx", sheetIndex=1, header=TRUE)
```

Leer solo parte del archivo:

```
colIndex<-c(1,3)
rowIndex<-1:15
data2_subset<-read.xlsx("./data/data2.xlsx", sheetIndex=1, header=TRUE,
                        colIndex=colIndex, rowIndex=rowIndex)
```

Otras opciones:

colClasses = rep("numeric",9), stringsAsFactors=FALSE

Leyendo datos

Leer archivos XML (extensible markup language) (para web scraping):

install.packages("**XML**")
library(XML)

"self-describing data abstraction for the storage and manipulation of multidimensional data in a discipline-independent fashion"

```
fileUrl<-"http://xml.comp-phys.org/example.xml"
doc<-xmlTreeParse(fileUrl, useInternal=TRUE)
rootNode<-xmlRoot(doc)
xmlName(rootNode)

names(rootNode)

Acceder directamente a partes del documento
rootNode[[1]]
rootNode[[2]][[1]]

xmlSApply(rootNode,xmlValue)
```

XML:
Markup: labels, estructura texto
Content: texto del documento

Tags: general labels:
Start: <section>
End: </section>
Attributes:

CDF: common data format

Leyendo datos

Leer archivos XML (extensible markup language) (para web scraping):

Extraer contenidos:

xpathSApply(rootNode,"//MEAN", xmlValue)

Label del documento

- */node* Top level node
- *//node* Node at any level
- *node[@attr-name]* Node with an attribute name

Extraer contenidos con atributos:

xpathSApply(rootNode,"//SCALAR_AVERAGE[@name='Energy']",getChildrenStrings)

Leyendo datos

Leer archivos JSON (Javascript Object Notation) (similar a XML):

```
install.packages("jsonlite")
library(jsonlite)

fileUrl<-http://opendata.cern.ch/record/306/files/Zee.json
jsonData<-fromJSON(fileUrl)

names(jsonData)
```

Nested objects:
jsonData$name$name2

Escribir data frames a JSON:
myjson<-toJSON(dataframe, pretty=TRUE)

JSON:
Datos almacenados como:
Numbers (double)
Strings (double quoted)
Boolean (true, false)
Array (ordered, comma separated enclosed in [])
Object (unordered, comma separated collection of key:value pairs in {})

Leyendo datos

Leer mySQL (Structured Query Language):
open-source relational database management system

```
install.packages("RMySQL")
library(RMySQL)

ucscDb<-dbConnect(MySQL(), user="genome", host="genome-mysql.cse.ucsc.edu")
Result<-dbGetQuery(ucscDb,"show databases;")

hg38<-dbConnect(MySQL(), user="genome", db="hg38", host="genome-mysql.cse.ucsc.edu")
allTables<-dbListTables(hg38)
dbListFields(hg38, "affyU133")
dbGetQuery(hg38, "select count(*) from affyU133")
affyData->dbReadTable(hg38,"affyU133")

dbDisconnect(ucscDb)
```

Giordon Stark: "I once tried to take a ROOT file that was 2 gigabytes and store it in a MySQL database (23 gigabytes)"

mySQL:
Datos estructurados en:
Bases de datos
Tablas dentro de la base
Campos dentro de tablas
Cada fila es un record

Leyendo datos

Crear archivos HDF5 (Hierarchical Data Format):
Store and organize large amounts of data

```
source("http://bioconductor.org/biocLite.R")
biocLite("rhdf5")
library(rhdf5)

Hdf5 = h5createFile("example.h5")
Hdf5 = h5createGroup("example.h5","energy")
Hdf5 = h5createGroup("example.h5","time")
Hdf5 = h5createGroup("example.h5","energy/corrected")
h5ls("example.h5")

df = data.frame(1L:5L, seq(0,1,length.out=10),c("a","b","x","z"),
stringsAsFactors=FALSE)
h5write(df, "example.h5","df")
```

HDF5:
Grupos que contienen datasets y metadata
Datasets: arreglos multidimensionales de elementos de datos con metadata

Leyendo datos

Leer archivos HDF5 :

read = **h5read**(“example.h5”,”df”)

http://www.illustris-project.org/data/

http://www.lofar.org/wiki/doku.php?id=public:hdf5

http://software.icecube.wisc.edu/offline/projects/hdfwriter/index.html

Leyendo datos

Convertir un archivo de root a hdf5

http://www.rootpy.org/commands/root2hdf5.html

root2hdf5

> **Note**
>
> To use this command you must have HDF5, PyTables, NumPy and root_numpy installed.

Modo de uso:

```
$ root2hdf5 ../rootpy/testdata/test_tree.root
INFO:rootpy.root2hdf5] Converting ../rootpy/testdata/test_tree.root ...
INFO:rootpy.root2hdf5] Will convert 1 tree in this directory
INFO:rootpy.root2hdf5] Converting tree 'test' with 1000 entries ...
INFO:rootpy.root2hdf5] Created ../rootpy/testdata/test_tree.h5
[?1034h
```

Fuentes de Datos

Fermi
IceCube
ASDC

CERN

Condensed Matter research data

NOAA
Earth Data

INEI

Baltimore City

Resumiendo datos

```
fileUrl<-"http://opendata.cern.ch/record/302/files/dielectron-Jpsi.csv"
download.file(fileUrl, destfile="./data/cern.csv")
cern<-read.csv("./data/cern.csv")

dim(cern)

head(cern, n=5)   #ver primeros datos
tail(cern, n=5)   #ver últimos datos

summary(cern)  #resumen

str(cern)   #estructura

quantile(cern$E1,na.rm = TRUE, probs = c(.01,0.1,0.5,0.99))

table(cern$Run, useNA="no")   #cuentas según factores
table(cern$E1>150,cern$Run)

any(is.na(cern))    sum(is.na(cern))   colSums(is.na(cern))  #verificar valores faltantes
```

Resumiendo datos

Tamaño del juego de datos:

```
print(object.size(cern),units = "Mb")
```

Ejemplo con datos de R:

```
data(UCBAdmissions)
adm=as.data.frame(UCBAdmissions)
summary(adm)

xtabs(Freq~Gender,data=adm)        # dar resultado de fórmula (antes de ~) por categorías
xtabs(Freq~Admit+Dept,data=adm)   #factores clasificadores se añaden con +


ftable(xtabs(Freq~.,data=adm))    # flat table
```

Manipulando datos

Ejemplo con data.frame:

```
set.seed(7)
df<-data.frame("var1"=sample(1:5), "var2"=sample(6:10), "var3"=sample(11:15))
df$var1[c(1,5)]=NA
```

Subsetting

```
df[,1]      df[,"var1"]
df[1:3,"var3"]

df[ df$var2<=9 & df$var3>12, ]
```

```
df$var5<-c("a","a","b","c","a")

df[df$var5 %in% c("a","b"),]
```

Si hay valores faltantes:

```
df[ which(df$var1>2),]
```

Manipulando datos

Ordenar

sort(df$var2)

sort(df$var2, decreasing=TRUE)

sort(df$var1, na.last=TRUE)

df[order(df$var2),]

Usando paquete **plyr**, library(plyr)

arrange(df,var2)

arrange(df,desc(var2))

Añadir columnas:

df$var4 <- rnorm(5,10,2)

df<-cbind(df,runif(5,0,10))

Añadir filas:

df<-rbind(df,runif(5,-10,10))

Remover columnas:

df$var4 <- NULL
df[1:2] <- list(NULL)

Remover filas

df <- df[-6),]

data.table

Hereda de data.frame, sus funciones son aceptadas y es más rápido

```
library(data.table)

DT=data.table(x=rnorm(9), y=rep(c("a","b","c"),each=3), z=rnorm(9))
```

Data tables en memoria:
tables()

Subsetting rows:

```
DT[1,]
DT[DT$y=="b",]
DT[c(4,7,9),]
```

```
> DT
           x y          z
1:  0.82257434 a  0.6319952
2: -1.02419113 a  0.3450599
3: -1.87683433 a -1.0382439
4:  0.41529243 b -1.1990756
5: -1.67612210 b  1.3490184
6: -0.55506509 b  1.5612518
7:  0.01490873 c  1.2543759
8:  0.82769607 c -1.1499443
9:  0.32167231 c -0.2557623
> tables()
     NAME NROW NCOL MB COLS  KEY
[1,] DT      9    3  1 x,y,z
Total: 1MB
```

data.table

Subsetting columns: (acepta expresiones)

```
DT[,mean(x)]
DT[,list(max(x),min(z))]
DT[,table(y)]
```

Añadir una nueva columna

```
DT[,w:=(x-z)^2]
DT[,w2:={ tmp<-x+z; log(tmp+10) } ]
DT[,a:=x>0 & z>0]
DT[,b:=mean(x+z),by=a]
```

Contar:
DT[, **.N**, by=a]

Ordenar en orden ascendiente:
setkeys[DT,w2,x]

Juntar data tables:
merge(DT1, DT2)

Cuidado, si se copia una tabla DT2<-DT y se modifica la copia (DT2[,x:=1]) también se modifica el original y viceversa

Crear nuevas variables

Secuencias de números:

seq(1,30, by=3) seq(1,10, length=5) seq(along = c(1,5,2,6,7,6))

Variables binarias:

```
cern$HE = ifelse(cern$E1>100, TRUE, FALSE)
```

Variables categóricas:

```
cern$Egroups = cut(cern$E1, breaks=quantile(cern$E1, probs = c(0,0.1,0.5,0.9,1) ) )

library(Hmisc)
cern$Egroups = cut2(cern$E1,c(0,50,150,200))  #grupos con valores límites
cern$Egroups = cut2(cern$E1,g=4) # grupos de cuantiles
cern$Egroups = cut2(cern$E1,m=200) # grupos de por lo menos m observaciones
```

Variables de factores

```
cern$runf <- factor(cern$Run)
```

Otros cálculos

```
tapply(cern$E2,cern$Egroups,mean)  #promediar valores por categoría

split(cern$E2, cern$Egroups)    #dividir por categoría

lapply(split(cern$E2, cern$Egroups), mean) #promediar valores por categoría

library(plyr)
ddply(cern,.(Egroups),summarize,mean=mean(E2))
```

Funciones comunes

- `abs(x)` absolute value
- `sqrt(x)` square root
- `ceiling(x)` ceiling(3.475) is 4
- `floor(x)` floor(3.475) is 3
- `round(x,digits=n)` roun(3.475,digits=2) is 3.48
- `signif(x,digits=n)` signif(3.475,digits=2) is 3.5
- `cos(x), sin(x)` etc.
- `log(x)` natural logarithm
- `log2(x), log10(x)` other common logs
- `exp(x)` exponentiating x

Juntando datos

```
fileUrl1<-"http://opendata.cern.ch/record/302/files/dielectron-Jpsi.csv"
fileUrl2<-"http://opendata.cern.ch/record/301/files/dimuon-Jpsi.csv"

download.file(fileUrl1, destfile="./data/cern1.csv")
download.file(fileUrl2, destfile="./data/cern2.csv")

cern1<-read.csv("./data/cern1.csv")
cern2<-read.csv("./data/cern2.csv")

intersect(names(cern1),names(cern2))

merge(cern1, cern2, by.x="Run", by.y="Run", all=TRUE)   #juntar archivos
```

dplyr

```
install.packages("dplyr")
library(dplyr)

select(cern,3:6)
select(cern,E1:pz1)      select(cern,-(E1:pz1))

filter(cern,E2>100)

arrange(cern, desc(E1))

rename(cern,energy1=E1)

mutate(cern,totE=E1+E2)  #añadir columna

group_by(cern,Egroups)

summarize(group_by(cern,Egroups), Etotm=mean(E1+E2), ptmax=max(pt1))
```