

Memo R basics

Plan

I. R Basics

- A. What is R
- B. Ressources
- C. Basic knowledge (variable type, classic functions)

II. R objects for NGS

- A. Data bases
- B. Genomic object
- C. Specific tools

III. R plots

R Basics



What is R?

- ❑ A calculator

```
## All mathematical / statistical functions implemented in the basics  
 $(5+5)*10^4$   
sqrt(16) / factorial(50)
```

- ❑ A programming language

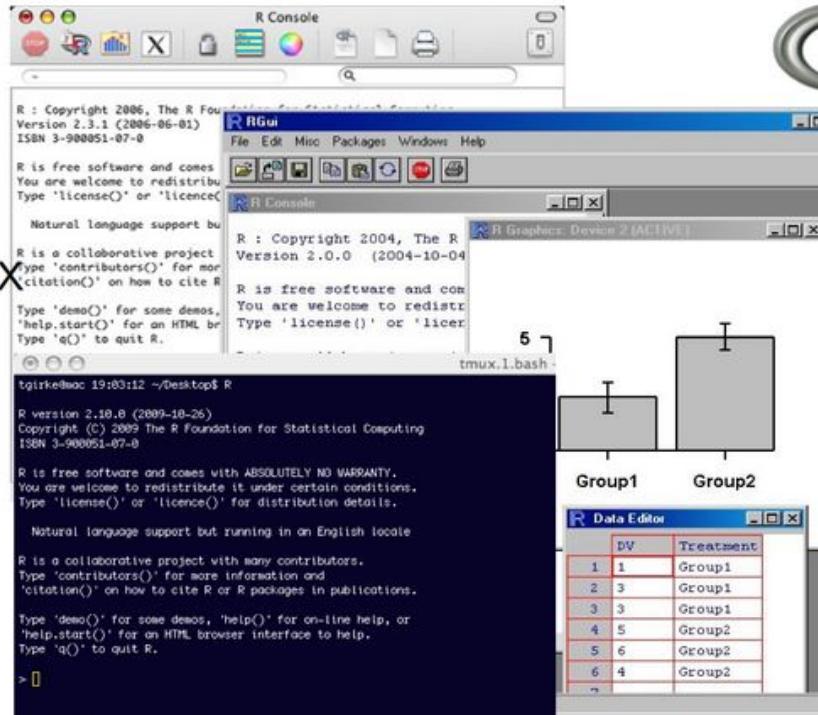
```
## Automatisation of repetitive tasks / functions  
  
for (i in 1:50){  
  print(i^2)  
}
```

```
hello.world <- function(){  
  print("Hello World")  
}
```

- ❑ An open source software
- ❑ A very active community that have already provided > 4000 extensions.

How to play with R? Console

R Gui: OS X

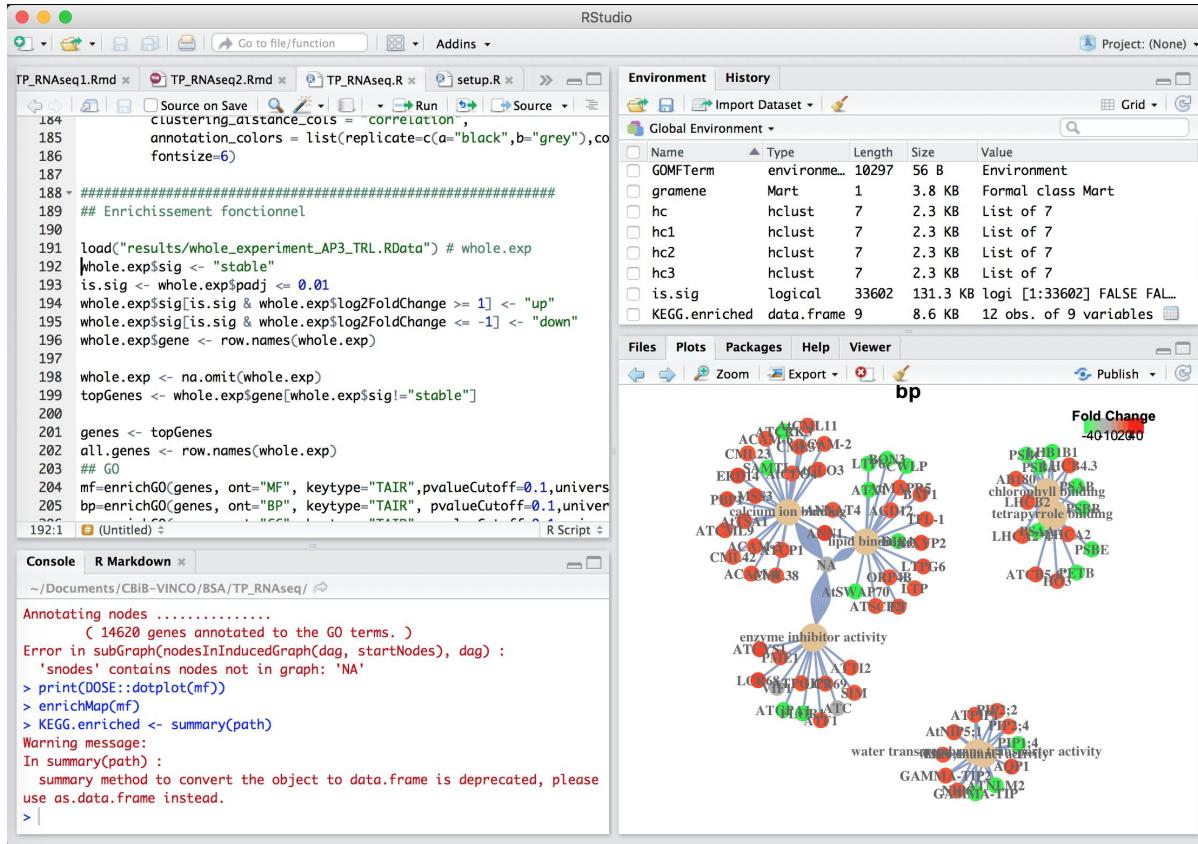


Command-line R: Linux/OS X
Environnements R par défaut

R Gui: Windows

How to play with R? RStudio

Interactive, integrated, visual environment



Ressources

- ❑ R Packages
 - ❑ When downloading R, it comes with “basic” functions
 - ❑ Due to R popularity, thousands of packages **freely** available
- ❑ Where to get the packages
 - ❑ Principal repository: **CRAN** (<https://cran.r-project.org/>)
 - ❑ ~ 10,000 packages organized by views

Ressources

❑ R Packages

- ❑ When download
- ❑ Due to R packa

❑ Where to get the pa

- ❑ Principal repository
- ❑ ~ 10,000 packages

CRAN Task Views	
Bayesian	Bayesian Inference
ChemPhys	Chemometrics and Computational Physics
ClinicalTrials	Clinical Trial Design, Monitoring, and Analysis
Cluster	Cluster Analysis & Finite Mixture Models
DifferentialEquations	Differential Equations
Distributions	Probability Distributions
Econometrics	Econometrics
Environmetrics	Analysis of Ecological and Environmental Data
ExperimentalDesign	Design of Experiments (DoE) & Analysis of Experimental Data
ExtremeValue	Extreme Value Analysis
Finance	Empirical Finance
FunctionalData	Functional Data Analysis
Genetics	Statistical Genetics
Graphics	Graphic Displays & Dynamic Graphics & Graphic Devices & Visualization
HighPerformanceComputing	High-Performance and Parallel Computing with R
MachineLearning	Machine Learning & Statistical Learning
MedicalImaging	Medical Image Analysis
MetaAnalysis	Meta-Analysis
Multivariate	Multivariate Statistics
NaturalLanguageProcessing	Natural Language Processing
NumericalMathematics	Numerical Mathematics
OfficialStatistics	Official Statistics & Survey Methodology
Optimization	Optimization and Mathematical Programming
Pharmacokinetics	Analysis of Pharmacokinetic Data
Phylogenetics	Phylogenetics, Especially Comparative Methods
Psychometrics	Psychometric Models and Methods
ReproducibleResearch	Reproducible Research
Robust	Robust Statistical Methods
SocialSciences	Statistics for the Social Sciences
Spatial	Analysis of Spatial Data
SpatioTemporal	Handling and Analyzing Spatio-Temporal Data
Survival	Survival Analysis
TimeSeries	Time Series Analysis
WebTechnologies	Web Technologies and Services
gR	Graphical Models in R

functions
freely available

[object.org/\)](http://CRAN.R-project.org/)

Ressources

- ❑ R Packages
 - ❑ When downloading R, it comes with “basic” functions
 - ❑ Due to R popularity, thousands of packages **freely** available
- ❑ Where to get the packages?
 - ❑ Principal: **CRAN** (<https://cran.r-project.org/>)
 - ❑ ~ 10,000 packages organized by views
 - ❑ principal function is `install.packages()`

Ressources

- ❑ R Packages
 - ❑ When downloading R, it comes with “basic” functions
 - ❑ Due to R popularity, thousands of packages **freely** available
- ❑ Where to get the packages?
 - ❑ Principal: **CRAN** (<https://cran.r-project.org/>)
 - ❑ Dedicated to bioinformatics: **BioConductor** (<http://bioconductor.org/>)
 - ❑ More than 10 years experience and 1000 packages

Ressources

❑ R Packages

- ❑ Where
- ❑ Due

The screenshot shows the Bioconductor website's "About" page. At the top, there is a navigation bar with links for Home, Install, Help, Developers, and About. A search bar is also present. The main content area has a breadcrumb trail "Home » About". It describes Bioconductor as an open source project for genomic data analysis using R. It mentions the release version (twice yearly), development version (prior to incorporation), and meta-data packages. Citations for the project are provided, along with sections on Bioconductor Packages and Project Goals.

Bioconductor
OPEN SOURCE SOFTWARE FOR BIOINFORMATICS

Home Install Help Developers About

Search:

Home » About

Bioconductor is an open source, open development software project to provide tools for the analysis and comprehension of high-throughput genomic data. It is based primarily on the [R](#) programming language.

The Bioconductor [release version](#) is updated twice each year, and is appropriate for most users. There is also a [development version](#), to which new features and packages are added prior to incorporation in the release. A large number of [meta-data packages](#) provide pathway, organism, microarray and other annotations.

The Bioconductor project started in 2001 and is overseen by a [core team](#), based primarily at [Roswell Park Cancer Institute](#), and by other members coming from US and international institutions.

Key citations to the project include Huber et al., 2015 [Nature Methods 12:115-121](#) and Gentleman et al., 2004 [Genome Biology 5:R80](#).

Bioconductor Packages

Most Bioconductor components are distributed as [R packages](#). The functional scope of [Bioconductor packages](#) includes the analysis of DNA microarray, sequence, flow, SNP, and other data.

Project Goals

The broad goals of the Bioconductor project are:

- To provide widespread access to a broad range of powerful statistical and graphical methods for the analysis of genomic data.
- To facilitate the inclusion of biological metadata in the analysis of genomic data, e.g. literature data from PubMed, annotation data from Entrez genes.
- To provide a common software platform that enables the rapid development and deployment of extensible, scalable, and interoperable software.
- To further scientific understanding by producing high-quality [documentation](#) and reproducible research.
- To [train](#) researchers on computational and statistical methods for the analysis of genomic data.

ctor.org/)

Ressources

- ❑ R Packages
 - ❑ When downloading R, it comes with “basic” functions
 - ❑ Due to R popularity, thousands of packages **freely** available
- ❑ Where to get the packages?
 - ❑ Principal: **CRAN** (<https://cran.r-project.org/>)
 - ❑ Dedicated to bioinformatics: **BioConductor** (<http://bioconductor.org/>)
 - ❑ More than 10 years experience and 1000 packages
 - ❑ Need to source the biocLite() function from BioConductor

```
source("http://bioconductor.org/biocLite.R")
biocLite("package.name")
```

Ressources

- ❑ R Packages
 - ❑ When downloading R, it comes with “basic” functions
 - ❑ Due to R popularity, thousands of packages **freely** available
- ❑ Where to get the packages?
 - ❑ Principal: **CRAN** (<https://cran.r-project.org/>)
 - ❑ Dedicated to bioinformatics: **BioConductor** (<http://bioconductor.org/>)
 - ❑ More than 10 years experience and 1000 packages
 - ❑ Need to source the biocLite() function from BioConductor
 - ❑ Forum dedicated to bioinformatics and BioConductor packages (<https://support.bioconductor.org/t/>) - No one can know all packages.

Ressources

sign up / log in • about • faq • rss 

 Bioconductor
OPEN SOURCE SOFTWARE FOR BIOINFORMATICS

ASK QUESTION LATEST NEWS JOBS TUTORIALS TAGS USERS

Search Go!

limma × 3280	microarray × 2132	go × 1859	affy × 1741	annotation × 1711	deseq2 × 1566
cancer × 1492	normalization × 1447	probe × 1376	edger × 1214	bioconductor × 927	cdf × 918
biomart × 755	rnaseq × 745	process × 655	R × 572	convert × 456	deseq × 448
software error × 420	gcrma × 398	oligo × 388	graph × 354	clustering × 347	annotate × 338
snp × 323	genetics × 317	bsgenome × 290	diffbind × 290	sequencing × 287	genomicranges × 268
dexseq × 265	biobase × 256	gviz × 241	lumi × 237	gostats × 237	mirna × 234
ranges × 231	job × 226	pathways × 223	organism × 219	rgraphviz × 211	genomicfeatures × 205
differential gene	biostings × 202	genefilter × 197	rtracklayer × 191	minfi × 181	alignment × 178
expression × 203	geoquery × 177	news × 174	beadarray × 172	preprocessing × 171	rna-seq × 168
variantannotation × 167	vsn × 166	category × 159	network × 157	hgu133plus2 × 156	granges × 152
xps × 151	topgo × 149	coverage × 146	sva × 146	annbuilder × 146	chippeakanno × 141
flowcore × 139	multtest × 139	marray × 138	proteomics × 135	visualization × 134	simpleaffy × 131
annotationdbi × 128	regression × 124	assign × 122	wgcna × 122	goseq × 121	gui × 117
rsamtools × 116	hgu133a × 114	htqpcr × 113	transcription × 112	yeast × 111	qpcr × 111
survival × 107	repostools × 106	chipseq × 103	microrna × 103	affyio × 102	pathview × 96
arrayqualitymetrics × 96	genomes × 95	rsbread × 94	affyplm × 93	error × 93	multiple factor design × 91
limmagui × 91	affylmgui × 90	biocinstaller × 90	hgu95av2 × 89	ggbio × 88	

<prev • 3,890 results • page 1 of 39 • next >

(<https://support.bioconductor.org/t/>) - No one can know all packages.



Which type of variable?

No dimension

- ❑ Numeric (float)

```
x <- c(3,2,0,5);y <- "85"  
is.numeric(x); as.numeric(y)
```

- ❑ Integer

```
is.integer(x); as.integer(x)
```

- ❑ Character / String

```
as.character(x); nchar("Boo")  
paste("Boo", "Boo")
```

- ❑ Logical (T/F)

```
3 %in% x; !(x < 3); TRUE==T  
which(x < 3); T+TRUE+F+FALSE
```

Undefined values

- ❑ Non existing number

```
## NaN, Inf  
x / 0
```

- ❑ Null data (NULL)

```
z <- NULL; length(z); is.null(z)  
z + 2
```

- ❑ Missing data (NA)

```
x <- c(3,2,0,5,NA); class(x)  
is.na(x); na.omit(x)  
mean(x); mean(x,na.rm=T)
```

Combine elements: Constraint on type

❑ Vector (1D)

```
x <- c(3,2,0,5)
y <- c("85",8,9)
x[2]
seq(1,6,1)
rep(y,3); 1:6; x[2:3]
x^2; x + x[2:3]
```

❑ matrix (2D) / Arrays (ND)

```
# 2 dimensions
x2 <- matrix(1:12,nrow=4,ncol=3,byrow=TRUE)
x2[3,2]; x2[,2]
x2 + x2
x2 + c(3,2,5)
x2[,1] <- c(3,2,5)
dim(x2); row.names(x2)
# 3 dimensions
x3 <- array(1:12,dim=c(2,2,3))
x3[1,2,2] ; x3[1,2,]
```

❑ Factor: categorical vector(1D)

```
# limited number of different values, encoded
in integer, used by read.table()
data <- factor(c(3,2,0,5,3,2,0,5,3,2,0,5))
as.integer(data)
data==5; data=="5"
sum(data)
levels(data) <- c("bleu", "jaune", "rouge", "gris")
```

❑ In a general manner, to access the data:

```
x[indexes.dim1]
x2[indexes.dim1,indexes.dim2]
x3[indexes.dim1,indexes.dim2,indexes.dim3]
xN[indexes.dim1,indexes.dim2,indexes.dim3,...,indexes.NDim]
```

Combine elements: Not constraint on type

❑ data frame (2D)

```
# collection of vectors and/or factors  
# constrained by column  
firstNames <- c("Remy", "Lol", "Pierre", "Domi", "Ben")  
IMC <- data.frame(sex=c("H", "F", "H", "F", "H") ,  
                   height=c(1.83,1.76,1.82,1.60,1.90),  
                   weight=c(67,58,66,48,75),  
                   row.names=firstNames)
```

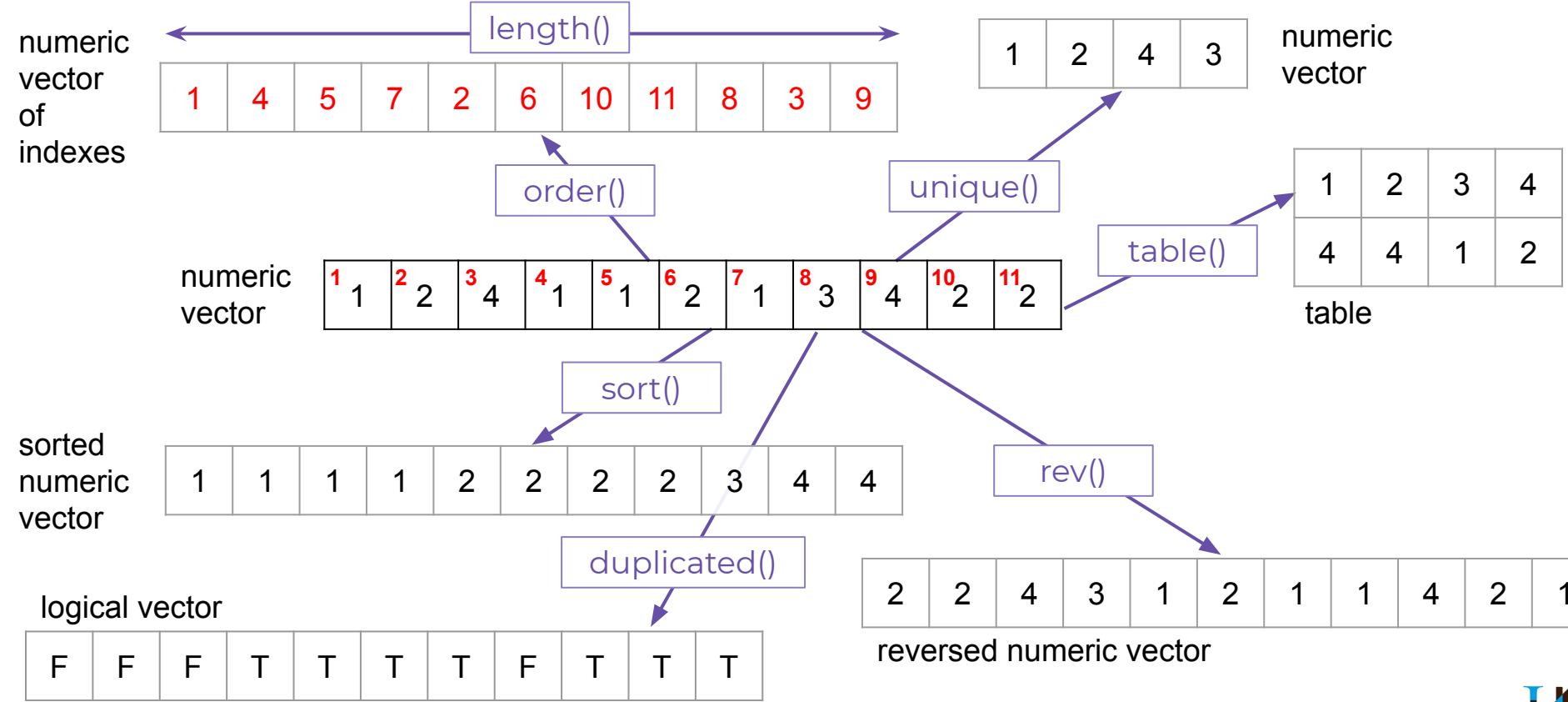
```
IMC$height  
IMC[, "sex"]  
IMC["Remy",]  
IMC[3,2]  
head(IMC)  
colnames(IMC)
```

❑ list

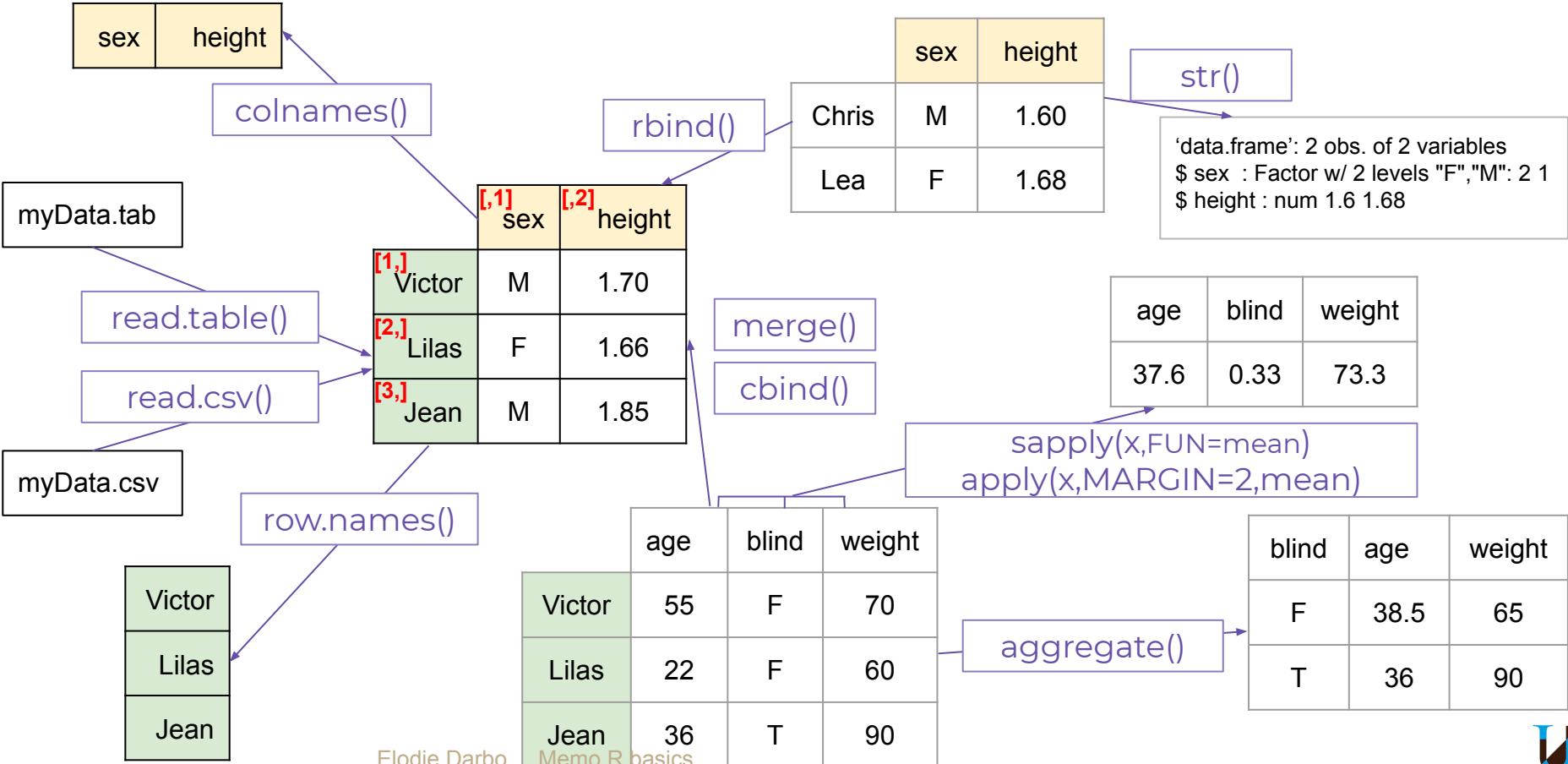
```
# very flexible, store everything  
list.ex <- list(one_vec=1:12,  
                 one_name="Boo",  
                 one_tab=matrix(1:4,nrow=2))
```

```
list.ex$one_tab  
list.ex[[1]]  
list.ex$new <- list(a="a",b="b")  
list.ex$new$a
```

Manipulate vectors



Manipulate data frames



How would you?

```
load("TP_RNAseq/R_course_objects.RData")
# We will use the data.frame objects imc and imc_age
```

- Assemble the 2 data frames?
 - What do they have in common?
- Get the mean weight per gender?
- Get the mean values for all numeric information?

How would you? ... Answer

```
load("TP_RNAseq/R_course_objects.RData")
# We will use the data.frame objects imc and imc_age
```

- Assemble the 2 data frames?
 - What do they have in common?
- Get the mean weight per gender?
- Get the mean values for all numeric information?

```
# merge the data frames
imc_merged=merge(imc, imc_age, by=c( "prenom", "sexe"))
# Aggregate
aggregate(poids~sexe,imc_merged,mean)
# get numeric columns
numCol <- sapply(imc_merged,is.numeric) # returns logical
imc_num <- imc_merged[,numCol]
# apply on columns the function mean
apply(imc_num,2,mean)
sapply(imc_merged[,numCol],mean)
```

R Objects for NGS



Why using R for NGS analysis?



To replace existing command-line tools?

- R NGS treatment built on existing tools (BWA, BowTie, Samtools ...) and data bases (EnsEMBL ...)



To have a analysis workflow?

- No magic ready-to-use solution, constant evolving tools, constant new, more performant methods. Package documentation **must** be consulted !



To take advantage of statistical methods?



To manipulate large data (consulting, filtering, modifying)?

- Import/export of NGS format files (BAM, vcf etc) in specific R objects



To visualise graphically data and results?

NGS oriented packages & functions

nucleic/protein sequences (FASTA)

GC content, pattern matching

Biostring: e.g. `getSeq()`, `vcoutPattern()`

reads (FASTQ)

Quality control, cleaning

ShortRead: e.g. `FastqSampler()`,
`readFastq()`, `qa()`, `report()`, `sread()`

alignment (BAM/SAM)

Manipulate, storage

GenomicAlignment: e.g. `filterBam()`,
`readAlignment()`, `scanBam()`

SNPs (VCF)

Predict protein impact, prioritize mutations (drivers)

VariantAnnotation e.g. `readVcf()`, `rowRanges()`,
`predictCoding()`

regions (GFF/BED)

Genomic interval overlap, storage

GenomicRanges e.g. `findOverlaps()`,
`intersect()`, `reduce()`

metadata (GFF/BED/WIG/ucsc tracks)

Annotations with informations on the
biological objects (gene, genomic
region)

BiomaRt e.g. `getBM()`

rtracklayer e.g. `import()`, `export()`

GenomicFeatures e.g. `makeTxDB()`

all-in-one (from FASTQ to specific results)

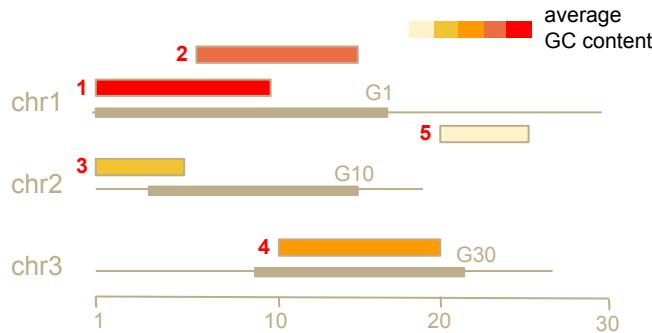
Build a workflow with no problem of
compatibility

QuasR e.g. `qa()`, `qAlign()`, `qCount()`

Rsubread e.g. `buildIndex()`, `align()`,
`featureCounts()`, `extractSNP()`

Genomic regions: extract info from GRanges

GRanges(seqnames, range=IRanges(start,end), strand, gene, GC)

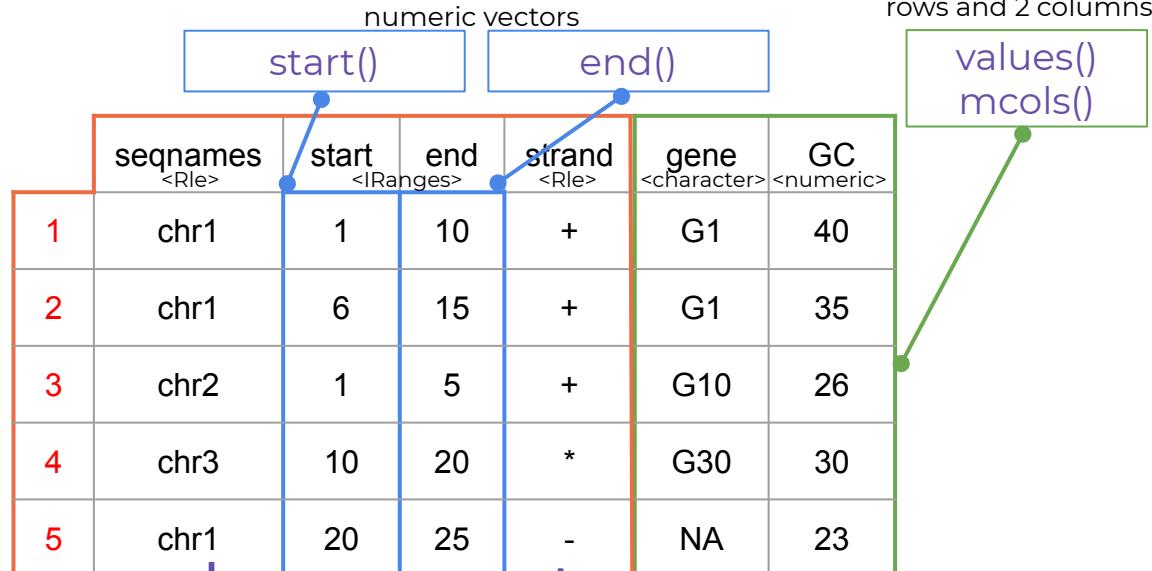


"chr1"	"chr1"	"chr2"	"chr3"	"chr1"
--------	--------	--------	--------	--------

as.vector()

character-Rle of length 5 with 4 runs

Lengths	2	1	1	1
Values	"chr1"	"chr2"	"chr3"	"chr1"



DataFrame with 10 rows and 2 columns

values()
mcols()

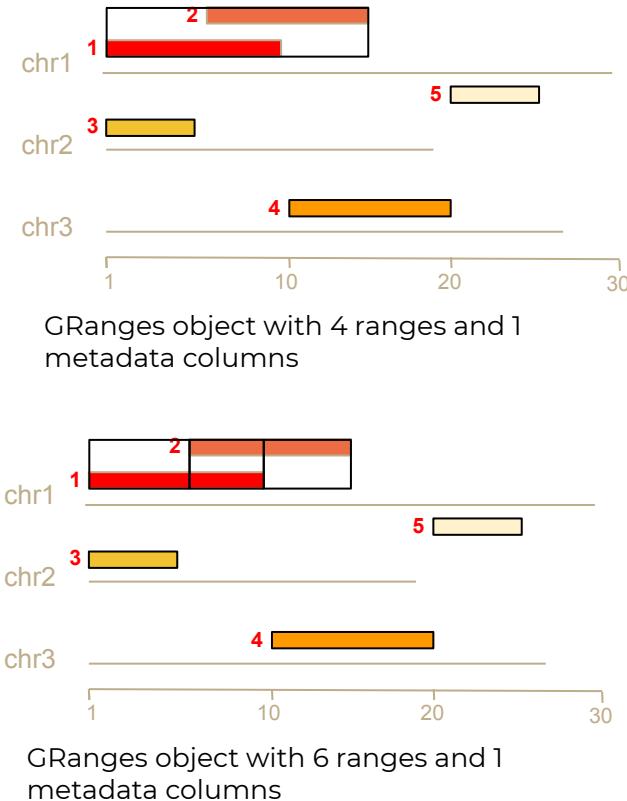
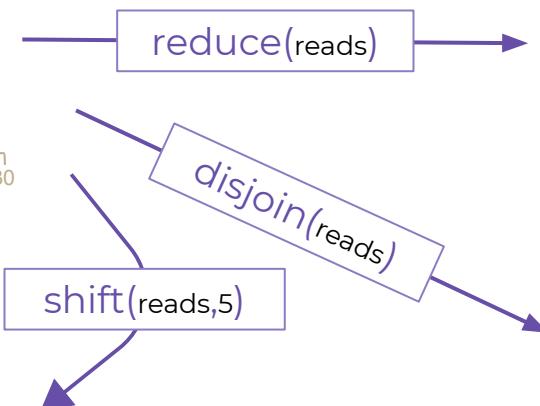
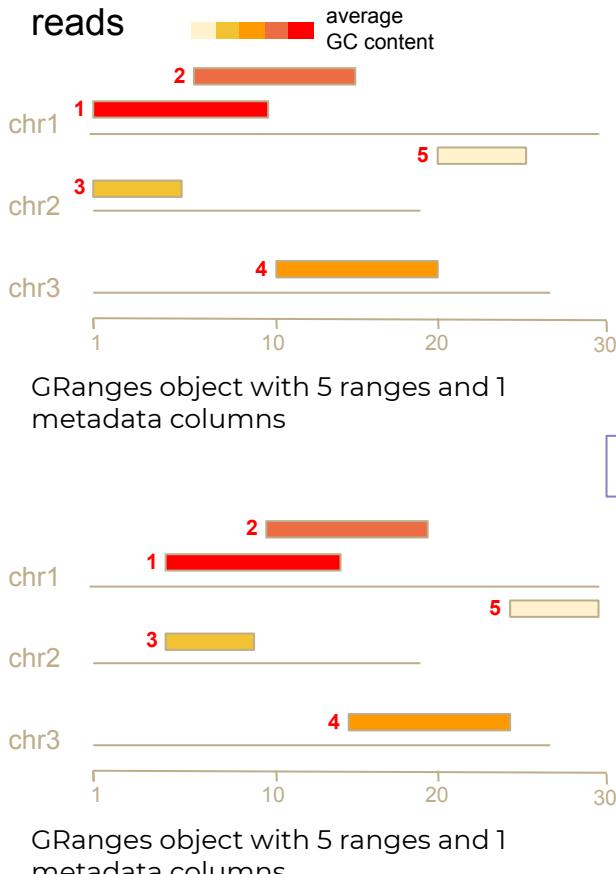
seqnames() strand()

range()

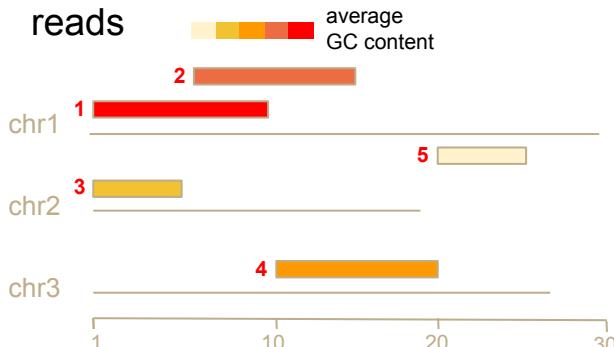
GRanges object with 7 ranges
and 0 metadata columns

Lengths	3	1	1
Values	+"	*	-

Genomic regions: manipulate GRanges



Genomic regions: manipulate *GRanges*

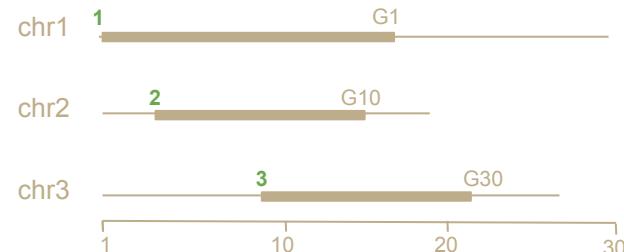


GRanges object with 5 ranges and 1 metadata columns

	queryHits <integer>	subjectHits <integer>
1	1	1
2	2	1
3	3	2
4	4	3

queryLength: 5 / subjectLength: 3

genomic_features

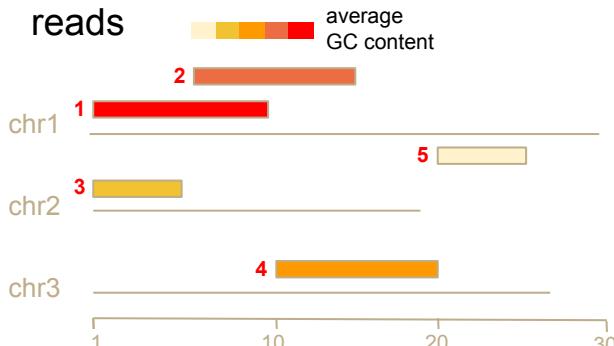


GRanges object with 3 ranges and 1 metadata columns

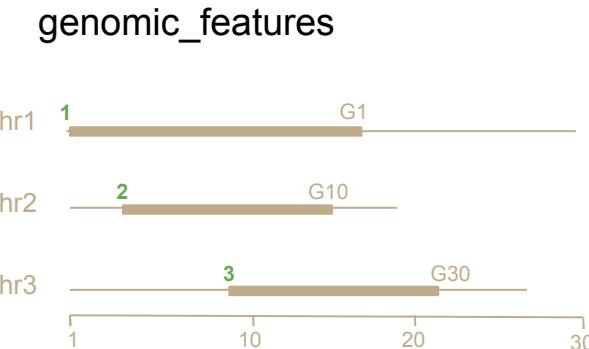
`findOverlaps(reads,genomic_features)`

Hits object with 4 hits and 0 metadata columns

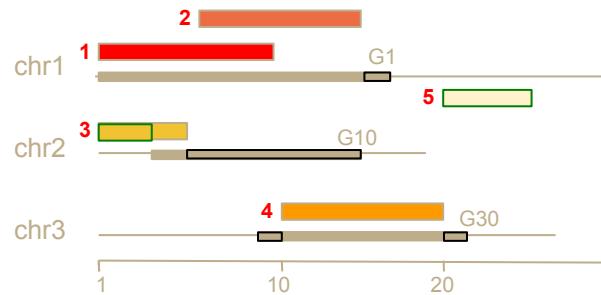
Genomic regions: manipulate GRanges



GRanges object with 5 ranges and 1 metadata columns



GRanges object with 3 ranges and 1 metadata columns



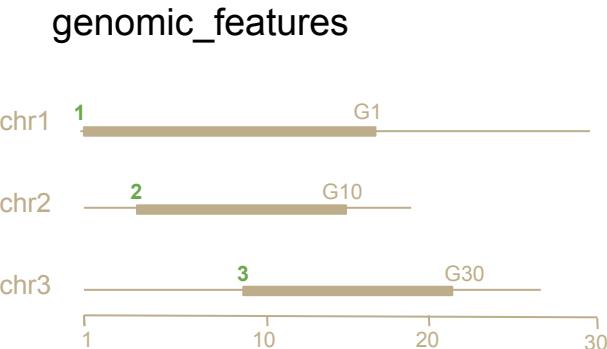
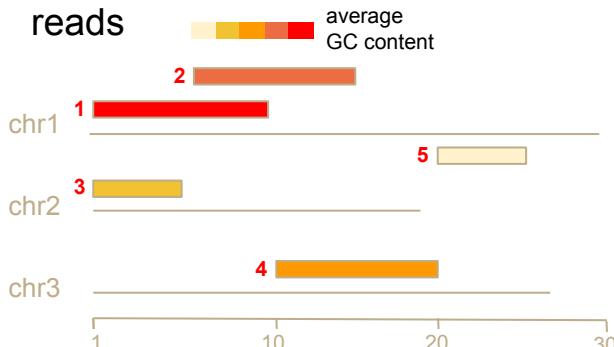
`setdiff(genomic_features, reads)`

GRanges object with 4 ranges and 0 metadata columns

`setdiff(reads, genomic_features)`

GRanges object with 2 ranges and 0 metadata columns

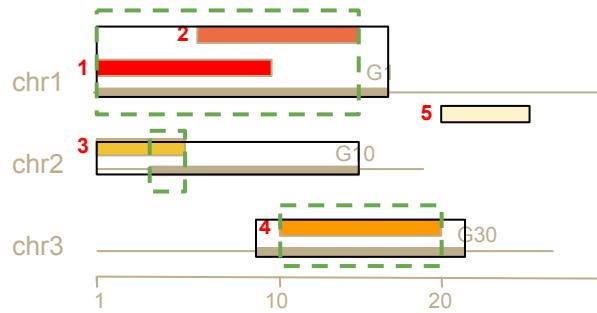
Genomic regions: manipulate GRanges



GRanges object with 5 ranges and 1 metadata columns

`intersect(reads,genomic_features)`

GRanges object with 3 ranges and 0 metadata columns



`union(reads,genomic_features)`

GRanges object with 4 ranges and 0 metadata columns

Annotation Objects

By species and genome version

BSGenome

TxDB

OrgDB

Give access to
genomic sequences

Give access to gene,
transcript genomic
positions

Give access to gene
functional annotations

GenomicFeatures: build TxDB objects from scratch

`makeTxDb(transcripts, splicings, genes)`

data.frame with 6 columns and a row per exon

tx_id	exon_rank	exon_start	exon_end	cds_start	cds_end
1	1	1	999	1	999
2	1	2001	2085	2022	2085
2	2	2101	2144	2101	2144
2	3	2131	2199	2131	2193
3	1	2001	2085	NA	NA
3	2	2131	2199	NA	NA

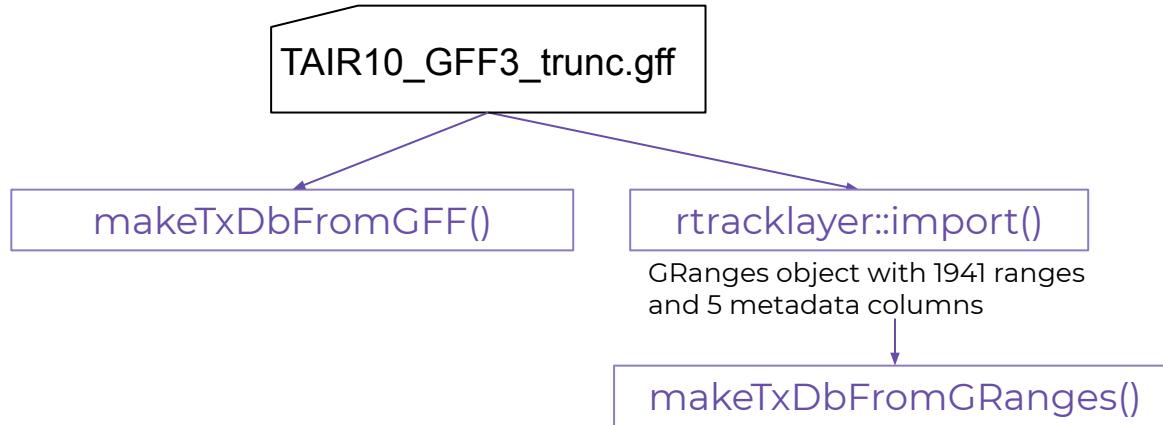
data.frame with 5 columns and a row per transcript

tx_id	tx_chrom	tx_strand	tx_start	tx_end
1	chr1	-	1	999
2	chr1	+	2001	2199
3	chr1	+	2001	2199

data.frame with 2 columns and a row per gene

tx_id	gene_id
1	gene1
2	gene2
3	gene3

GenomicFeatures: build TxDB objects from known annotations



[\(http://www.biomaRt.org/\)](http://www.biomaRt.org/)
[\(http://www.ensembl.org/biomart/martview/\)](http://www.ensembl.org/biomart/martview/)

`makeTxDbFromBiomart()`

Needs internet connection

WARNING



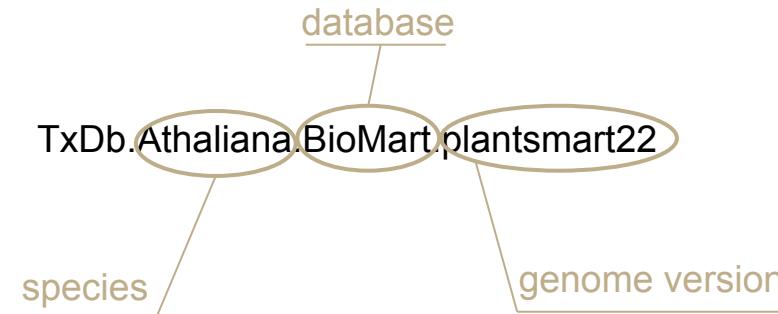
`makeTxDbFromUCSC()`

GenomicFeatures: Ready-to-use TxDB objects

There is a collection of packages containing TxDB objects in BioConductor. You can find a list in the BioConductor GenomicFeatures page.

<https://bioconductor.org/packages/release/bioc/html/GenomicFeatures.html>

```
TxDb.Athaliana.BioMart.plantsmart25, TxDb.Athaliana.BioMart.plantsmart28,  
TxDb.Btaurus.UCSC.bosTau8.refGene, TxDb.Celegans.UCSC.ce11.refGene,  
TxDb.Celegans.UCSC.ce6.ensGene, TxDb.Cfamiliaris.UCSC.canFam3.refGene,  
TxDb.Dmelanogaster.UCSC.dm3.ensGene,  
TxDb.Dmelanogaster.UCSC.dm6.ensGene, TxDb.Dreroi.UCSC.danRer10.refGene,  
TxDb.Ggallus.UCSC.galGal4.refGene, TxDb.Ggallus.UCSC.galGal5.refGene,  
TxDb.Hsapiens.BioMart.igis, TxDb.Hsapiens.UCSC.hg18.knownGene,  
TxDb.Hsapiens.UCSC.hg19.knownGene,  
TxDb.Hsapiens.UCSC.hg19.lincRNAsTranscripts,  
TxDb.Hsapiens.UCSC.hg38.knownGene, TxDb.Mmulatta.UCSC.rheMac3.refGene,  
TxDb.Mmulatta.UCSC.rheMac8.refGene, TxDb.Mmusculus.UCSC.mm10.ensGene,  
TxDb.Mmusculus.UCSC.mm10.knownGene,  
TxDb.Mmusculus.UCSC.mm9.knownGene,  
TxDb.Ptroglodytes.UCSC.panTro4.refGene, TxDb.Rnorvegicus.BioMart.igis,  
TxDb.Rnorvegicus.UCSC.rn4.ensGene, TxDb.Rnorvegicus.UCSC.rn5.refGene,  
TxDb.Rnorvegicus.UCSC.rn6.refGene, TxDb.Scerevisiae.UCSC.sacCer2.sgdGene,  
TxDb.Scerevisiae.UCSC.sacCer3.sgdGene, TxDb.Sscrofa.UCSC.susScr3.refGene
```

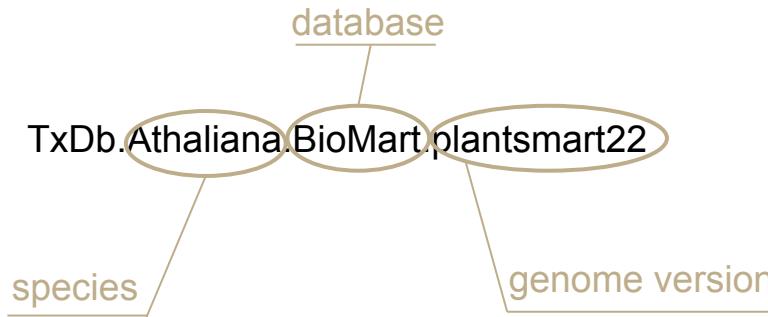


GenomicFeatures: Ready-to-use TxDB objects

There is a collection of packages containing TxDB objects in BioConductor. You can find a list in the BioConductor GenomicFeatures page.

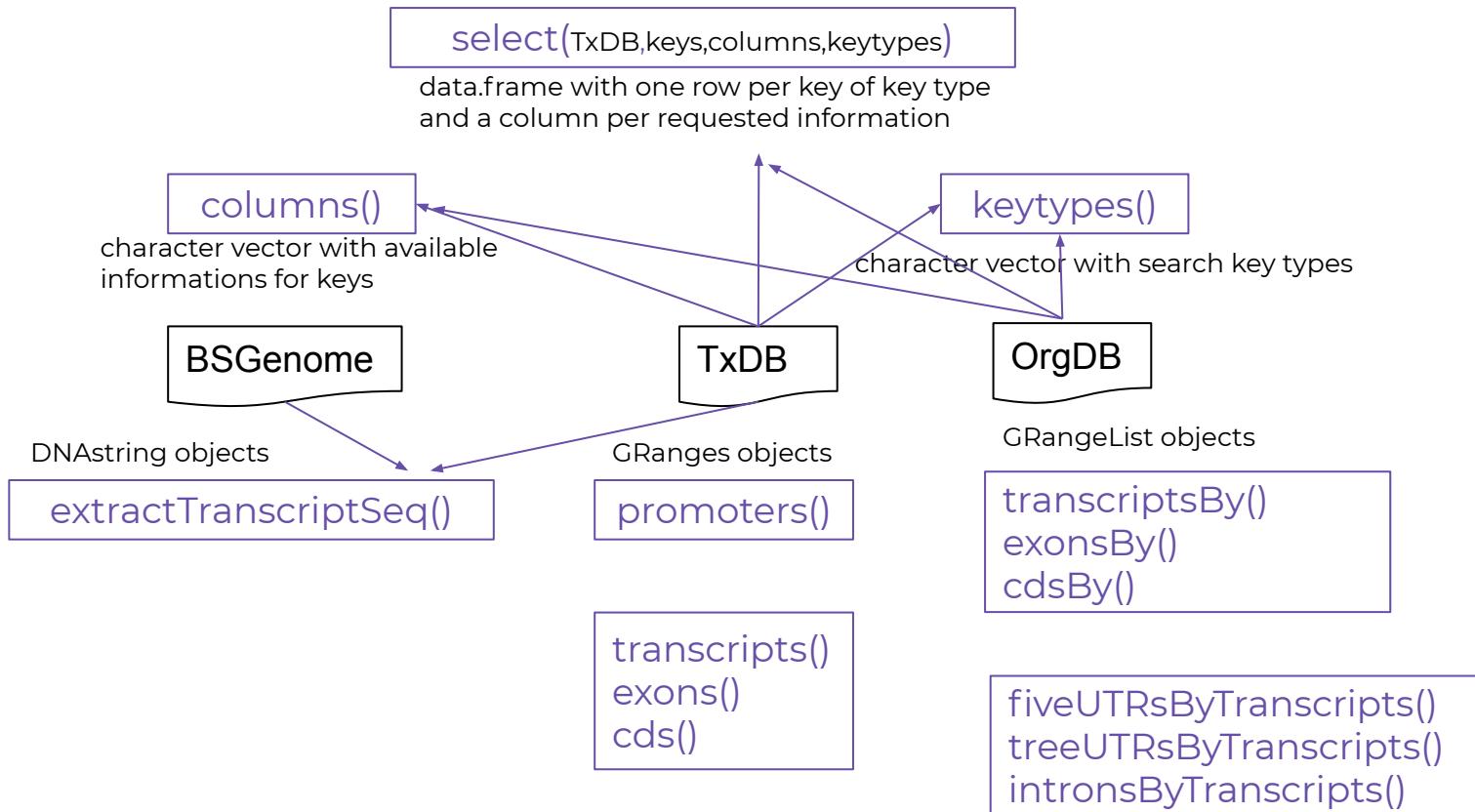
<https://bioconductor.org/packages/release/bioc/html/GenomicFeatures.html>

```
TxDb.Athaliana.BioMart.plantsmart25, TxDb.Athaliana.BioMart.plantsmart28,  
TxDb.Btaurus.UCSC.bosTau8.refGene, TxDb.Celegans.UCSC.ce11.refGene,  
TxDb.Celegans.UCSC.ce6.ensGene, TxDb.Cfamiliaris.UCSC.canFam3.refGene,  
TxDb.Dmelanogaster.UCSC.dm3.ensGene,  
TxDb.Dmelanogaster.UCSC.dm6.ensGene, TxDb.Dreroi.UCSC.danRer10.refGene,  
TxDb.Ggallus.UCSC.galGal4.refGene, TxDb.Ggallus.UCSC.galGal5.refGene,  
TxDb.Hsapiens.BioMart.igis, TxDb.Hsapiens.UCSC.hg18.knownGene,  
TxDb.Hsapiens.UCSC.hg19.knownGene,  
TxDb.Hsapiens.UCSC.hg19.lincRNAsTranscripts,  
TxDb.Hsapiens.UCSC.hg38.knownGene, TxDb.Mmulatta.UCSC.rheMac3.refGene,  
TxDb.Mmulatta.UCSC.rheMac8.refGene, TxDb.Mmusculus.UCSC.mm10.ensGene,  
TxDb.Mmusculus.UCSC.mm10.knownGene,  
TxDb.Mmusculus.UCSC.mm9.knownGene,  
TxDb.Ptroglobutes.UCSC.panTro4.refGene, TxDb.Rnorvegicus.BioMart.igis,  
TxDb.Rnorvegicus.UCSC.rn4.ensGene, TxDb.Rnorvegicus.UCSC.rn5.refGene,  
TxDb.Rnorvegicus.UCSC.rn6.refGene, TxDb.Scerevisiae.UCSC.sacCer2.sgdGene,  
TxDb.Scerevisiae.UCSC.sacCer3.sgdGene, TxDb.Sscrofa.UCSC.susScr3.refGene
```



```
source("https://bioconductor.org/biocLite.R")  
biocLite("TxDb.Athaliana.BioMart.plantsmart22")  
library(TxDb.Athaliana.BioMart.plantsmart22)
```

GenomicFeatures: Extract informations from TxDB objects



How would you?

```
library(org.Hs.eg.db)  
library(TxDb.Hsapiens.UCSC.hg19.knownGene)
```

- Get CDS from BRCA1 (key type is SYMBOL) in human genome hg19?
 - Does it exist in both OrgDB and TxDB objects?
 - Knowing ENTREZID from OrgDB correspond to GENEID in TxDB, how to get CDS list by transcripts?

How would you? ... Answers

```
library(org.Hs.eg.db)
library(TxDb.Hsapiens.UCSC.hg19.knownGene)
```

- Get CDS from BRCA1 (key type is SYMBOL) in human genome hg19?
 - Does it exist in both OrgDB and TxDB objects?
 - Knowing ENTREZID from OrgDB correspond to GENEID in TxDB, how to get CDS list by transcripts?

```
# The SYMBOLs are not present in the TxDB, from OrgDB find the corresponding ENTREZID. The key is the gene symbol, the required information in the ENTREZID.
```

```
eid=select(org.Hs.eg.db, "BRCA1", "ENTREZID", "SYMBOL")[[["ENTREZID"]]]
```

```
# Now you can get the transcript names (TXNAME) from the GENEID in TxDB
```

```
txid <- select(x=TxDb.Hsapiens.UCSC.hg19.knownGene, eid, "TXNAME", "GENEID")$TXNAME
```

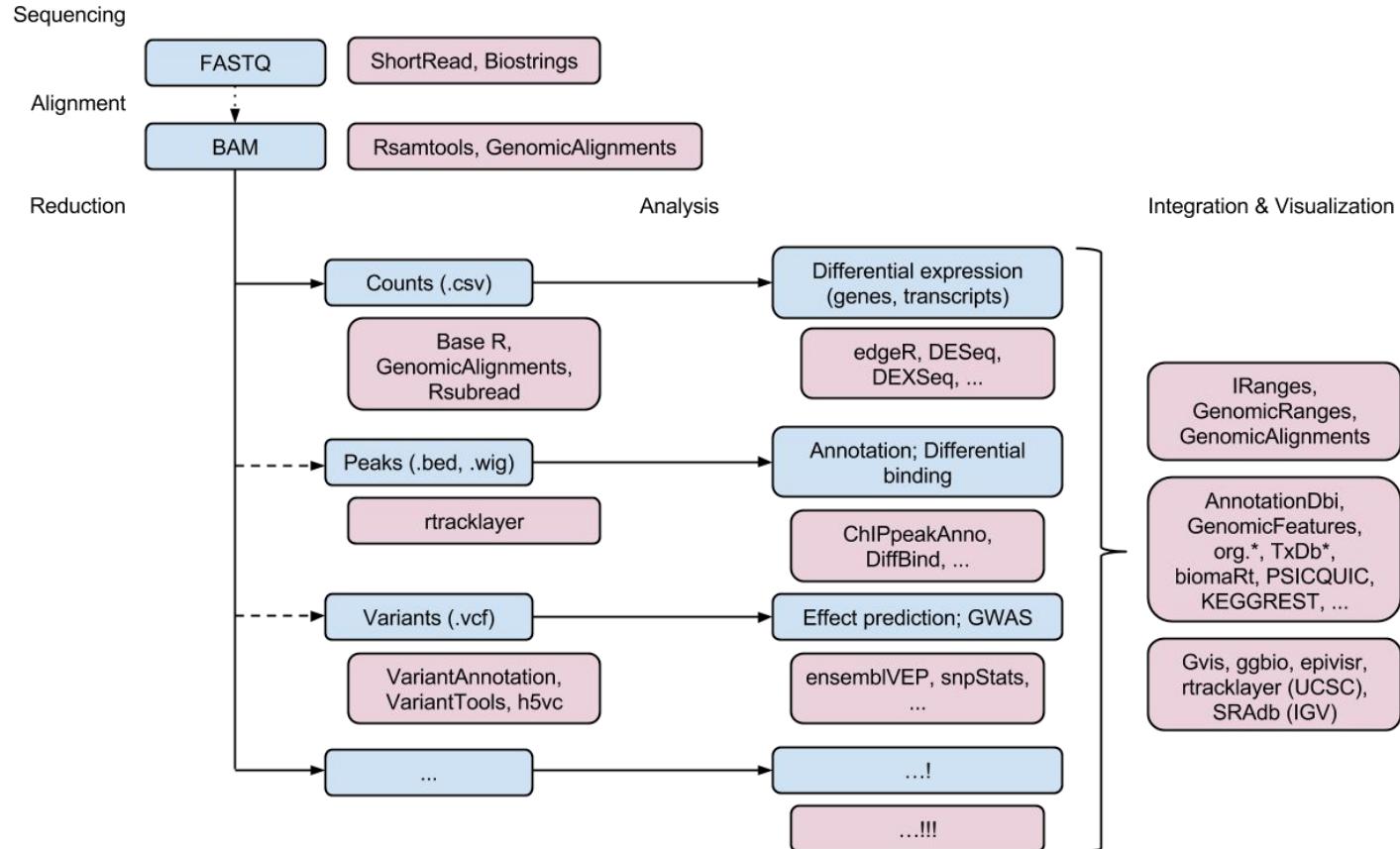
```
# Extraction of CDS by transcripts specifying its transcript names
```

```
cds <- cdsBy(TxDb.Hsapiens.UCSC.hg19.knownGene, by="tx", use.names=TRUE)
```

```
# Select in the cds object (GRanges) the cds relative to the BRCA1 transcripts
```

```
brca1cds <- cds[names(cds) %in% txid]
```

Specific tools



R plots



Questions about data analysis

Exploratory

Behaviour of the data

Not to be generalized / Not predictive
Unsupervised and supervised

*discover new connections
determine further analyses*

Inferential

Generalise to population from
representative subset
Statistics and confidence

survey, poll

Descriptive

Summary of the data

*data type, sample number, MAF,
mean gene expression*

Predictive

Use measures to predict other
measure values
Not necessarily causal

*Amazon recommendations
Transcriptional signature*

Causal

Understand impact of a
variable on another
Need of randomized and
prospective experiments

*Clinical test
phenotypic consequence of
environmental factors*

Questions about data analysis

Exploratory

Behaviour of the data

Not to be generalized / Not predictive
Unsupervised and supervised

*discover new connections
determine further analyses*

Inferential

Generalise to population from
representative subset
Statistics and confidence

survey, poll

Descriptive

Summary of the data

*data type, sample number, MAF,
mean gene expression*

Predictive

Use measures to predict other
measure values
Not necessarily causal

*Amazon recommendations
Transcriptional signature*

Causal

Understand impact of a
variable on another
Need of randomized and
prospective experiments

*Clinical test
phenotypic consequence of
environmental factors*

Why R plots?



Customized to infinity and beyond



Generate high resolution graphs (publication, printing ...)



Completely reproducible and automatizable



In base R functions & dedicated packages (ggplot2)

Some useful links

CRAN Task View: Graphic Displays & Dynamic Graphics & Graphic Devices & Visualization

Maintainer: Nicholas Lewin-Koh
Contact: nikko at halmail.net
Version: 2015-01-07
URL: <https://CRAN.R-project.org/view=Graphics>

R is rich with facilities for creating and developing interesting graphics. Base R contains functionality for many plot types including : as device drivers for all platforms running R, [lattice](#) and grid are supplied with R's recommended packages and are included in every graphics environment than the base R graphics.

R's base graphics are implemented in the same way as in the S3 system developed by Becker, Chambers, and Wilks. There is a static global parameters such as margins and layouts which can be manipulated by the user using `par()` commands. The R graphics engine redrawing a whole plot. This situation may change in R 2.7.x, where developers are working on double buffering for R devices. Even

One can quickly run into trouble with R's base graphic system if one wants to design complex layouts where scaling is maintained or limitations and as a result packages like [lattice](#), [ggplot2](#), [grid](#) or [hexbin](#) use grid for the underlying primitives. When using plots grid commands, e.g., `grid.polygon()` rather than `polygon()`. Also grid maintains a stack of viewports from the device and one needs vignettes.

The graphics packages in R can be organized roughly into the following topics, which range from the more user oriented at the top to

- Plotting :** Enhancements for specialized plots can be found in [plotrix](#), for polar plotting, [vcd](#) for categorical data, [hexbin](#) for hexagonal binning, which also has an implementation of Tukey's bag plot. For 3D plots [lattice](#), [scatterplot3d](#) and [mice](#) or [rgl](#). The package [onion](#) for visualizing quantiles and octonions is well suited to display 3D graphics based on derived metrics.
- Graphical Applications :** This is not much different from the plotting section except that these packages have tools that may subject areas, like medical imaging, the relevant task view contributed by other dedicated userR's is an excellent place to start.
- Effect ordering :** The [grid](#) package focuses on the ordering of graphs to accentuate cluster structure or natural ordering t criteria. For ordering an array of displays, [grid.layout](#) can be useful.
- Large Data Sets :** Large data sets pose very different challenges from moderate and small datasets. Aside from very quickly, and [hexbin](#) can bin bivariate data onto a hexagonal lattice, the advantage being that the irregular lines and t with [lattice](#). An alternative is to use [seaglass](#) to produce a scatterplot matrix of "data about the data", and look for interesting patterns.
- Trees and Graphs :** [ape](#) and [ade4](#) have functions for plotting phylogenetic trees, which can be used for plotting dendrogram placement, so may be useful for very large trees. [igraph](#) has the Tifford-Rheingold algorithm implemented and is use [igraph](#) have functions for plotting and layout, especially useful for representing large networks.
- Graphics Systems :** [lattice](#) is built on top of the grid graphics system and is an R implementation of William Cleveland's trellis implementation of the system described in "A Grammar of Graphics" by Leland Wilkinson. Like [lattice](#), [ggplot2](#) (also built on more emphasis on reshaping data, transformation, and assembling the elements of a plot...).
- Devices :** Whereas grid is built on top of the R graphics engine, many in the R community have found the R graphics engine sc basic supplies devices for PostScript, PDF, JPEG and other formats. Devices on CRAN include [cairoDevice](#) which is a device t the Gimp Tool Kit, similar to pyGTK2. [RSvgDevice](#) is an SVG device driver and interfaces well with vector drawing program standard. Trust Microsoft, [rgl](#) provides a device driver based on OpenGL, and is good for 3D and interactive development. Las
- Colors :** The package [colorspace](#) provides a set of functions for transforming between color spaces and [mixcolor\(\)](#) for mixing suitable for coding categorical variables (`rainbow`, `heat.colors`) and numerical information (`sequential_hcl()`, `diverge_hcl()`).
- Interactive Graphics :** There are several efforts to implement interactive graphics systems that interface well with R. In the case of edge as well as link with other views of the data, [rgobi](#) embeds the GGobi interactive graphics system within R, so that t GGobi's edge set functionality. The RoSuDA repository maintained and developed by the University of Augsburg group has two graphics tools contain functions for alpha blending, which produces darker shading around areas with more data. This is except versions of R graphics using the [cairoDevice](#) and [RGtk2](#). Lastly, the [rgl](#) package has mechanisms for interactive manipulation.
- Development :** For development of specialized graphics packages in R, grid should probably be the first consideration for any Java device in the RoSuDA packages, though Java has its own drawbacks. For porting plotting code to grid, using the pack

CRAN packages:

- [ade4](#)
- [animation](#)
- [ape](#)
- [grid](#)

<https://cran.r-project.org/web/views/Graphics.html>

R graph gallery

The blog is a collection of script examples with example data and output plots. R produce excellent quality graphs for research, science and business presentation, publications and other purposes. Self-help codes and examples are provided. Enjoy nice graphs !!

The screenshot shows the homepage of the R graph gallery. At the top, there is a large logo with the letters 'R' in a stylized font. Below the logo, the text 'Graph Gallery: A collection' is displayed. The main content area features several small thumbnail images of R plots, each with a caption below it. To the left of the thumbnails, there is a column of text containing R code snippets. The code snippets are related to various plotting functions in R, such as 'barplot', 'hexbin', 'heatmap', 'histogram', 'kerneldensity', 'lattice', 'lxy', 'points', 'stackedbar', 'treemap', and 'xypoints'. The R code is presented in a monospaced font. The overall layout is clean and organized, making it easy to browse through different types of R plots and their corresponding source code.

<http://rgraphgallery.blogspot.fr/>

Fiche TD avec le logiciel : tdr75

Les paramètres graphiques

J.R. Lobry, A.B. Dufour & D. Chessel

Table des matières

1	Introduction	
1.1	Le jeu de données pour les exercices	...
1.2	Les paramètres graphiques étudiés	...
1.3	Les paramètres graphiques non modifiables	...
1.3.1	par("cex")	...
1.3.2	par("cex") et par("cex")	...
1.3.3	par("cex") et par("cex")	...
1.3.4	par("page")	...
2	Les paramètres graphiques modifiables	
2.1	Les paramètres graphiques logiques	...
2.1.1	par("ann")	...
2.1.2	par("ask")	...
2.1.3	par("bg")	...
2.1.4	par("cex") et par("cex")	...
2.1.5	par("cex")	...
2.2	Les paramètres graphiques entiers	...
2.2.1	par("font")	...
2.2.2	par("las")	...
2.2.3	par("lwd")	...
2.2.4	par("narrow") et par("arcol")	...
2.2.6	par("lab")	...
2.2.7	par("lty")	...
2.2.8	par("cex")	...
2.3	Les paramètres graphiques de type chaîne de caractères	...
2.3.1	par("bg")	...
2.3.2	par("bsy")	...
2.3.3	par("col")	...
2.3.5	par("fg")	...
2.3.6	par("lead")	...
2.3.7	par("lend")	...
2.3.8	par("lty")	...
2.3.9	par("pty")	...
2.3.11	par("xaxt") et par("yaxt")	...
2.4	Les paramètres graphiques numériques	...
2.4.2	par("adj")	...
2.4.3	par("cex") et par("cex")	...
2.4.4	par("lheight")	...
2.4.5	par("lwd")	...
2.4.6	par("lwd")	...
2.4.7	par("lwd")	...
2.4.8	par("lwd") et par("lwd")	...
2.4.9	par("tck") et par("tck")	...

<http://pbil.univ-lyon1.fr/R/pdf/tdr75.pdf>

GGPlot

<http://ggplot2.org/>

Statistiques

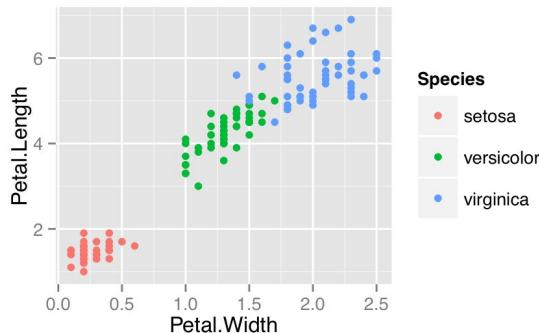


length	width	depth	trt
2	3	4	a
1	2	1	a
4	5	15	b
9	10	80	b

Mapping aesthetics

x	y	colour
2	3	a
1	2	a
4	5	b
9	10	b

scales



geom

x	y	colour
25	11	red
0	0	red
75	53	blue
200	300	blue

Frequently asked questions

Is there any correlation between these variables, samples ?

XY plots

heatmaps

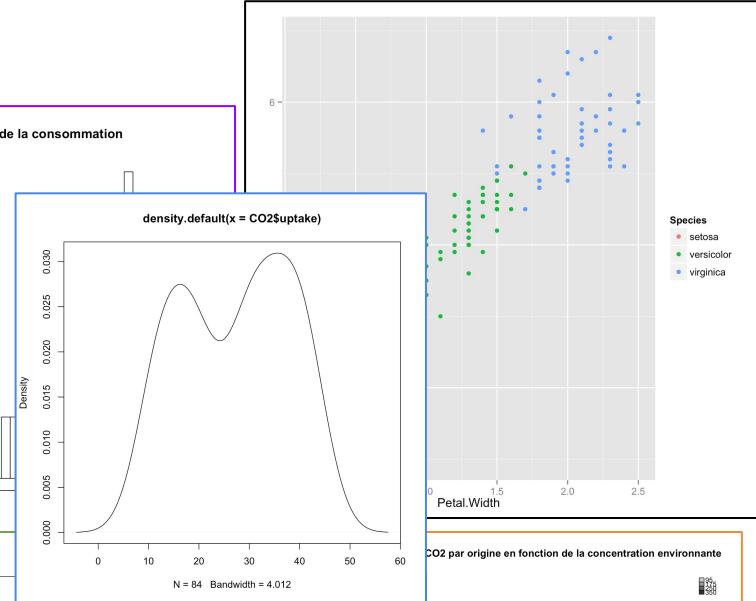
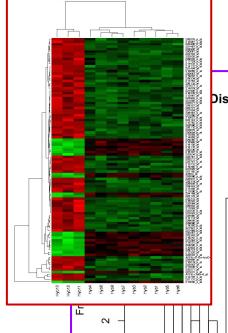
Histograms

Densities

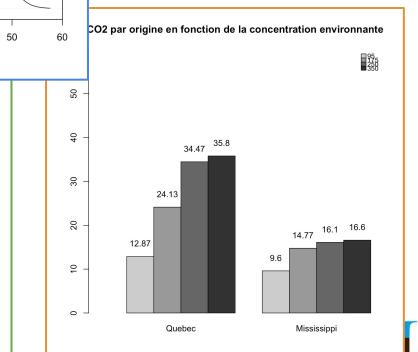
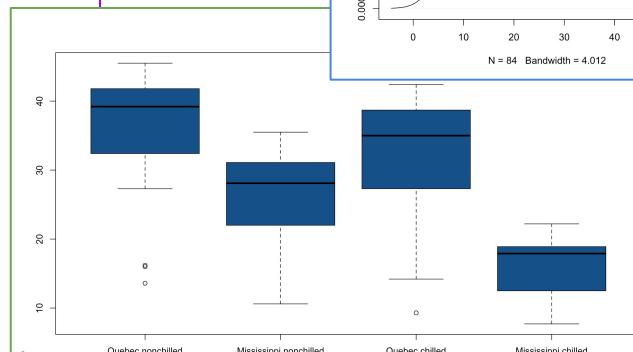
Boxplots

Barplots

How is the distribution of the data?



Is there a difference of value distribution between groups ?



Example data

```
library(ggplot2)  
data(CO2)
```

How would you?

```
library(ggplot2)  
data(CO2)
```

- Visualise if there is a correlation between CO₂ plant uptake and atmospheric CO₂ concentration
 - Which informations are contained in CO2?
 - Which kind of plot?

How would you?

```
library(ggplot2)  
data(CO2)
```

- Visualise if there is a correlation between CO₂ plant uptake and atmospheric CO₂ concentration
 - Which informations are contained in CO2?
 - Which kind of plot?

```
head(CO2)  
summary(CO2)  
str(CO2)
```

How would you?

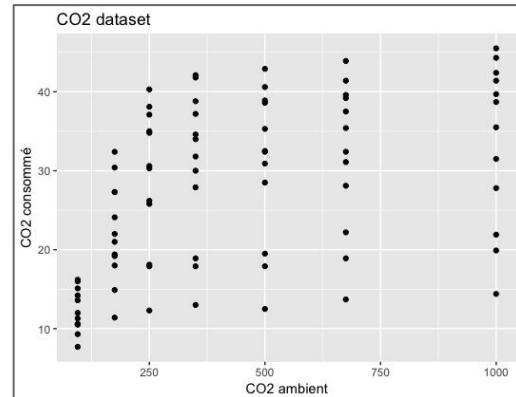
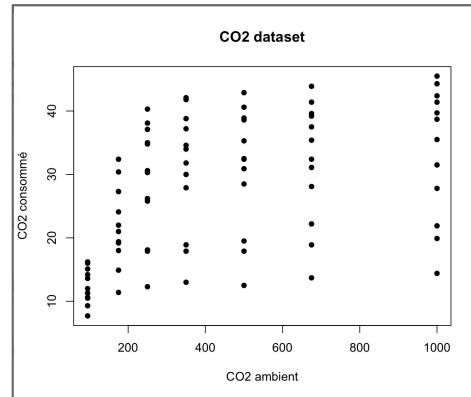
```
library(ggplot2)  
data(CO2)
```

- Visualise if there is a correlation between CO₂ plant uptake and atmospheric CO₂ concentration
 - Which informations are contained in the dataset?
 - Which kind of plot?

```
plot(CO2$conc,CO2$uptake,  
      pch=16,  
      xlab="CO2 ambient",  
      ylab="CO2 consommé",  
      main="CO2 dataset")
```

```
ggplot(CO2, aes(conc,uptake)) +  
  geom_point() +  
  xlab("CO2 ambient") +  
  ylab("CO2 consommé") +  
  ggtitle("CO2 dataset")
```

```
head(CO2)  
summary(CO2)  
str(CO2)
```



How would you?

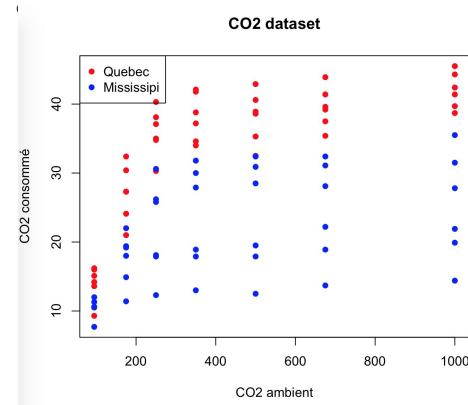
```
library(ggplot2)  
data(CO2)
```

- Visualise if there is a correlation between CO₂ plant uptake and atmospheric CO₂ concentration
 - Which informations are contained in CO2?
 - Which kind of plot?
- Is there a difference between the two locations?
 - Add color by location

How would you?

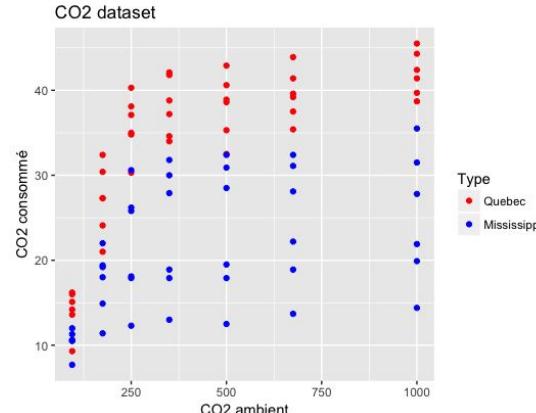
```
library(ggplot2)  
data(CO2)
```

```
point_color <- ifelse(CO2>Type=="Quebec","red","blue")  
plot(CO2$conc,CO2$uptake,  
      pch=16,  
      col= point_color,  
      xlab="CO2 ambient",  
      ylab="CO2 consommé",  
      main="CO2 dataset")  
legend("topleft", legend=c("Quebec","Mississippi"),  
      col=c("red","blue"),pch=16)
```



ptake

```
ggplot(CO2, aes(conc,uptake)) +  
  geom_point(aes(colour=Type)) +  
  xlab("CO2 ambient") +  
  ylab("CO2 consommé") +  
  ggtitle("CO2 dataset") +  
  scale_color_manual(values=c("red","blue"))
```



How would you?

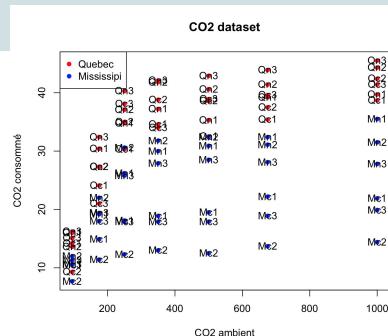
```
library(ggplot2)  
data(CO2)
```

- Visualise if there is a correlation between CO₂ plant uptake and atmospheric CO₂ concentration
 - Which informations are contained in CO2?
 - Which kind of plot?
- Is there a difference between the two locations?
 - Add color by location
 - Add the names of the plant

How would you?

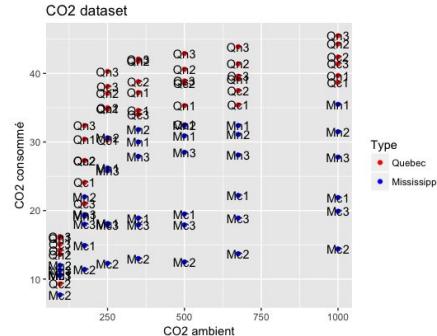
```
library(ggplot2)
data(CO2)
```

```
point_color <- ifelse(CO2>Type=="Quebec","red","blue")
plot(CO2$conc,CO2$uptake,
      pch=16,
      col= point_color,
      xlab="CO2 ambient",
      ylab="CO2 consommé",
      main="CO2 dataset")
legend("topleft", legend=c("Quebec","Mississippi"),
      col=c("red","blue"),pch=16)
text(CO2$conc,CO2$uptake, labels=CO2$Plant)
```



ptake

```
ggplot(CO2, aes(conc,uptake)) +
  geom_point(aes(colour=Type)) +
  xlab("CO2 ambient") +
  ylab("CO2 consommé") +
  ggtitle("CO2 dataset") +
  scale_color_manual(values=c("red","blue")) +
  geom_text(label=Plant)
```



How would you?

```
library(ggplot2)  
data(CO2)
```

- Visualise if there is a correlation between CO₂ plant uptake and atmospheric CO₂ concentration
 - Which informations are contained in CO2?
 - Which kind of plot?
- Is there a difference between the two locations?
 - Add color by location
 - Add the names of the plant
 - Plot a linear regression curve per location

How would you?

```
data(iris)  
data(CO2)
```

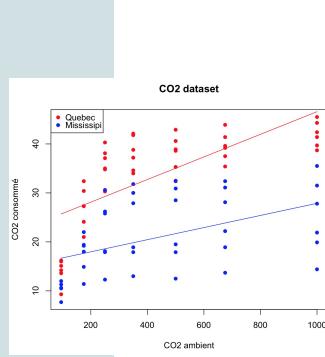
```
point_color <- ifelse(CO2$Type=="Quebec","red","blue")  
plot(CO2$conc,CO2$uptake,  
     pch=16,  
     col= point_color,  
     xlab="CO2 ambient",  
     ylab="CO2 consommé",  
     main="CO2 dataset")  
legend("topleft", legend=c("Quebec","Mississippi"),  
      col=c("red","blue"),pch=16)
```

```
CO2.miss <- CO2[CO2$Type=="Mississippi",]  
lm.miss <- lm(CO2.miss$uptake~CO2.miss$conc)  
CO2.miss$fitted <- lm.miss$fitted.values
```

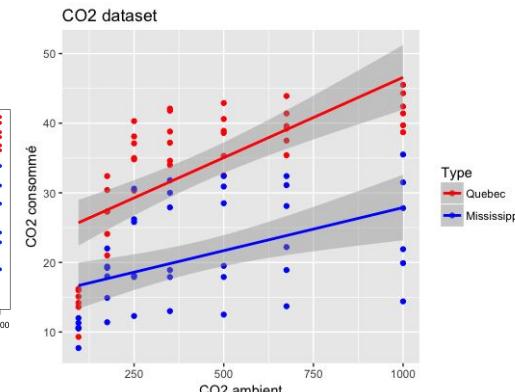
```
CO2.quebec <- CO2[CO2$Type=="Quebec",]  
lm.quebec <- lm(CO2.quebec$uptake~CO2.quebec$conc)  
CO2.quebec$fitted <- lm.quebec$fitted.values
```

```
lines(CO2.miss$conc,CO2.miss$fitted,col="blue")  
lines(CO2.quebec$conc,CO2.quebec$fitted,col="red")
```

ptake



```
ggplot(CO2, aes(conc,uptake)) +  
  geom_point(aes(colour=Type)) +  
  xlab("CO2 ambient") +  
  ylab("CO2 consommé") +  
  ggtitle("CO2 dataset") +  
  scale_color_manual(values=c("red","blue")) +  
  geom_smooth(method="lm",aes(colour=Type))
```



How would you?

```
library(ggplot2)  
data(CO2)
```

- Visualise if there is a correlation between CO₂ plant uptake and atmospheric CO₂ concentration
 - Which informations are contained in CO2?
 - Which kind of plot?
- Is there a difference between the two locations?
 - Add color by location
 - Add the names of the plant
 - Plot a linear regression curve per location
- Is there a difference between plants?

How would you?

```
library(ggplot2)  
data(CO2)
```

```
point_color <- ifelse(CO2>Type=="Quebec","red","blue")  
plot(CO2$conc,CO2$uptake,  
     pch=16,  
     col= point_color,  
     xlab="CO2 ambient",  
     ylab="CO2 consommé",  
     main="CO2 dataset")  
legend("topleft", legend=c("Quebec","Mississippi"),  
      col=c("red","blue"),pch=16)
```

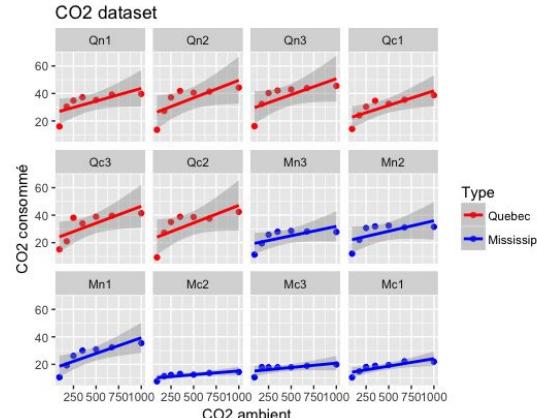
```
CO2.miss <- CO2[CO2>Type=="Mississippi",]  
lm.miss <- lm(CO2.miss$uptake~CO2.miss$conc)  
CO2.miss$fitted <- lm.miss$fitted.values
```

```
CO2.quebec <- CO2[CO2>Type=="Quebec",]  
lm.quebec <- lm(CO2.quebec$uptake~CO2.quebec$conc)  
CO2.quebec$fitted <- lm.quebec$fitted.values
```

```
lines(CO2.miss$conc,CO2.miss$fitted,col="blue")  
lines(CO2.quebec$conc,CO2.quebec$fitted,col="red")
```

ptake

```
ggplot(CO2, aes(conc,uptake)) +  
  geom_point(aes(colour=Type)) +  
  xlab("CO2 ambient") +  
  ylab("CO2 consommé") +  
  ggtitle("CO2 dataset") +  
  scale_color_manual(values=c("red","blue")) +  
  geom_smooth(method="lm",aes(colour=Type)) +  
  facet_wrap(~Plant)
```



How would you?

```
library(ggplot2)  
data(CO2)
```

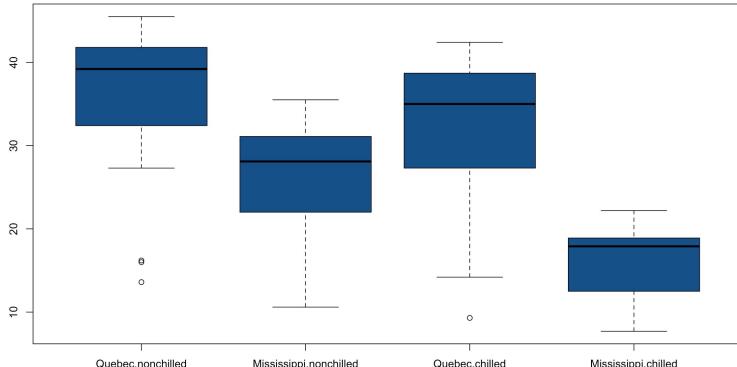
- Visualise if there is a difference in the plant CO₂ uptake according to the location and the treatment

How would you?

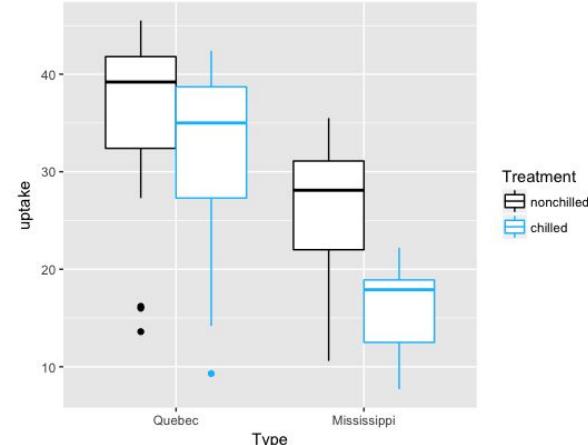
```
library(ggplot2)  
data(CO2)
```

- Visualise if there is a difference in the plant CO₂ uptake according to the location and the treatment

```
boxplot(CO2$uptake~CO2$Type*CO2$Treatment,  
       col="dodgerblue4")
```



```
ggplot(CO2, aes(x=Type,y=uptake)) +  
  geom_boxplot(aes(colour=Treatment))+  
  scale_color_manual(values=c("black",colours()[121]))
```



How would you?

```
library(ggplot2)  
data(CO2)
```

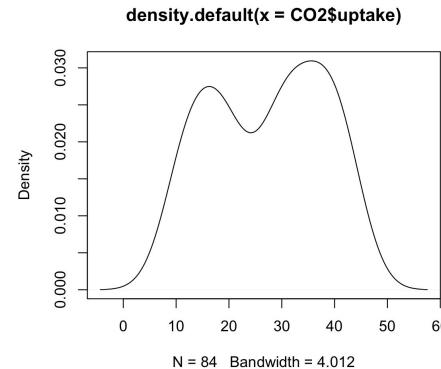
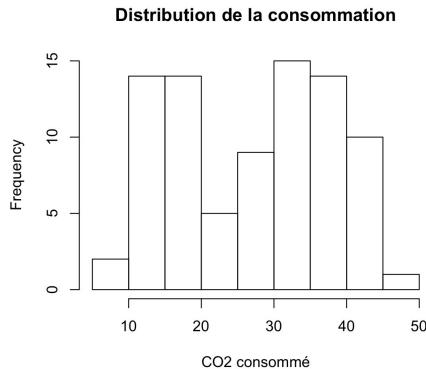
- Visualise if there is a difference in the plant CO₂ uptake according to the location and the treatment
- Check if the distribution on the plant C02 uptake looks normal

How would you?

```
library(ggplot2)  
data(CO2)
```

- Visualise if there is a difference in the plant CO₂ uptake according to the

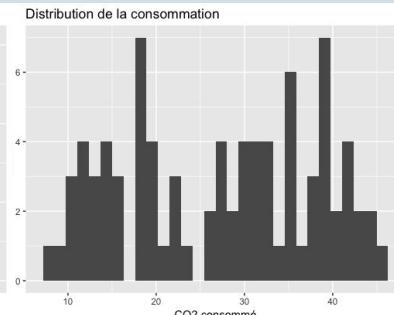
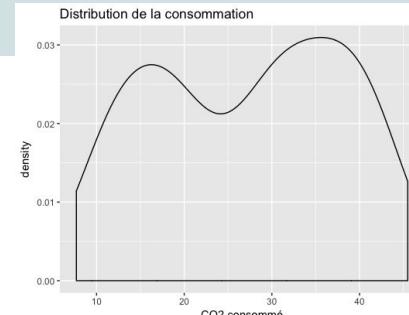
```
par(mfrow=c(1,2))  
hist(CO2$uptake,  
xlab="CO2 consommé",  
main="Distribution de la consommation")  
plot(density(CO2$uptake))  
par(mfrow=c(1,1))
```



plant CO₂ uptake looks

```
ggplot(CO2, aes(x=uptake)) +  
  geom_histogram() +  
  ggtitle("Distribution de la consommation") +  
  xlab("CO2 consommé")
```

```
ggplot(CO2, aes(x=uptake)) +  
  geom_density() +  
  ggtitle("Distribution de la consommation") +  
  xlab("CO2 consommé")
```

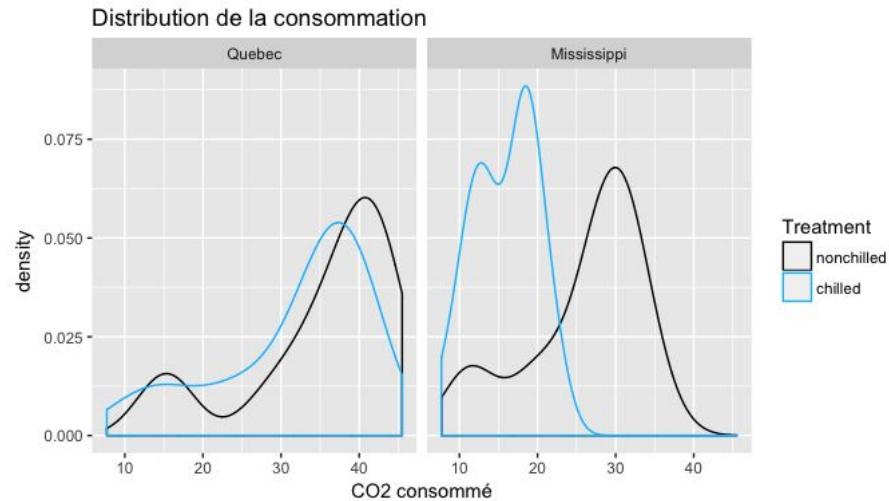


How would you?

```
library(ggplot2)  
data(CO2)
```

- Visualise if there is a difference in the plant CO₂ uptake according to the location and the treatment
- Check if the distribution on the plant C02 uptake looks normal
- Is this distribution due to some variables?

```
ggplot(CO2, aes(x=uptake)) +  
  geom_density(aes(colour=Treatment)) +  
  scale_color_manual(values=c("black",colours()[121])) +  
  ggtitle("Distribution de la consommation") +  
  xlab("CO2 consommé") +  
  facet_wrap(~Type)
```



- ❑ ggplot quickly adopted by analyst
- ❑ ggbio uses ggplot to offer genomic specificities
- ❑ Adapted to read NGS data (BAM/GFF/VCF) in object as GRanges, GAlignments etc.
- ❑ The generic function autoplot() works often very well
- ❑ Combine several tracks of the same regions (reads, coverage, transcripts) using the tracks() function from rtracklayer package