A very short, sketchy, introduction to Bioconductor

Abhijit Dasgupta

Fall, 2019

Bioconductor

Bioconductor provides tools for the analysis and comprehension of high-throughput genomic and biological data, using R.

- 1823 packages
- Covers the bioinformatic pipeline

- Analysis [`GenomicRanges`, `Biostrings`, `GenomicAlignments`, `SummarizedExperiment`]
- Annotation (species/platform specific, system) [`biomaRt`, `org.Hs.eg.db`, `GO.db`, `KEGG.db`]
- Experiments [`TENxPBMCData`, `airway`, `ALL`]
- Workflows [`rnaseqGene`, `TCGAWorkflow`]

Bioconductor

Bioconductor v. 3.10 packages

- Software (1823)
  - AssayDomain (732)
  - BiologicalQuestion (756)
  - Infrastructure (404)
  - ResearchField (810)
  - StatisticalMethod (652)
  - Technology (1160)
  - WorkflowStep (986)
- AnnotationData (953)
- ExperimentData (385)
- Workflow (27)

- Software (1823)
- AnnotationData (953)
- ExperimentData (385)
  - AssayDomainData (72)
  - DiseaseModel (88)
  - OrganismData (132)
  - PackageTypeData (27)
  - RepositoryData (91)
  - ReproducibleResearch (20)
  - SpecimenSource (101)
  - TechnologyData (254)
- Workflow (27)

- Software (1823)
- AnnotationData (953)
  - ChipManufacturer (388)
  - ChipName (196)
  - CustomArray (2)
  - CustomDBSchema (5)
  - FunctionalAnnotation (29)
  - Organism (618)
  - PackageType (664)
  - SequenceAnnotation (1)
- ExperimentData (385)
- Workflow (27)

Installing Bioconductor packages

Bioconductor is a separate repository and system which uses R. So the process is a bit different than `install.packages`. The following works for R version 3.5 and higher.

```
install.packages("BiocManager")
BiocManager::install(c('Biobase','limma','hgu95av2.db','Biostrings'))
```

There are several packages that are often installed for each Bioconductor package, and some have functions that have the same name as one's you've used. So

- Use `package::function` format for calling functions from non-Bioconductor packages

Bioconductor basics

```
library(Biostrings)
dna <- DNAStringSet(c("AACAT", "GGCGCCT"))
reverseComplement(dna)
```

```
#>    A DNAStringSet instance of length 2
#>      width seq
#>  [1]     5 ATGTT
#>  [2]     7 AGGCGCC
```

```
library(Biostrings)
data("phiX174Phage")
phiX174Phage
```

```
#>    A DNAStringSet instance of length 6
#>      width seq                                                names
#>  [1]  5386 GAGTTTTATCGCTTCCATGACGCAGAA...AAAATGATTGGCGTATCCAACCTGCA Genbank
#>  [2]  5386 GAGTTTTATCGCTTCCATGACGCAGAA...AAAATGATTGGCGTATCCAACCTGCA RF70s
#>  [3]  5386 GAGTTTTATCGCTTCCATGACGCAGAA...AAAATGATTGGCGTATCCAACCTGCA SS78
#>  [4]  5386 GAGTTTTATCGCTTCCATGACGCAGAA...AAAATGATTGGCGTATCCAACCTGCA Bull
#>  [5]  5386 GAGTTTTATCGCTTCCATGACGCAGAA...AAAATGATTGGCGTATCCAACCTGCA G97
#>  [6]  5386 GAGTTTTATCGCTTCCATGACGCAGAA...AAAATGATTGGCGTATCCAACCTGCA NEB03
```

Bioconductor basics

```
letterFrequency(phiX174Phage, 'GC', as.prob=TRUE)
```

```
#>             G|C
#>  [1,] 0.4476420
#>  [2,] 0.4472707
#>  [3,] 0.4472707
#>  [4,] 0.4470850
#>  [5,] 0.4472707
#>  [6,] 0.4470850
```

Bioconductor data structures

- So far we've seen the `data.frame` or `tibble` be the unit of data storage
- In Bioconductor, data are stored in **containers** which can contain many elements of data for an experiment
    - Actual quantitative results of experiments
    - Phenotype data
    - Genotype meta-data
    - Results of analysis
- In Bioconductor workflows, the same container is updated with new elements, which can then be accessed using accessor functions

An ExpressionSet

```
library(Biobase)
data('sample.ExpressionSet')
str(sample.ExpressionSet)
```

```
#>  Formal class 'ExpressionSet' [package "Biobase"] with 7 slots
#>    ..@ experimentData   :Formal class 'MIAME' [package "Biobase"] with 13 slots
#>    .. .. ..@ name             : chr "Pierre Fermat"
#>    .. .. ..@ lab              : chr "Francis Galton Lab"
#>    .. .. ..@ contact          : chr "pfermat@lab.not.exist"
#>    .. .. ..@ title            : chr "Smoking-Cancer Experiment"
#>    .. .. ..@ abstract         : chr "An example object of expression set (ExpressionSet) class"
#>    .. .. ..@ url              : chr "www.lab.not.exist"
#>    .. .. ..@ pubMedIds        : chr ""
#>    .. .. ..@ samples          : list()
#>    .. .. ..@ hybridizations   : list()
#>    .. .. ..@ normControls     : list()
#>    .. .. ..@ preprocessing    : list()
#>    .. .. ..@ other            :List of 1
#>    .. .. .. ..$ notes: chr "An example object of expression set (exprSet) class"
#>    .. .. ..@ .__classVersion__:Formal class 'Versions' [package "Biobase"] with 1 slot
#>    .. .. .. .. ..@ .Data:List of 2
#>    .. .. .. .. .. ..$ : int [1:3] 1 0 0
#>    .. .. .. .. .. ..$ : int [1:3] 1 1 0
#>    ..@ assayData        :<environment: 0x7fee1c47dc90>
#>    ..@ phenoData        :Formal class 'AnnotatedDataFrame' [package "Biobase"] with 4 slots
#>    .. .. ..@ varMetadata      :'data.frame':    3 obs. of  1 variable:
#>    .. .. .. ..$ labelDescription: chr [1:3] "Female/Male" "Case/Control" "Testing Score"
#>    .. .. ..@ data             :'data.frame':    26 obs. of  3 variables:
```

An ExpressionSet

These objects are based on a different R structure. Instead of extracting elements using `$`, this structure uses **slots** which are accessed using `@`

```
slotNames(sample.ExpressionSet)
```

```
#>  [1] "experimentData"    "assayData"         "phenoData"         "featureData"
#>  [5] "annotation"        "protocolData"      ".__classVersion__"
```

```
sample.ExpressionSet@phenoData
```

```
#>  An object of class 'AnnotatedDataFrame'
#>    sampleNames: A B ... Z (26 total)
#>    varLabels: sex type score
#>    varMetadata: labelDescription
```

An ExpressionSet

We almost never use `@`. Instead we use *accessor functions* to extract data

```
pData(sample.ExpressionSet) # Phenotype data
```

```
#>       sex    type score
#>  A Female Control  0.75
#>  B   Male    Case  0.40
#>  C   Male Control  0.73
#>  D   Male    Case  0.42
#>  E Female    Case  0.93
#>  F   Male Control  0.22
#>  G   Male    Case  0.96
#>  H   Male    Case  0.79
#>  I Female    Case  0.37
#>  J   Male Control  0.63
#>  K   Male    Case  0.26
#>  L Female Control  0.36
#>  M   Male    Case  0.41
#>  N   Male    Case  0.80
#>  O Female    Case  0.10
#>  P Female Control  0.41
#>  Q Female    Case  0.16
#>  R   Male Control  0.72
#>  S   Male    Case  0.17
#>  T Female    Case  0.74
#>  U   Male Control  0.35
#>  V Female Control  0.77
```

An ExpressionSet

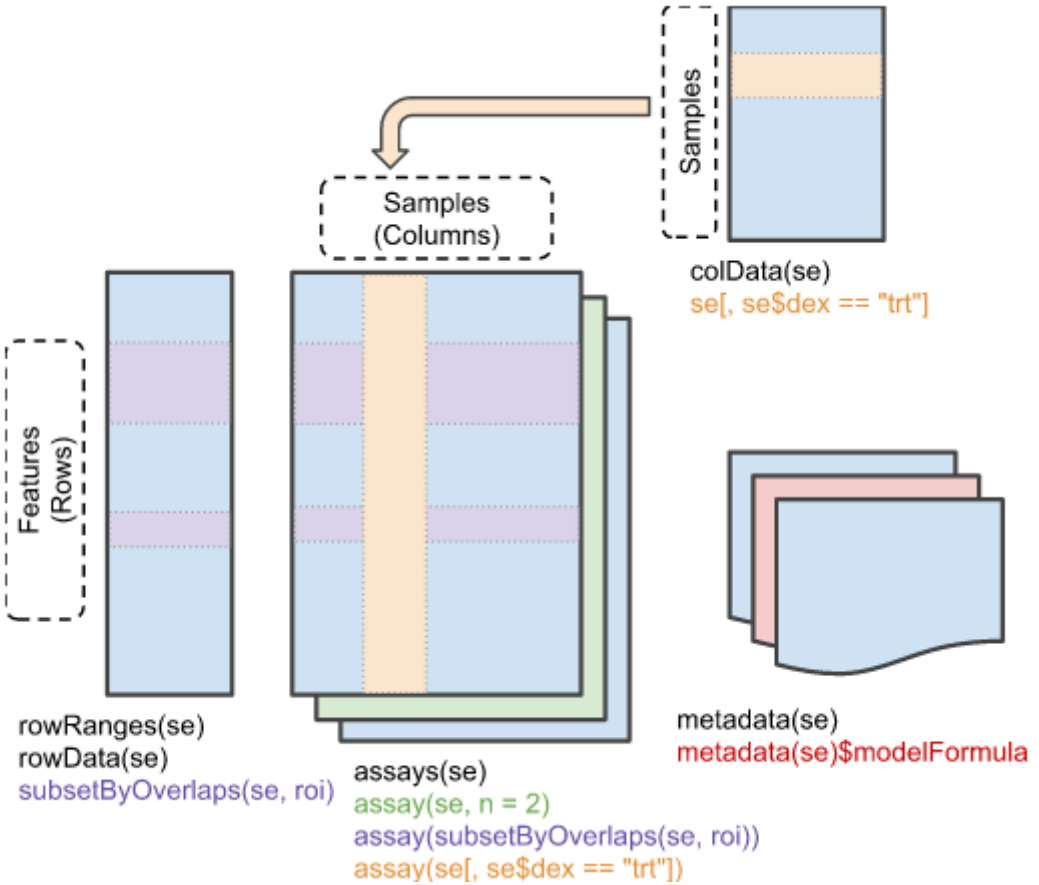
We almost never use `@`. Instead we use *accessor functions* to extract data

```
head(exprs(sample.ExpressionSet)) # Expression data
```

```
#>                        A         B        C        D        E       F        G       H
#>  AFFX-MurIL2_at  192.7420  85.75330 176.7570 135.5750 64.49390 76.3569 160.5050 65.9631
#>  AFFX-MurIL10_at  97.1370 126.19600  77.9216  93.3713 24.39860 85.5088  98.9086 81.6932
#>  AFFX-MurIL4_at   45.8192   8.83135  33.0632  28.7072  5.94492 28.2925  30.9694 14.7923
#>  AFFX-MurFAS_at   22.5445   3.60093  14.6883  12.3397 36.86630 11.2568  23.0034 16.2134
#>  AFFX-BioB-5_at   96.7875  30.43800  46.1271  70.9319 56.17440 42.6756  86.5156 30.7927
#>  AFFX-BioB-M_at   89.0730  25.84610  57.2033  69.9766 49.58220 26.1262  75.0083 42.3352
#>                        I         J        K       L       M       N       O       P
#>  AFFX-MurIL2_at  56.9039 135.60800 63.44320 78.2126 83.0943 89.3372 91.0615 95.9377
#>  AFFX-MurIL10_at 97.8015  90.48380 70.57330 94.5418 75.3455 68.5827 87.4050 84.4581
#>  AFFX-MurIL4_at  14.2399  34.48740 20.35210 14.1554 20.6251 15.9231 20.1579 27.8139
#>  AFFX-MurFAS_at  12.0375   4.54978  8.51782 27.2852 10.1616 20.2488 15.7849 14.3276
#>  AFFX-BioB-5_at  19.7183  46.35200 39.13260 41.7698 80.2197 36.4903 36.4021 35.3054
#>  AFFX-BioB-M_at  41.1207  91.53070 39.91360 49.8397 63.4794 24.7007 47.4641 47.3578
#>                        Q        R         S       T         U        V       W
#>  AFFX-MurIL2_at  179.8450 152.4670 180.83400 85.4146 157.98900 146.8000 93.8829
#>  AFFX-MurIL10_at  87.6806 108.0320 134.26300 91.4031  -8.68811  85.0212 79.2998
#>  AFFX-MurIL4_at   32.7911  33.5292  19.81720 20.4190  26.87200  31.1488 22.3420
#>  AFFX-MurFAS_at   15.9488  14.6753  -7.91911 12.8875  11.91860  12.8324 11.1390
#>  AFFX-BioB-5_at   58.6239 114.0620  93.44020 22.5168  48.64620  90.2215 42.0053
#>  AFFX-BioB-M_at   58.1331 104.1220 115.83100 58.1224  73.42210  64.6066 40.3068
#>                         X       Y         Z
#>  AFFX-MurIL2_at  103.85500 64.4340 175.61500
```

SummarizedExperiment

This is a more common structure related to modern experiments with different technologies



An Example

```
library(SummarizedExperiment)
data(airway, package="airway")
se <- airway
se
```

```
#>  class: RangedSummarizedExperiment
#>  dim: 64102 8
#>  metadata(1): ''
#>  assays(1): counts
#>  rownames(64102): ENSG00000000003 ENSG00000000005 ... LRG_98 LRG_99
#>  rowData names(0):
#>  colnames(8): SRR1039508 SRR1039509 ... SRR1039520 SRR1039521
#>  colData names(9): SampleName cell ... Sample BioSample
```

An Example

Count data from the scRNA-seq experiment

```
assay(se)
```

```
#>                 SRR1039508 SRR1039509 SRR1039512 SRR1039513 SRR1039516 SRR1039517
#>  ENSG00000000003        679        448        873        408       1138       1047
#>  ENSG00000000005          0          0          0          0          0          0
#>  ENSG00000000419        467        515        621        365        587        799
#>  ENSG00000000457        260        211        263        164        245        331
#>  ENSG00000000460         60         55         40         35         78         63
#>  ENSG00000000938          0          0          2          0          1          0
#>  ENSG00000000971       3251       3679       6177       4252       6721      11027
#>  ENSG00000001036       1433       1062       1733        881       1424       1439
#>  ENSG00000001084        519        380        595        493        820        714
#>  ENSG00000001167        394        236        464        175        658        584
#>  ENSG00000001460        172        168        264        118        241        210
#>  ENSG00000001461       2112       1867       5137       2657       2735       2751
#>  ENSG00000001497        524        488        638        357        676        806
#>  ENSG00000001561         71         51        211        156         23         38
#>  ENSG00000001617        555        394        905        415        727        697
#>  ENSG00000001626         10          2          9          2         10          6
#>  ENSG00000001629       1660       1251       2259       1079       2462       2514
#>  ENSG00000001630         59         54         66         23         84         87
#>  ENSG00000001631        729        692        943        475       1034       1163
#>  ENSG00000002016        201        161        256         99        268        257
#>  ENSG00000002079          3          0          3          1          4          0
#>  ENSG00000002330        206        174        184        111        194        260
```

An Example

Genomic ranges for each transcript

```
rowRanges(se)
```

```
#>  GRangesList object of length 64102:
#>  $ENSG00000000003
#>  GRanges object with 17 ranges and 2 metadata columns:
#>         seqnames            ranges strand |   exon_id       exon_name
#>            <Rle>         <IRanges>  <Rle> | <integer>     <character>
#>     [1]        X 99883667-99884983      - |    667145 ENSE00001459322
#>     [2]        X 99885756-99885863      - |    667146 ENSE00000868868
#>     [3]        X 99887482-99887565      - |    667147 ENSE00000401072
#>     [4]        X 99887538-99887565      - |    667148 ENSE00001849132
#>     [5]        X 99888402-99888536      - |    667149 ENSE00003554016
#>     ...      ...                 ...    ... .       ...               ...
#>    [13]        X 99890555-99890743      - |    667156 ENSE00003512331
#>    [14]        X 99891188-99891686      - |    667158 ENSE00001886883
#>    [15]        X 99891605-99891803      - |    667159 ENSE00001855382
#>    [16]        X 99891790-99892101      - |    667160 ENSE00001863395
#>    [17]        X 99894942-99894988      - |    667161 ENSE00001828996
#>
#>  ...
#>  <64101 more elements>
#>  -------
#>  seqinfo: 722 sequences (1 circular) from an unspecified genome
```

An Example

Phenotype data

```
colData(se)
```

```
#>  DataFrame with 8 rows and 9 columns
#>             SampleName     cell      dex    albut        Run avgLength Experiment
#>               <factor> <factor> <factor> <factor>   <factor> <integer>   <factor>
#>  SRR1039508 GSM1275862   N61311    untrt    untrt SRR1039508       126  SRX384345
#>  SRR1039509 GSM1275863   N61311      trt    untrt SRR1039509       126  SRX384346
#>  SRR1039512 GSM1275866  N052611    untrt    untrt SRR1039512       126  SRX384349
#>  SRR1039513 GSM1275867  N052611      trt    untrt SRR1039513        87  SRX384350
#>  SRR1039516 GSM1275870  N080611    untrt    untrt SRR1039516       120  SRX384353
#>  SRR1039517 GSM1275871  N080611      trt    untrt SRR1039517       126  SRX384354
#>  SRR1039520 GSM1275874  N061011    untrt    untrt SRR1039520       101  SRX384357
#>  SRR1039521 GSM1275875  N061011      trt    untrt SRR1039521        98  SRX384358
#>                Sample    BioSample
#>              <factor>     <factor>
#>  SRR1039508 SRS508568 SAMN02422669
#>  SRR1039509 SRS508567 SAMN02422675
#>  SRR1039512 SRS508571 SAMN02422678
#>  SRR1039513 SRS508572 SAMN02422670
#>  SRR1039516 SRS508575 SAMN02422682
#>  SRR1039517 SRS508576 SAMN02422673
#>  SRR1039520 SRS508579 SAMN02422683
#>  SRR1039521 SRS508580 SAMN02422677
```

An Example

Experimental meta-data

```
metadata(se)
```

```
#>  [[1]]
#>  Experiment data
#>    Experimenter name: Himes BE
#>    Laboratory: NA
#>    Contact information:
#>    Title: RNA-Seq transcriptome profiling identifies CRISPLD2 as a glucocorticoid responsive gene that modulate
#>    URL: http://www.ncbi.nlm.nih.gov/pubmed/24926665
#>    PMIDs: 24926665
#>
#>    Abstract: A 226 word abstract is available. Use 'abstract' method.
```

An Example

Data subsetting

```
se[1:5, 1:3]
```

```
#>  class: RangedSummarizedExperiment
#>  dim: 5 3
#>  metadata(1): ''
#>  assays(1): counts
#>  rownames(5): ENSG00000000003 ENSG00000000005 ENSG
#>    ENSG00000000460
#>  rowData names(0):
#>  colnames(3): SRR1039508 SRR1039509 SRR1039512
#>  colData names(9): SampleName cell ... Sample BioS
```

```
se[,se$cell=='N61311']
```

```
#>  class: RangedSummarizedExperiment
#>  dim: 64102 2
#>  metadata(1): ''
#>  assays(1): counts
#>  rownames(64102): ENSG00000000003 ENSG00000000005
#>  rowData names(0):
#>  colnames(2): SRR1039508 SRR1039509
#>  colData names(9): SampleName cell ... Sample BioS
```

Annotation

biomaRt

The `biomaRt` package allows access to many public annotation databases

```
library(biomaRt)
ensemblDB <- useMart('ensembl')
searchDatasets(mart=ensemblDB, pattern='Human')
```

```
#>                  dataset              description    version
#>  85 hsapiens_gene_ensembl Human genes (GRCh38.p13) GRCh38.p13
```

```
ensemblHuman <- useMart('ensembl', dataset='hsapiens_gene_ensembl')
```

Identifying attributes

```
searchAttributes(mart=ensemblHuman, pattern='affy')
```

```
#>                     name                description        page
#>  104          affy_hc_g110          AFFY HC G110 probe feature_page
#>  105         affy_hg_focus         AFFY HG Focus probe feature_page
#>  106         affy_hg_u133a         AFFY HG U133A probe feature_page
#>  107       affy_hg_u133a_2       AFFY HG U133A 2 probe feature_page
#>  108         affy_hg_u133b         AFFY HG U133B probe feature_page
#>  109   affy_hg_u133_plus_2   AFFY HG U133 Plus 2 probe feature_page
#>  110          affy_hg_u95a          AFFY HG U95A probe feature_page
#>  111        affy_hg_u95av2        AFFY HG U95Av2 probe feature_page
#>  112          affy_hg_u95b          AFFY HG U95B probe feature_page
#>  113          affy_hg_u95c          AFFY HG U95C probe feature_page
#>  114          affy_hg_u95d          AFFY HG U95D probe feature_page
#>  115          affy_hg_u95e          AFFY HG U95E probe feature_page
#>  116          affy_hta_2_0          AFFY HTA 2 0 probe feature_page
#>  117   affy_huex_1_0_st_v2   AFFY HuEx 1 0 st v2 probe feature_page
#>  118         affy_hugenefl         AFFY HuGeneFL probe feature_page
#>  119 affy_hugene_1_0_st_v1 AFFY HuGene 1 0 st v1 probe feature_page
#>  120 affy_hugene_2_0_st_v1 AFFY HuGene 2 0 st v1 probe feature_page
#>  121        affy_primeview        AFFY PrimeView probe feature_page
#>  122         affy_u133_x3p         AFFY U133 X3P probe feature_page
```

Identifying attributes

```
searchAttributes(mart=ensemblHuman, pattern='hgnc')
```

```
#>                name        description        page
#>  62         hgnc_id           HGNC ID feature_page
#>  63     hgnc_symbol        HGNC symbol feature_page
#>  94 hgnc_trans_name Transcript name ID feature_page
```

Annotating probsets

We first grab some probesets from the `sample.ExpressionSet` Affy experiment

```
affyIDs <- rownames(featureData(sample.ExpressionSet))
```

Now let's find annotation

```
getBM(attributes = c('affy_hg_u95av2','hgnc_symbol'),
      filters = 'affy_hg_u95av2',
      values = affyIDs[200:203],
      mart = ensemblHuman)
```

```
#>    affy_hg_u95av2 hgnc_symbol
#>  1     31439_f_at         RHD
#>  2       31440_at        TCF7
#>  3     31439_f_at        RHCE
#>  4       31442_at     CEACAM5
```

A RNA-seq pipeline

The workflow

- Exploratory data analysis
- Differential expression analysis with DESeq2
- Visualization

- We will start after reads have been aligned to a reference genome and reads overlapping known genes have been counted

The experiment

- In the experiment, four primary human airway smooth muscle cell lines were treated with 1 micromolar dexamethasone for 18 hours.
- For each of the four cell lines, we have a treated and an untreated sample.

Get data

```
# BiocManager::install('airway')
library(airway)
data(airway)
se <- airway
head(assay(airway))
```

```
#>                 SRR1039508 SRR1039509 SRR1039512 SRR1039513 SRR1039516 SRR1039517
#>  ENSG00000000003        679        448        873        408       1138       1047
#>  ENSG00000000005          0          0          0          0          0          0
#>  ENSG00000000419        467        515        621        365        587        799
#>  ENSG00000000457        260        211        263        164        245        331
#>  ENSG00000000460         60         55         40         35         78         63
#>  ENSG00000000938          0          0          2          0          1          0
#>                 SRR1039520 SRR1039521
#>  ENSG00000000003        770        572
#>  ENSG00000000005          0          0
#>  ENSG00000000419        417        508
#>  ENSG00000000457        233        229
#>  ENSG00000000460         76         60
#>  ENSG00000000938          0          0
```

Create a DESeqDataSet

```
# BiocManager::install('DESeq2')
library("DESeq2")
dds <- DESeqDataSet(se, design = ~ cell + dex)
dds
```

```
#>  class: DESeqDataSet
#>  dim: 64102 8
#>  metadata(2): '' version
#>  assays(1): counts
#>  rownames(64102): ENSG00000000003 ENSG00000000005 ... LRG_98 LRG_99
#>  rowData names(0):
#>  colnames(8): SRR1039508 SRR1039509 ... SRR1039520 SRR1039521
#>  colData names(9): SampleName cell ... Sample BioSample
```

Run differential expression pipeline

```
dds <- DESeq(dds)
```

```
#>  estimating size factors
```

```
#>  estimating dispersions
```

```
#>  gene-wise dispersion estimates
```

```
#>  mean-dispersion relationship
```

```
#>  final dispersion estimates
```

```
#>  fitting model and testing
```

Run differential expression pipeline

```
(res <- results(dds))
```

```
#>  log2 fold change (MLE): dex untrt vs trt
#>  Wald test p-value: dex untrt vs trt
#>  DataFrame with 64102 rows and 6 columns
#>                          baseMean      log2FoldChange             lfcSE
#>                         <numeric>           <numeric>         <numeric>
#>  ENSG00000000003 708.602169691234   0.381253982523043 0.100654411867396
#>  ENSG00000000005               0                  NA                NA
#>  ENSG00000000419 520.297900552084  -0.206812601085043 0.112218646508368
#>  ENSG00000000457 237.163036796015 -0.0379204312050886 0.143444654933749
#>  ENSG00000000460 57.9326331250967  0.0881681758708941 0.287141822933388
#>  ...                          ...                 ...               ...
#>  LRG_94                         0                  NA                NA
#>  LRG_96                         0                  NA                NA
#>  LRG_97                         0                  NA                NA
#>  LRG_98                         0                  NA                NA
#>  LRG_99                         0                  NA                NA
#>                             stat             pvalue               padj
#>                        <numeric>          <numeric>          <numeric>
#>  ENSG00000000003   3.78775232451125 0.000152016273081244 0.00128362968201811
#>  ENSG00000000005                 NA                 NA                 NA
#>  ENSG00000000419  -1.84294328545142  0.0653372915124384  0.196546085664315
#>  ENSG00000000457 -0.264355832725887   0.791505741607837   0.911459479590443
#>  ENSG00000000460  0.307054454729667   0.758801923977348   0.895032784968832
#>  ...                            ...                ...                ...
#>  LRG_94                          NA                 NA                 NA
#>  LRG_96                          NA                 NA                 NA
```

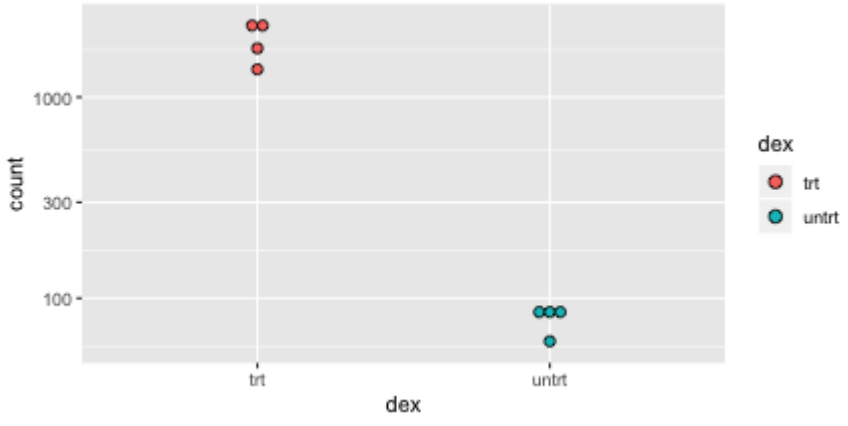
Summarizing results

```
library(tidyverse)
as.data.frame(res) %>%
  rownames_to_column(var = 'ID') %>%
  filter(padj < 0.1) %>%
  arrange(desc(abs(log2FoldChange))) %>% head()
```

```
#>                ID   baseMean log2FoldChange     lfcSE       stat       pvalue
#>  1 ENSG00000179593  67.243048      -9.505975 1.0545032  -9.014647 1.975049e-19
#>  2 ENSG00000109906 385.071029      -7.352626 0.5363887 -13.707645 9.137621e-43
#>  3 ENSG00000250978  56.318194      -6.327383 0.6777973  -9.335214 1.007876e-20
#>  4 ENSG00000132518   5.654654      -5.885114 1.3240439  -4.444803 8.797262e-06
#>  5 ENSG00000128285   6.624741       5.325904 1.2578147   4.234251 2.293144e-05
#>  6 ENSG00000127954 286.384119      -5.207158 0.4930818 -10.560435 4.545484e-26
#>            padj
#>  1 1.253739e-17
#>  2 2.256617e-40
#>  3 7.210311e-19
#>  4 1.000612e-04
#>  5 2.380012e-04
#>  6 5.058395e-24
```

A visualization

```
topGene <- rownames(res)[which.min(res$padj)]
dat <- plotCounts(dds, gene=topGene, intgroup=c("dex"), returnData=TRUE)
ggplot(dat, aes(x = dex, y = count, fill=dex))+
  geom_dotplot(binaxis='y', stackdir='center')+
  scale_y_log10()
```



Another visualization

```
plotMA(res, ylim=c(-5,5))
```

A Seurat pipeline

Grab the data and convert for Seurat

```
library(TENxPBMCData)
library(Seurat)

pbmc_3k <- TENxPBMCData(dataset='pbmc3k')
colnames(pbmc_3k) <- paste0('Cell', seq_len(ncol(pbmc_3k)))
pbmc <- CreateSeuratObject( counts=assay(pbmc_3k, 'counts'),
                            project='pbmc3k',
                            min.cells=3,
                            min.features = 200)

pbmc
```

```
#>  An object of class Seurat
#>  13714 features across 2700 samples within 1 assay
#>  Active assay: RNA (13714 features)
```

A bit of QC

```
# The [[ operator can add columns to object metadata. This is a great place to stash QC stats
rownames(pbmc@assays$RNA@counts) <- r2
rownames(pbmc[['RNA']]@meta.features) <- r2
rownames(pbmc@assays$RNA@data) <- r2 # Change to gene names from Ensembl
pbmc[["percent.mt"]] <- PercentageFeatureSet(pbmc, pattern = "^MT-") # percentage in mitochondria genome
head(pbmc@meta.data)
```

```
#>        orig.ident nCount_RNA nFeature_RNA percent.mt
#>  Cell1     pbmc3k       2419          779  3.0177759
#>  Cell2     pbmc3k       4903         1352  3.7935958
#>  Cell3     pbmc3k       3147         1129  0.8897363
#>  Cell4     pbmc3k       2639          960  1.7430845
#>  Cell5     pbmc3k        980          521  1.2244898
#>  Cell6     pbmc3k       2163          781  1.6643551
```

> See how analyses results are added to the same container. The idea is to keep all the experimental information together. This was a philosophic choice maide by the Bioconductor developers, inspired by the MIAME requirements and how data are stored on genomics databases like GEO

Visualization

```
# Visualize QC metrics as a violin plot
(plt <- VlnPlot(pbmc, features = c("nFeature_RNA", "nCount_RNA", "percent.mt"), ncol = 3))
```

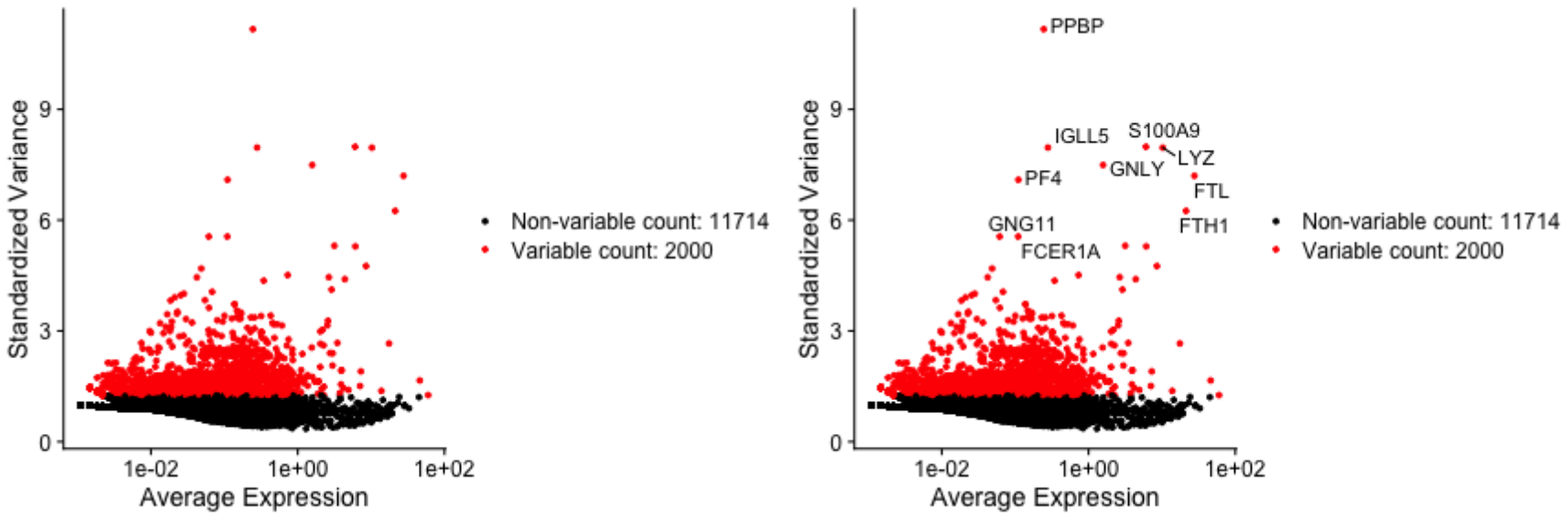
Normalization

```
pbmc <- NormalizeData(pbmc)
```

> Note, we're saving in the same object

Feature selection

```
rownames(pbmc[['RNA']]@meta.features) <- r2
pbmc <- FindVariableFeatures(pbmc, selection.method = "vst", nfeatures = 2000)

# Identify the 10 most highly variable genes
top10 <- head(VariableFeatures(pbmc), 10)

# plot variable features with and without labels
plot1 <- VariableFeaturePlot(pbmc)
plot2 <- LabelPoints(plot = plot1, points = top10, repel = TRUE)
CombinePlots(plots = list(plot1, plot2))
```



PCA

```
pbmc <- ScaleData(pbmc)
pbmc <- RunPCA(pbmc, features = VariableFeatures(object = pbmc))
print(pbmc[["pca"]], dims = 1:5, nfeatures = 5)
```

```
#>  PC_ 1
#>  Positive:  MALAT1, LTB, IL32, CD2, ACAP1
#>  Negative:  CST3, TYROBP, LST1, AIF1, FTL
#>  PC_ 2
#>  Positive:  CD79A, MS4A1, TCL1A, HLA-DQA1, HLA-DRA
#>  Negative:  NKG7, PRF1, CST7, GZMA, GZMB
#>  PC_ 3
#>  Positive:  HLA-DQA1, CD79A, CD79B, HLA-DQB1, HLA-DPB1
#>  Negative:  PPBP, PF4, SDPR, SPARC, GNG11
#>  PC_ 4
#>  Positive:  HLA-DQA1, CD79A, CD79B, HIST1H2AC, HLA-DQB1
#>  Negative:  VIM, S100A8, S100A6, S100A4, S100A9
#>  PC_ 5
#>  Positive:  GZMB, FGFBP2, NKG7, GNLY, PRF1
#>  Negative:  LTB, VIM, AQP3, PPA1, MAL
```

> Important to note that each step just adds elements to the Seurat object

PCA

```
DimPlot(pbmc, reduction = "pca")
```

t-SNE

```
pbmc <- RunTSNE(pbmc, dims=1:10)
DimPlot(pbmc, reduction='tsne')
```

Heatmaps

Heatmaps

There are several ways of doing heatmaps in R:

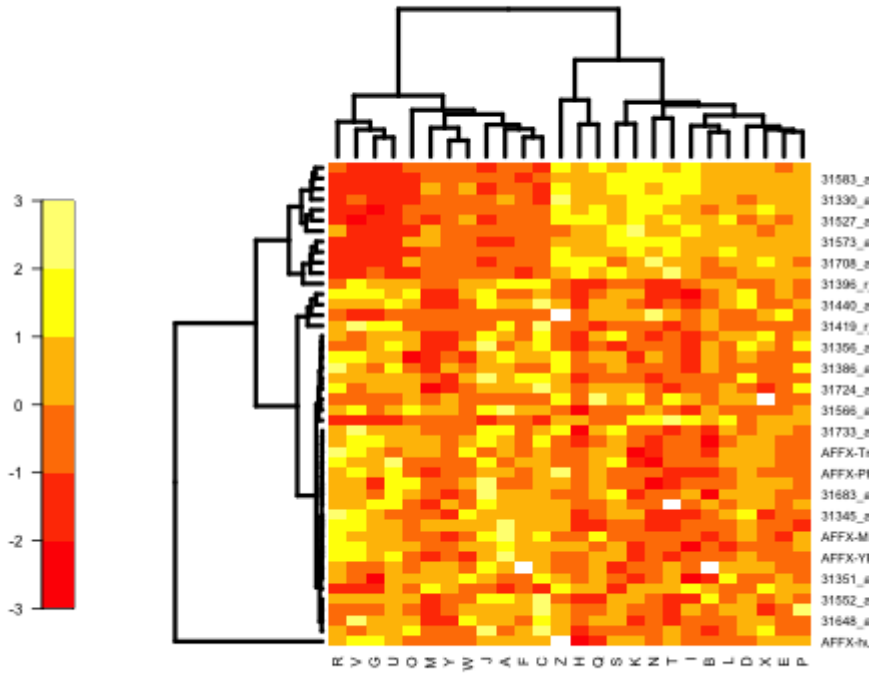
- http://sebastianraschka.com/Articles/heatmaps_in_r.html{target="_blank"}
- https://plot.ly/r/heatmaps/{target="_blank"}
- http://moderndata.plot.ly/interactive-heat-maps-for-r/{target="_blank"}
- http://www.siliconcreek.net/r/simple-heatmap-in-r-with-ggplot2{target="_blank"}
- https://rud.is/b/2016/02/14/making-faceted-heatmaps-with-ggplot2/{target="_blank"}

Some example data

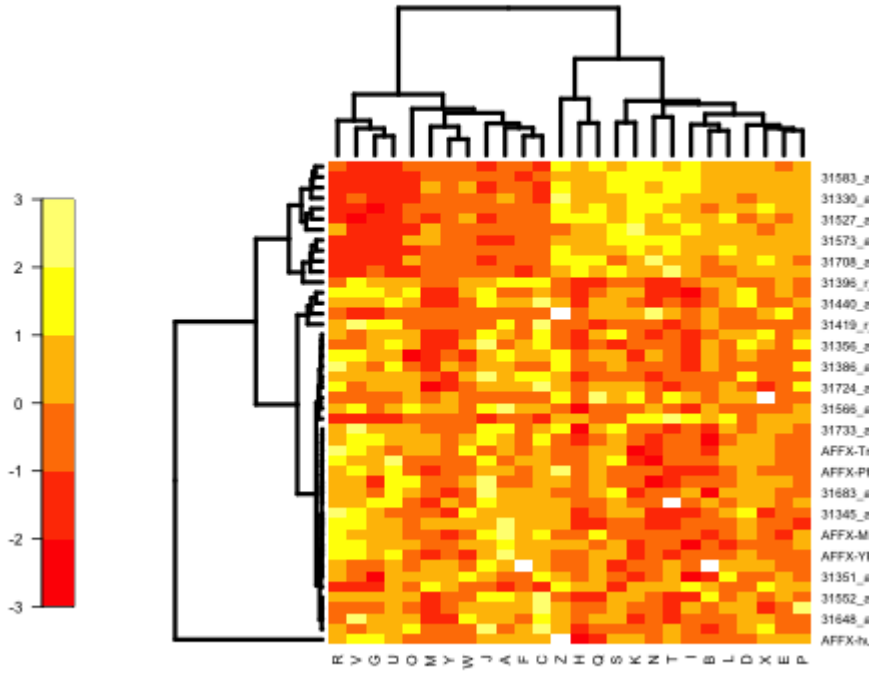
```
library(Biobase)
data(sample.ExpressionSet)
exdat <-  sample.ExpressionSet
library(limma)
design1 <- model.matrix(~type, data=pData(exdat))
lm1 <- lmFit(exprs(exdat), design1)
lm1 <- eBayes(lm1) # compute linear model for each probeset
geneID <- rownames(topTable(lm1, coef=2, num=100, adjust='none',p.value=0.05))
exdat2 <- exdat[geneID,] # Keep features with p-values < 0.05
exdat2
```

```
#>  ExpressionSet (storageMode: lockedEnvironment)
#>  assayData: 46 features, 26 samples
#>    element names: exprs, se.exprs
#>  protocolData: none
#>  phenoData
#>    sampleNames: A B ... Z (26 total)
#>    varLabels: sex type score
#>    varMetadata: labelDescription
#>  featureData: none
#>  experimentData: use 'experimentData(object)'
#>  Annotation: hgu95av2
```

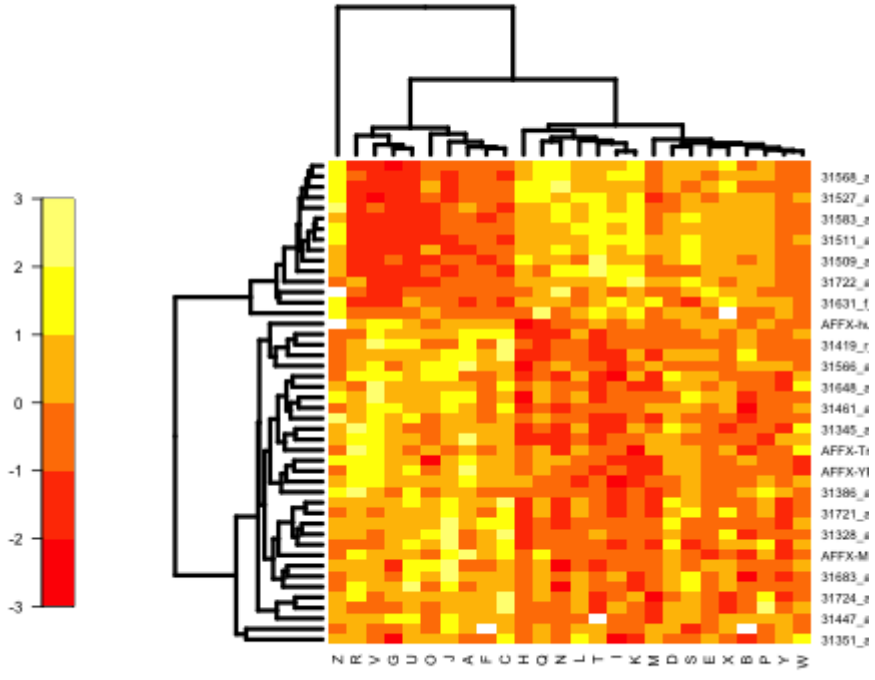
```
# BiocManager::install('Heatplus')

library(Heatplus)
reg1 <- regHeatmap(exprs(exdat2), legend=2, col=heat.colors,
                   breaks=-3:3)
plot(reg1)
```



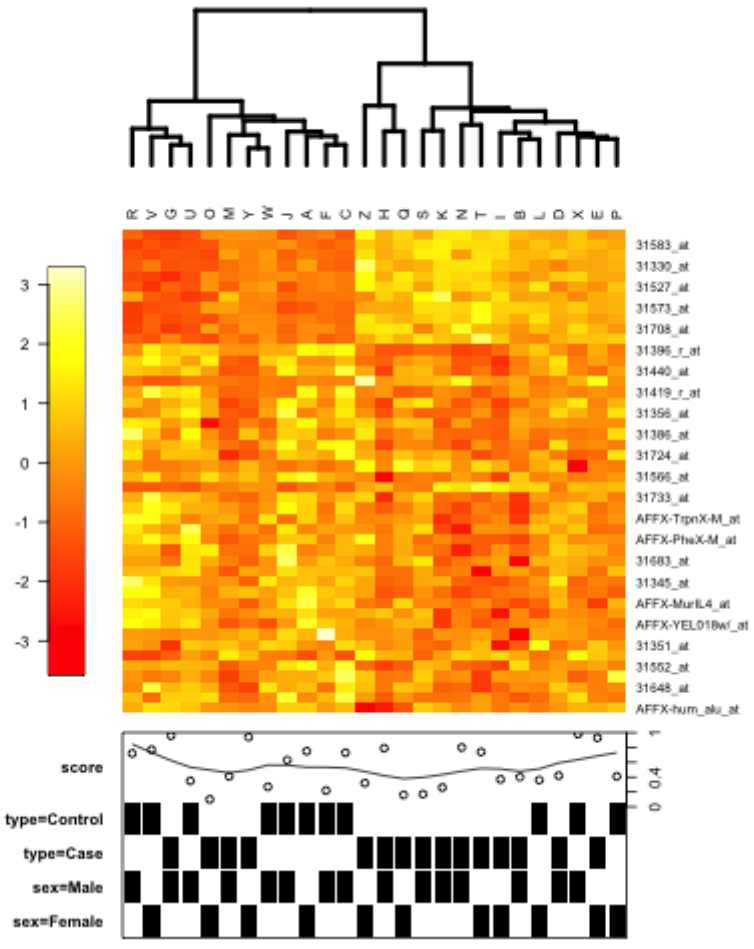
```
library(Heatplus)
reg1 <- regHeatmap(exprs(exdat2), legend=2, col=heat.colors,
                   breaks=-3:3)
plot(reg1)
```



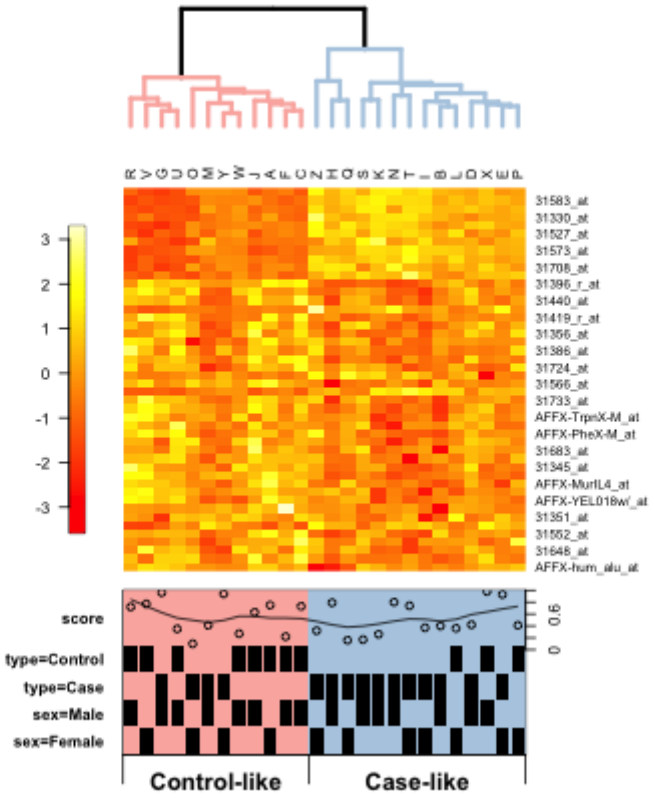
```
corrdist <- function(x) as.dist(1-cor(t(x)))
hclust.avl <-  function(x) hclust(x, method='average')
reg2 <- regHeatmap(exprs(exdat2), legend=2, col=heat.colors,
                   breaks=-3:3,
                   dendrogram = list(clustfun=hclust.avl, distfun=corrdist))
plot(reg2)
```



```
ann1 <- annHeatmap(exprs(exdat2), ann=pData(exdat2), col = heat.colors)
plot(ann1)
```



```
ann2 <- annHeatmap(exprs(exdat2), ann=pData(exdat2), col = heat.colors,
                   cluster = list(cuth=7500,
                                  label=c('Control-like','Case-like')))
plot(ann2)
```



```
# install.packages('d3heatmap')
library(d3heatmap)
d3heatmap(exprs(exdat2), distfun = corrdist,
          hclustfun = hclust.avl, scale='row')
```

Link Put your mouse over each point :)