Tutorial: Cancer Genomics Browser Data in *R* / *Bioconductor*

Martin Morgan mtmorgan@fhcrc.org

4 November, 2013

Contents

1 PAM50

We start with the material covered in the basic course, PAM50 Subtypes[1].

1.1 Data retrieval & extraction

Retrieve the data. Unfortunately, there are no programmatic ways to retrieved the data from the CGB (see the *Synapse R* client[2] for one alternative; the data from Synapse is formatted differently from the CGB, so the following is not applicable).

Click on the `datasets` button, open the TCGA Breast invasive carcinoma data, and select the `AgilentG4502_07_3` data set. Download it by clicking on the button to the right of the selection.

Unpack the downloaded data, from the command line

```
tar xzf TCGA_BRCA_G4502A_07_3-2013-10-29.tgz
```

or from within *R*

```
untar("TCGA_BRCA_G4502A_07_3-2013-10-29.tgz")
```

Switch to the directory containing the extracted files, and take a look at the directory content

[1] https://genome-cancer.ucsc.edu/proj/site/demo/#2
[2] https://www.synapse.org/#!Synapse:syn1834618

```
setwd("TCGA_BRCA_G4502A_07_3-2013-10-29")
dir()
```

```
##  [1] "AgilentG4502A_07_3"
##  [2] "AgilentG4502A_07_3.json"
##  [3] "BRCA_clinicalFeature"
##  [4] "BRCA_clinicalFeature.json"
##  [5] "BRCA_clinicalMatrix"
##  [6] "BRCA_clinicalMatrix.json"
##  [7] "collapsed_hugo_symbols_aliases_only_hg18.probeMap"
##  [8] "collapsed_hugo_symbols_aliases_only_hg18.probeMap.json"
##  [9] "md5.txt"
## [10] "sampleMap"
## [11] "sampleMap.json"
```

The files are as follows:

AgilentG4502A_07_3 Gene × sample normalized microarray expression values. HUGO gene symbols used as identifiers.
BRCA_clinicalMatrix Sample × clinical feature matrix
BRCA_clinicalFeature Description of clinical features.
collapsed_hugo_symbols_aliases_only_hg18.probeMap HUGO gene symbol start and end coordinates (genome build?)
sampleMap
***.json** Descriptions of the corresponding file.
md5.txt A file of 'check-sums' to validate that the data have been downloaded without corruption.

As a basic 'best practices', ensure that the checksums in 'md5.txt' match those of the data. The file does not have a header identifying columns `header=FALSE`, is white-space delimited `sep=""`, and contains text columns to be interpretted as character vectors rather than factors `stringsAsFactors=FALSE`

```
library(tools)
checks <- read.delim("md5.txt", header=FALSE, sep="",
    col.names=c("Sum", "File"), stringsAsFactors=FALSE)
head(checks)
```

```
##                                Sum                       File
## 1 0255e18a729d05fbdb292ccc690d711c          AgilentG4502A_07_3
## 2 3282cb7c007801ed71e160536449ffa8     AgilentG4502A_07_3.json
## 3 a29b802e527e61c2de63185d39d04262        BRCA_clinicalFeature
## 4 492da3f005988ec9af018f3318753196 BRCA_clinicalFeature.json
## 5 b0f64544342e938ce3f9ebc8fabc8808         BRCA_clinicalMatrix
## 6 273a5ccda0ddc18d580941db2303c686   BRCA_clinicalMatrix.json
```

```
stopifnot(all(md5sum(checks$File) == checks$Sum))
```

1.2 Samples

Discover information about a file by parsing the corresponding JSON. For instance, the 'BRCA_clinicalMatrix' file has the following metadata:

```
library(rjson)
str(fromJSON(file="BRCA_clinicalMatrix.json"))
```

```
## List of 18
##  $ wrangling_procedure: chr "Clinical data download from TCGA DCC, processed at UCSC into cgData reposi
```

```
##  $ wrangler           : chr "cgData TCGAscript Clinical processed on 2013-10-05,cgData TCGAscript oldCl
##  $ redistribution     : logi TRUE
##  $ description        : chr "This dataset is the TCGA Breast Cancer (BRCA) clinical data.,TCGA BRCA sam
##  $ shortTitle         : chr "BRCA Nature 2012 STable 1 sheet 1 tumor information,PAM50_RNAseq,BRCA Natu
##  $ cohort             : chr "TCGA.BRCA.sampleMap"
##  $ url                : chr "https://tcga-data.nci.nih.gov/tcgafiles/ftp_auth/distro_ftpusers/anonymous
##  $ :sampleMap         : chr "TCGA.BRCA.sampleMap"
##  $ name               : chr "BRCA_clinicalMatrix"
##  $ upToDate           : chr "4.0,2.1,1.5"
##  $ longTitle          : chr "TCGA BRCA clinical data from Nature 2012 STable1 sheet 1 tumor information
##  $ version            : chr "2013-10-05"
##  $ path               : chr "data_flatten/public/TCGA/BRCA/BRCA_clinicalMatrix"
##  $ dataProducer       : chr "TCGA biospecimen core resource,https://www.synapse.org/#!Synapse:syn175692
##  $ cgDataVersion      : num 1
##  $ type               : chr "clinicalMatrix"
##  $ outOfDate          : chr "Yes"
##  $ :clinicalFeature   : chr "BRCA_clinicalFeature"
```

The 'BRCA_clinicalMatrix' file can be input into *R*, after a little exploring, with the following. We use `row.names=1` to use the first column ('sampleID') as the row names of the input object, and translate empty entries in `character` columns as `NA`.

```
clinical <- read.delim("BRCA_clinicalMatrix", header=TRUE, sep="\t",
     row.names=1, na.strings="", stringsAsFactors=FALSE)
class(clinical)
```

```
## [1] "data.frame"
```

```
dim(clinical)
```

```
## [1] 1147  187
```

```
clinical[1:5, 1:5]
```

```
##                 AJCC_Stage_nature2012 Age_at_Initial_Pathologic_Diagnosis_nature2012
## TCGA-E9-A1RD-11                  <NA>                                             67
## TCGA-E9-A1RC-01             Stage IIIC                                            56
## TCGA-AC-A3TN-01                  <NA>                                             NA
## TCGA-BH-A0B1-01              Stage IIB                                            66
## TCGA-B6-A0RG-01              Stage IIB                                            26
##                 CN_Clusters_nature2012 Converted_Stage_nature2012
## TCGA-E9-A1RD-11                     NA                       <NA>
## TCGA-E9-A1RC-01                      3                  Stage IIIC
## TCGA-AC-A3TN-01                     NA                       <NA>
## TCGA-BH-A0B1-01                     NA                No_Conversion
## TCGA-B6-A0RG-01                      2                   Stage IIB
##                 Days_to_Date_of_Last_Contact_nature2012
## TCGA-E9-A1RD-11                                     20
## TCGA-E9-A1RC-01                                   1228
## TCGA-AC-A3TN-01                                     NA
## TCGA-BH-A0B1-01                                   1148
## TCGA-B6-A0RG-01                                   2082
```

```
table(sapply(clinical, class))
```

```
##
## character   integer   numeric
##       142        42         3
```

A more thorough input would carefully map the input columns to their corresponding data types (especially distinguishing between *character* and *factor*) using the `colClasses` argument.

The columns of the clinical matrix are described in the 'BRCA_clinicalFeature' file. There are no headers in the file

```
str(fromJSON(file="BRCA_clinicalFeature.json"))
```

```
## List of 5
##  $ cgDataVersion: num 1
##  $ name         : chr "BRCA_clinicalFeature"
##  $ path         : chr "data_flatten/public/TCGA/BRCA/BRCA_clinicalFeature"
##  $ type         : chr "clinicalFeature"
##  $ version      : chr "2013-10-05"
```

```
features <- read.delim("BRCA_clinicalFeature", header=FALSE, sep="\t",
    col.names=c("Feature", "Key", "Value"), stringsAsFactors=FALSE, )
class(features)
```

```
## [1] "data.frame"
```

```
dim(features)
```

```
## [1] 3799    3
```

```
features[1:5,]
```

```
##                              Feature        Key                             Value
## 1 additional_pharmaceutical_therapy shortTitle                      addtl pharm Tx
## 2 additional_pharmaceutical_therapy  longTitle additional pharmaceutical therapy
## 3 additional_pharmaceutical_therapy  valueType                          category
## 4 additional_pharmaceutical_therapy      state                                NO
## 5 additional_pharmaceutical_therapy      state                              NONO
```

Most but not all columns of the clinical matrix are described in the features file; these columns start with `X_`, but I'm not sure about why they are not documented.

```
contained <- names(clinical) %in% features$Feature
table(contained)
```

```
## contained
## FALSE  TRUE
##    18   169
```

```
head(names(clinical)[!contained], 5)
```

```
## [1] "X_EVENT"                          "X_OS"
## [3] "X_OS_IND"                         "X_PANCAN_CNA_PANCAN_K8"
## [5] "X_PANCAN_Cluster_Cluster_PANCAN"
```

The first column in `clinical` is `AJCC_Stage_nature2012`

```
table(clinical[,1], useNA="always")
```

```
##
##    Stage I   Stage IA   Stage IB   Stage II  Stage IIA  Stage IIB  Stage III Stage IIIA
##         58         64         11         40        249        157         18        102
## Stage IIIB Stage IIIC   Stage IV    Stage X       <NA>
##         23         32         15         10        368
```

(Looking at the data, we see that this column should have been imported as a factor; more on this later). The metadata available about this column includes

```
c1features <- features[features$Feature == names(clinical)[1],]
dim(c1features)
```

```
## [1] 18  3
```

```
c1features$Key
```

```
##  [1] "shortTitle" "longTitle"  "valueType"  "state"      "state"      "state"
##  [7] "state"      "state"      "state"      "state"      "state"      "state"
## [13] "state"      "state"      "state"      "stateOrder" "priority"   "visibility"
```

```
c1features[c(1:3, 17:18),]
```

```
##                    Feature        Key                    Value
## 1386 AJCC_Stage_nature2012 shortTitle                    Stage
## 1387 AJCC_Stage_nature2012  longTitle AJCC Stage (nature 2012)
## 1388 AJCC_Stage_nature2012  valueType                 category
## 1402 AJCC_Stage_nature2012   priority                      153
## 1403 AJCC_Stage_nature2012 visibility                      off
```

```
strsplit(c1features[16, "Value"], ",")
```

```
## [[1]]
##  [1] "Stage I"    "Stage IA"   "Stage IB"   "Stage II"   "Stage IIA"  "Stage IIB"
##  [7] "Stage III"  "Stage IIIA" "Stage IIIB" "Stage IIIC" "Stage IV"   "Stage X"
```

As an **advanced exercise**, implement the following function to coerce input columns from 'BRCA_clinicalMatrix' to the *R* types implied by 'BRCA_clinicalFeature'.

```
importAsFactors <-
    function(data, features)
{
    ## identify data columns with state and stateOrder entries
    ## create factor levels from features
    ## - match 'state' in 'stateOrder' for each factor column
    ## map columns from character to factor, obeying levels
}
```

An implementation is available as

```
importAsFactors <- CGBTutorial:::.import_as_factors
clinical <- importAsFactors(clinical, features)
```

with a preliminary sanity check

```
stopifnot(class(clinical[,"Gender_nature2012"]) == "factor")
table(clinical[,"Gender_nature2012"], useNA="always")
```

```
##
## FEMALE   MALE   <NA>
##    942      9    196
```

As another **advanced exercise**, implement the following function to coordinate the clinical data matrix and annotations into a single object.

```
importAsAnnotatedDataFrame <-
    function(clinical, features,
        featureKeys=c("shortTitle", "longTitle", "valueType"))
{
    ## transform 'features' to Feature x featureKeys matrix
    ## create Biobase::AnnotatedDataFrame with
```

```
    ## - clinical as 'pData'
    ## - feature x featureColumn matrix as 'metadata'
}
```

An implementation is available as

```
importAsAnnotatedDataFrame <- CGBTutorial:::.import_as_AnnotatedDataFrame
clinical <- importAsAnnotatedDataFrame(clinical, features)
```

What are the advantages of this complicated kind of data structure?

1.3 Expression data

The microarray expression data follow a similar pattern – a JSON file describing the data. . .

```
str(fromJSON(file="AgilentG4502A_07_3.json"))
```

```
## List of 30
##  $ domain             : chr "TCGA"
##  $ wrangling_procedure: chr "Level_3 Data download from TCGA DCC, processed at UCSC into cgData reposit
##  $ owner              : chr "TCGA"
##  $ dataProducer       : chr "University of North Carolina TCGA genome characterization center"
##  $ description        : chr "TCGA breast invasive carcinoma (BRCA) gene expression data by agilent arra
##  $ cohort             : chr "TCGA.BRCA.sampleMap"
##  $ tags               : chr "cancer"
##  $ :sampleMap         : chr "TCGA.BRCA.sampleMap"
##  $ sample_type        : chr "tumor"
##  $ label              : chr "BRCA gene expression (AgilentG4502A_07_3)"
##  $ PLATFORM           : chr "AgilentG4502A_07_3"
##  $ version            : chr "2013-10-29"
##  $ gdata_tags         : chr [1:2] "transcription" "mRNA"
##  $ :probeMap          : chr "hugo"
##  $ type               : chr "genomicMatrix"
##  $ colNormalization   : logi TRUE
##  $ anatomical_origin  : chr "Breast"
##  $ longTitle          : chr "TCGA breast invasive carcinoma (BRCA) gene expression (AgilentG4502A_07_3 
##  $ gain               : num 1
##  $ path               : chr "data_flatten/public/TCGA/BRCA/AgilentG4502A_07_3"
##  $ cgDataVersion      : num 1
##  $ :dataSubType       : chr "geneExp"
##  $ wrangler           : chr "cgData TCGAscript AgilentGeneExp processed on 2013-10-29"
##  $ redistribution     : logi TRUE
##  $ name               : chr "TCGA_BRCA_G4502A_07_3"
##  $ shortTitle         : chr "BRCA gene expression (AgilentG4502A_07_3)"
##  $ url                : chr "https://tcga-data.nci.nih.gov/tcgafiles/ftp_auth/distro_ftpusers/anonymous
##  $ primary_disease    : chr "breast invasive carcinoma"
##  $ security           : chr "public"
##  $ groupTitle         : chr "TCGA breast invasive carcinoma"
```

. . . and tab-delimited expression values. We use `check.names=FALSE` to accurately import the sample names, even though this makes it difficult to work with the data at other points *R*. Missing values in this file are indicated by the string `NA`.

```
expression <- read.delim("AgilentG4502A_07_3", header=TRUE, sep="\t",
    row.names=1, na.strings="NA", stringsAsFactors=FALSE,
```

```
    check.names=FALSE)
class(expression)
```

```
## [1] "data.frame"
```

```
dim(expression)
```

```
## [1] 17814   597
```

```
expression[1:5, 1:5]
```

```
##        TCGA-E2-A15G-01 TCGA-BH-A18U-11 TCGA-AR-A1AU-01 TCGA-BH-A1EU-01 TCGA-BH-A0DV-11
## RNF14           0.5676          1.0057          0.9700          0.8448          0.3184
## DUOXA1          0.2300          1.3320          0.8620          0.5425          1.4220
## UBE2Q2         -0.1628          1.3713          0.8053          0.9002         -0.1922
## RNF10           0.3484         -0.3304         -0.3548         -0.1145         -0.5298
## RNF11           0.7830          1.4855          0.8197          0.9415          0.3975
```

The expression values are imported as a *data.frame*, but once the row and column names are removed the remaining values are supposed to be numeric (we check this first) and then coerce the data to a matrix.

```
stopifnot(all(sapply(expression, class) == "numeric"))
expression <- as.matrix(expression)
```

1.4 Coordinating clinical and expression data

Clinical and expression data are tightly coupled; we would not wish to mix up the clinical variables and their corresponding expression. We reduce the chances of doing this by using the *ExpressionSet* class from the *Bioconductor* package *Biobase*. We start by coordinating the column names of the expression data with the (relevant) row names of the clinical data, and then create an *ExpressionSet*.

```
library(Biobase)
stopifnot(all(colnames(expression) %in% rownames(clinical)))
agilent <- ExpressionSet(expression, clinical[colnames(expression), ])
```

The *ExpressionSet* class offers a number of advantages, e.g., convenient display, coordinated subsetting of clinicial and expression data, and interoperability with the main microarray analysis work flows in *Bioconductor*.

```
agilent
```

```
## ExpressionSet (storageMode: lockedEnvironment)
## assayData: 17814 features, 597 samples
##   element names: exprs
## protocolData: none
## phenoData
##   sampleNames: TCGA-E2-A15G-01 TCGA-BH-A18U-11 ... TCGA-A8-A08Z-01 (597 total)
##   varLabels: AJCC_Stage_nature2012
##     Age_at_Initial_Pathologic_Diagnosis_nature2012 ... X_INTEGRATION (187 total)
##   varMetadata: shortTitle longTitle valueType labelDescription
## featureData: none
## experimentData: use 'experimentData(object)'
## Annotation:
```

```
table(agilent$Gender_nature2012, useNA="always")
```

```
##
## FEMALE   MALE   <NA>
##    589      6      2
```

```
agilent[, agilent$Gender_nature2012 %in% "FEMALE"]
```

```
## ExpressionSet (storageMode: lockedEnvironment)
## assayData: 17814 features, 589 samples
##   element names: exprs
## protocolData: none
## phenoData
##   sampleNames: TCGA-E2-A15G-01 TCGA-BH-A18U-11 ... TCGA-A8-A08Z-01 (589 total)
##   varLabels: AJCC_Stage_nature2012
##     Age_at_Initial_Pathologic_Diagnosis_nature2012 ... X_INTEGRATION (187 total)
##   varMetadata: shortTitle longTitle valueType labelDescription
## featureData: none
## experimentData: use 'experimentData(object)'
## Annotation:
```

1.5 Recapitulating the CGB PAM50 survival analysis

For illustration, we'll generate the Kaplan-Meirer survival curves for the PAM50 gene set. The genes in this set come from manually parsing the TCGA figure; a simple text description of this set is not in the original paper [1].

```
pam50genes <- c("UBE2T", "BIRC5", "NUF2", "CDC6", "CCNB1", "TYMS", "MYBL2",
    "CEP55", "MELK", "NDC80", "RRM2", "UBE2C", "CENPF", "PTTG1", "EXO1",
    "ORC6L", "ANLN", "CCNE1", "CDC20", "MKI67", "KIF2C", "ACTR3B", "MYC",
    "EGFR", "KRT5", "PHGDH", "CDH3", "MIA", "KRT17", "FOXC1", "SFRP1", "KRT14",
    "ESR1", "SLC39A6", "BAG1", "MAPT", "PGR", "CXXC5", "MLPH", "BCL2", "MDM2",
    "NAT1", "FOXA1", "BLVRA", "MMP11", "GPR160", "FGFR4", "GRB7", "TMEM45B",
    "ERBB2")
```

We start by restricting our *ExpressionSet* to the PAM50 gene set; there are 6 males and two individuals with unspecified gender.

```
pam50 <- agilent[match(pam50genes, rownames(agilent)),]
dim(pam50)
```

```
## Features  Samples
##       50      597
```

Samples are grouped by PAM50 stratum

```
lvls <- pam50$PAM50_mRNA_nature2012
o_samples <- order(lvls)
o_samples <- rev(o_samples[!is.na(lvls[o_samples])])
```

A heatmap with is shown in Figure 1.5, using color-blind friendly colors[3] and a 'divergent' color scheme as a side bar.

```
library(RColorBrewer)
pal_rows <- brewer.pal(length(levels(lvls)), "Dark2")
pal_heatmap <- rev(brewer.pal(11, "RdBu"))
pdf(file.path(figwd, "pam50heatmap.pdf"))
heatmap(t(exprs(pam50)[,o_samples]), col=pal_heatmap, Rowv=NA, Colv=NA,
    scale="column", margin=c(5, 10), labRow=character(),
    RowSideColors=pal_rows[lvls[o_samples]])
legend("topright", legend=levels(lvls), fill=pal_rows, cex=.9, box.lty=0)
invisible(dev.off())
```

[3] http://colorbrewer2.org/

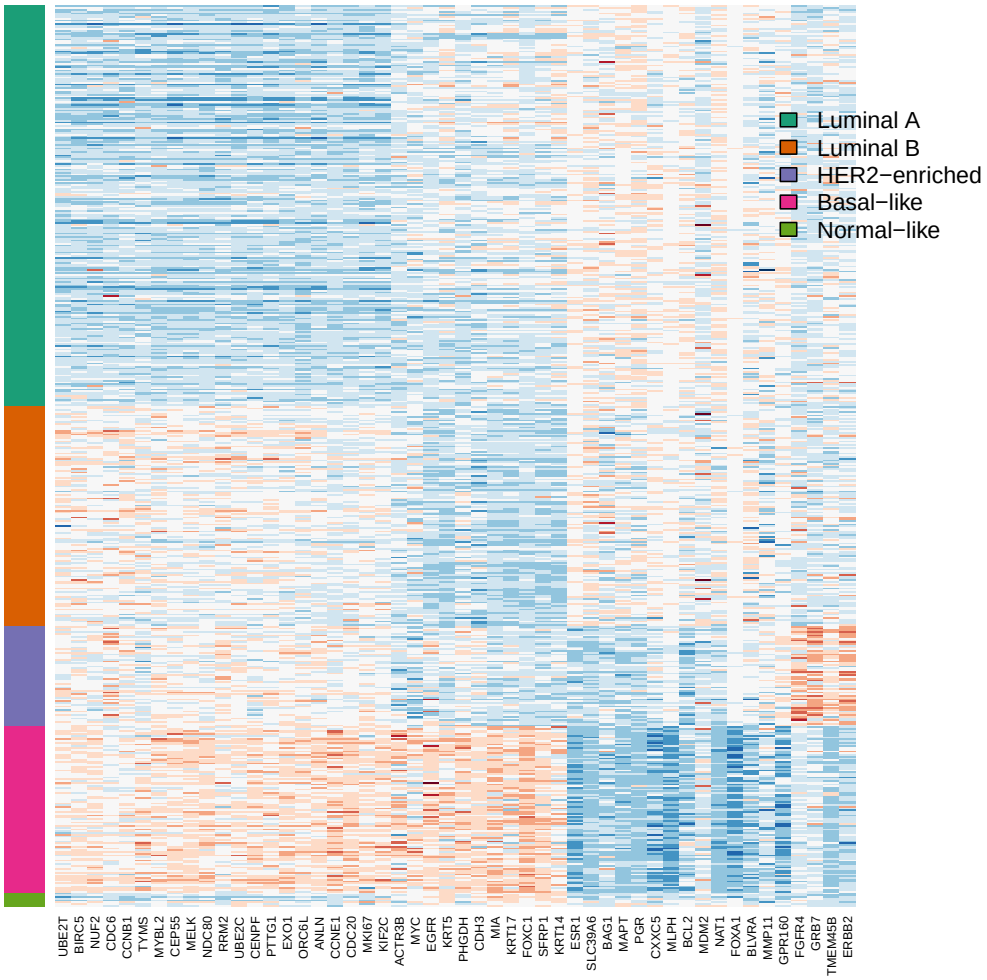

Figure 1: Heatmap of TCGA BRCA AgilentG4502A samples, for 48 of 50 genes in the PAM subset. Genes are hierarchically clustered; samples are grouped by stratum.

An *advanced exercise* is to use the *pamr* package to arrive, from first principles, at a gene set to be used in classifying cancers.

Kaplan-Meier curves are fit using the *survival* package from CRAN. This analysis involves purely phenotypic data, the stratification and survival data. Some detective work is needed to determine the columns used on the CGB web site; after clicking the 'KM' button in the browser, the 'advanced' button suggests the plot is of columns `_EVENT` and `_EVENT overall survival`. Which clincial data columns do these correspond to?

```
grep("_EVENT", colnames(clinical), fixed=TRUE, value=TRUE)
```

```
## [1] "X_EVENT"         "X_TIME_TO_EVENT"
```

From the plot, the expected number of each substratum is

```
c(`Luminal A`=223, `Luminal B`=119, `HER2-enriched`=48, `Basal-like`=93,
  `Normal-like`=8)
```

```
##     Luminal A     Luminal B HER2-enriched    Basal-like   Normal-like
##           223           119            48            93             8
```

In the clinical data as a whole and in the `agilent` subset identified above, the data appear to be consistent with the entire clinical data set

```
with(pData(clinical), {
    c_idx <<- !is.na(X_EVENT) & !is.na(X_TIME_TO_EVENT) &
        !is.na(PAM50_mRNA_nature2012)
    table(PAM50_mRNA_nature2012[c_idx], useNA="always")
})
```

```
##
##     Luminal A     Luminal B HER2-enriched    Basal-like   Normal-like          <NA>
##           223           119            48            93             8             0
```

```
with(pData(agilent), {
    p_idx <<- !is.na(X_EVENT) & !is.na(X_TIME_TO_EVENT) &
        !is.na(PAM50_mRNA_nature2012)
    table(PAM50_mRNA_nature2012[p_idx], useNA="always")
})
```

```
##
##     Luminal A     Luminal B HER2-enriched    Basal-like   Normal-like          <NA>
##           223           119            48            91             8             0
```

The data present in the clincial data as a whole but absent in the microarray subset would appear to be from a duplicate sample, apparently removed from the microarray.

```
c_names <- rownames(clinical)[c_idx]
c_names[!c_names %in% colnames(agilent)]
```

```
## [1] "TCGA-BH-A0HL-01" "TCGA-BH-A0HN-01"
```

We analyze the PAM50 data.

Survival analysis in *R* is done using the *survival* CRAN package; the details of formulating the model are not described here.

```
library(survival)
formula <- Surv(X_TIME_TO_EVENT, event=X_EVENT) ~ PAM50_mRNA_nature2012
fit <- survfit(formula, pData(agilent))
```

A visualization is in Figure 1.5.

```
lvls <- levels(agilent$PAM50_mRNA_nature2012)
pal <- brewer.pal(length(lvls), "Dark2")
pdf(file.path(figwd, "pam50-survival.pdf"), width=8, height=6)
plot(fit, col=pal, lty=1, lwd=2, xlab="X_TIME_TO_EVENT", ylab="Pr(X_EVENT)",
     main="AgilentG4502A subset")
legend("topright", legend=lvls, col=pal, lty=1, lwd=2, pch="+", box.lty=0,
       ncol=2)
invisible(dev.off())
```

2 Working with other data types

2.1 Containers for TCGA data types

The process when working with other data types is similar: download and extract the archive, input the sample and experimental data, create a data structure that coordinates inforamtion, and perform analysis. Table 1 suggests

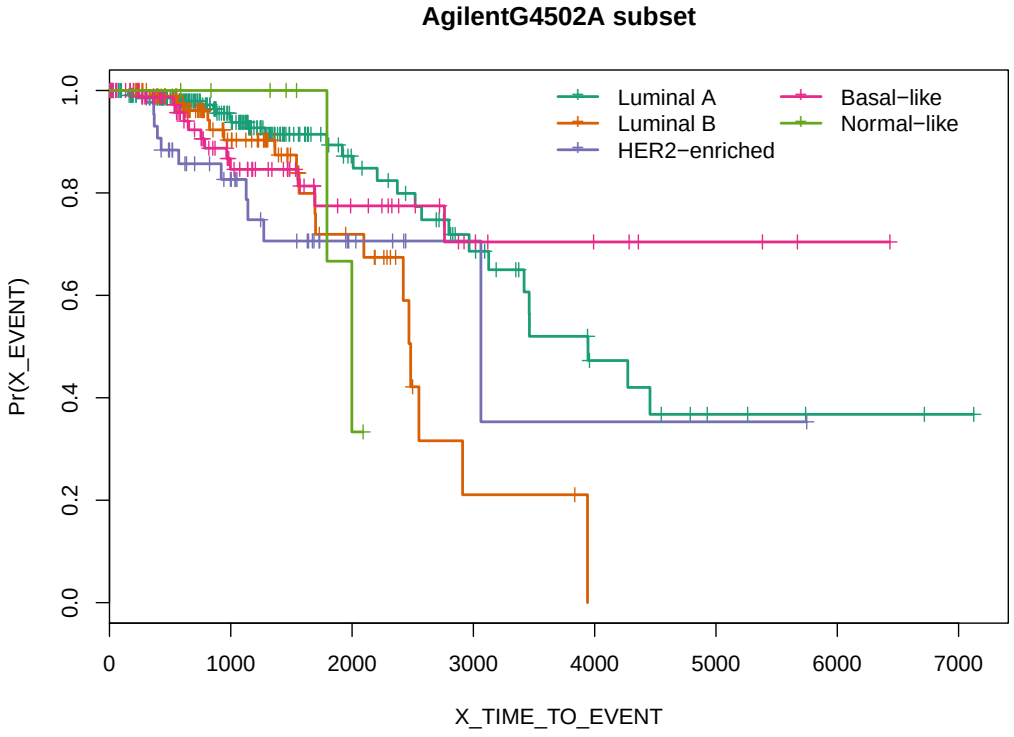

Figure 2: Kaplan-Meier survivorship curves for the AgilentG4502A subset of TCGA BRCA data

Data type	Class	Packages
Expression	*Biobase::ExpressionSet*	*limma* (microarrays); *edgeR*, *DESeq* (RNA-seq)
Copy number	*GenomicRanges::GRanges*	*DNAcopy*
Variants	*VariantAnnotation::VCF*	*VariantAnnotation*
Methylation	*GenomicRanges::SummarizedExperiment*	*minfi*

Table 1: Containers to integrate sample and experiment information

containers that integrate sample and experiment information appropriate for different data types.

2.2 Example: copy number

Here are the steps in common with previous analysis:

```
## extract
untar("TCGA_BRCA_GSNP6noCNV-2013-10-29.tgz")
setwd("TCGA_BRCA_GSNP6noCNV-2013-10-29")
## md5 check
checks <- read.delim("md5.txt", header=FALSE, sep="",
    col.names=c("Sum", "File"), stringsAsFactors=FALSE)
head(checks)
```

```
##                                Sum                            File
## 1 a29b802e527e61c2de63185d39d04262           BRCA_clinicalFeature
## 2 492da3f005988ec9af018f3318753196      BRCA_clinicalFeature.json
```

```
## 3 b0f64544342e938ce3f9ebc8fabc8808            BRCA_clinicalMatrix
## 4 273a5ccda0ddc18d580941db2303c686       BRCA_clinicalMatrix.json
## 5 d7ace38bb7fe67b00f95b74e176dc8b0      SNP6_nocnv_genomicSegment
## 6 43b800ba385a5ee5210ff651cedacab6 SNP6_nocnv_genomicSegment.json
```

```
stopifnot(all(md5sum(checks$File) == checks$Sum))
## clinical data as before, though data is differentx...
## experiment data JSON
str(fromJSON(file="SNP6_nocnv_genomicSegment.json"))
```

```
## List of 27
##  $ :assembly          : chr "hg18"
##  $ :dataSubType       : chr "cna"
##  $ :sampleMap         : chr "TCGA.BRCA.sampleMap"
##  $ PLATFORM           : chr "SNP6"
##  $ anatomical_origin  : chr "Breast"
##  $ cgDataVersion      : num 1
##  $ cohort             : chr "TCGA_BRCA"
##  $ dataProducer       : chr "Broad Institute of MIT and Harvard University cancer genomic characterizat
##  $ description        : chr "TCGA breast invasive carcinoma (BRCA) segmented copy number variation prof
##  $ domain             : chr "TCGA"
##  $ gain               : num 1
##  $ groupTitle         : chr "TCGA breast invasive carcinoma"
##  $ label              : chr "BRCA copy number (delete germline cnv)"
##  $ longTitle          : chr "TCGA breast invasive carcinoma (BRCA) segmented copy number (delete germli
##  $ name               : chr "TCGA_BRCA_GSNP6noCNV_gSeg"
##  $ owner              : chr "TCGA"
##  $ path               : chr "data_flatten/public/TCGA/BRCA/SNP6_nocnv_genomicSegment"
##  $ primary_disease    : chr "breast invasive carcinoma"
##  $ redistribution     : logi TRUE
##  $ sample_type        : chr "tumor"
##  $ shortTitle         : chr "Copy Number (delete germline cnv)"
##  $ tags               : chr "cancer"
##  $ type               : chr "genomicSegment"
##  $ url                : chr "https://tcga-data.nci.nih.gov/tcgafiles/ftp_auth/distro_ftpusers/anonymous
##  $ version            : chr "2013-10-05"
##  $ wrangler           : chr "cgData TCGAscript SNP6 processed on 2013-10-05"
##  $ wrangling_procedure: chr "Level_3 data download from TCGA DCC, processed at UCSC into cgData reposit
```

The clinical data is the same, so no need to re-process.

The copy number experimental data requires distinct processing. Here are the first few lines of data

```
readLines("SNP6_nocnv_genomicSegment", 3)
```

```
## [1] "b5370ac3-c55f-464c-bb98-6baa05d8b6b8\tchr1\t3208470\t10350677\t.\t-0.0695"
## [2] "b5370ac3-c55f-464c-bb98-6baa05d8b6b8\tchr1\t10359013\t10370662\t.\t-0.7602"
## [3] "b5370ac3-c55f-464c-bb98-6baa05d8b6b8\tchr1\t10372619\t10378734\t.\t0.011"
```

This is tab-delimited data. The first column is an identifier, the next columns represent the chromosome ('seqname' in *Bioconductor* parlance), start, end, and strand (. indicates that strand is not relevant), and the final column is a score representing the direction and magnitude of the copy number change.

```
snp <- read.delim("SNP6_nocnv_genomicSegment", header=FALSE, sep="\t",
    stringsAsFactors=FALSE,
    col.names=c("id", "seqname", "start", "end", "strand", "score"))
class(snp)
```

```
## [1] "data.frame"
```

```
dim(snp)
```

```
## [1] 239209      6
```

```
head(snp)
```

```
##                                     id seqname    start      end strand   score
## 1 b5370ac3-c55f-464c-bb98-6baa05d8b6b8    chr1  3208470 10350677      . -0.0695
## 2 b5370ac3-c55f-464c-bb98-6baa05d8b6b8    chr1 10359013 10370662      . -0.7602
## 3 b5370ac3-c55f-464c-bb98-6baa05d8b6b8    chr1 10372619 10378734      .  0.0110
## 4 b5370ac3-c55f-464c-bb98-6baa05d8b6b8    chr1 10380965 10952851      . -0.5922
## 5 b5370ac3-c55f-464c-bb98-6baa05d8b6b8    chr1 10953372 30871010      . -0.0722
## 6 b5370ac3-c55f-464c-bb98-6baa05d8b6b8    chr1 30873255 30873905      . -1.5216
```

fixme: 0-based or 1-based coordinates? Range-based data such as this is represented effectively in *Bioconductor* as *GRanges* objects from the *GenomicRanges* package.

```
library(GenomicRanges)
grsnp <- with(snp, {
    strand <- sub(".", "*", strand)
    GRanges(seqname, IRanges(start, end), strand, id=id, score=score)
})
head(grsnp, 3)
```

```
## GRanges with 3 ranges and 2 metadata columns:
##       seqnames              ranges strand |                                     id
##          <Rle>           <IRanges>  <Rle> |                            <character>
##   [1]     chr1 [ 3208470, 10350677]      * | b5370ac3-c55f-464c-bb98-6baa05d8b6b8
##   [2]     chr1 [10359013, 10370662]      * | b5370ac3-c55f-464c-bb98-6baa05d8b6b8
##   [3]     chr1 [10372619, 10378734]      * | b5370ac3-c55f-464c-bb98-6baa05d8b6b8
##           score
##       <numeric>
##   [1]   -0.0695
##   [2]   -0.7602
##   [3]    0.0110
##   ---
##   seqlengths:
##     chr1 chr10 chr11 chr12 chr13 chr14 chr15 ...  chr4  chr5  chr6  chr7  chr8  chr9  chrX
##       NA    NA    NA    NA    NA    NA    NA ...    NA    NA    NA    NA    NA    NA    NA
```

(The `seqinfo` at the end of *GRanges* object can be used to specify chromosome lengths, and when present is used as a very valuable sanity check when performing operations on different ranges).

2.3 Coverage and other range-based operations

GRanges objects enable many very useful operations. For instance, one can summarize overall copy number across all samples with

```
cvg <- coverage(grsnp, weight=grsnp$score) / length(unique(grsnp$id))
names(cvg)
```

```
##  [1] "chr1"  "chr10" "chr11" "chr12" "chr13" "chr14" "chr15" "chr16" "chr17" "chr18"
## [11] "chr19" "chr2"  "chr20" "chr21" "chr22" "chr3"  "chr4"  "chr5"  "chr6"  "chr7"
## [21] "chr8"  "chr9"  "chrX"
```

```
cvg$chr1
```

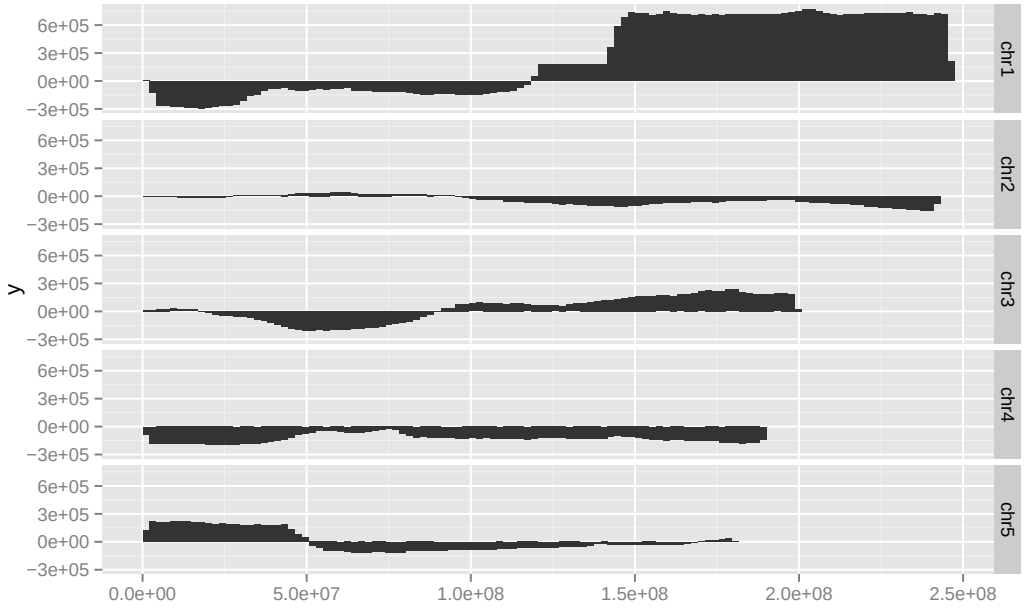

Figure 3: Average coverage from TCGA BRCA SNP6 arrays across all samples

```
## numeric-Rle of length 245880329 with 27129 runs
##   Lengths:            3208469               3012 ...               1275
##   Values :                  0  -0.11558411122145 ...  0.336825620655412
```

The result is a list, one element for each chromosome, of *Rle* (run-length encoding) objects. An *Rle* is a compact way of storing genome scale data. The snippet above shows that there are 27129 'runs', the first consists of 3208469 nucleotides where over all samples the average copy number is unchanged from diploid, 3012 nucleotides where average copy number is -0.1156

GRanges is used extensively in diverse *Bioconductor* packages. For instance, the flexible *ggbio* package provides plotting facilities following Wickham's popular *ggbio2* paradigm, tailored to genomic data, e.g., Figure 3.

```
library(ggbio)
pdf(file.path(figwd, "cvgplot.pdf"), width=8, height=5)
suppressWarnings({
    print(autoplot(cvg[paste0("chr", 1:5)], nbin=100))
})

## Default use binwidth:  range/100

invisible(dev.off())
```

Another great visualization package is *Gviz*, which emaphasizes elegant presentation of data analogous to tracks in a genome browser, but more elegantly and flexibly presented.

It is straight-forward, using functions such as `findOverlaps` or `subsetByOverlaps`, to tally or select overlaps between regions, e.g., of average copy number elevation and enhanced gene expression.

3 Summary

1. The CGB can be used to navigate download level III TCGA data archives.

2. TCGA data can be input into *R* using standard commands, but it pays to got the 'extra mile' and create integrated data objects.
3. Once in *R*, there are diverse analysis and visualization opportunities in base, CRAN, and *Bioconductor* packages; the *GenomicRanges* package and allies is the starting point for any range-based, genome scale data analysis task.

***Bioconductor* resources** There are a large number of *Bioconductor* packages for microarray, sequence, flow cytometry, and other high-throughput genomic analyses. Check out the complete list[4] or for particular 'views' such as HighThroughputSequecing[5]. Each package has a 'landing page' (e.g., for *GenomicRanges*) that include links to vignettes (for overall orientation and use) and manuals (for documentation of individual functions). Material from previous courses[6] can provide valuable orientation. Short forthcoming courses are available through the Computation Biology shared resource; the next two-day course offered by the *Bioconductor* group will be in February. There are very knowledgeable experts here at the Hutch, feel free to contact me if you'd like *R* / *Bioconductor* help.

References

[1] J. S. Parker, M. Mullins, M. C. Cheang, S. Leung, D. Voduc, T. Vickery, S. Davies, C. Fauron, X. He, Z. Hu, J. F. Quackenbush, I. J. Stijleman, J. Palazzo, J. S. Marron, A. B. Nobel, E. Mardis, T. O. Nielsen, M. J. Ellis, C. M. Perou, and P. S. Bernard. Supervised risk predictor of breast cancer based on intrinsic subtypes. *J. Clin. Oncol.*, 27(8):1160–1167, Mar 2009.

[4] http://bioconductor.org/packages/release/BiocViews.html
[5] http://bioconductor.org/packages/release/BiocViews.html#___HighThroughputSequencing
[6] http://bioconductor.org/help/course-materials/