The GDS Package: Vignette

Marijke Van Moerbeke & Geert-Jan Bex

September 4, 2015

Contents

1 Introduction

This vignette describes how to use the GDS package starting from raw microarray data contained in .CEL files. For the computation of the GDS model we do recommend the use of a supercomputer as it requires memory and time for the data to be processed. All other methods can be conducted in reasonable time on an ordinary laptop but those who have a high-performance computing (HPC) service freely available have the advantage to use this also for the regular analysis. We will demonstrate an example step by step after giving a short introduction into the used methodology. The package accompagnies the paper by Van Moerbeke et al. (2016) which introduced the GDS model.

2 Data

The example data is the tissue data publicly available on the Affymetrix website. The data set contains 33 arrays from 11 tissues with three replicates per tissue and was generated with the GeneChip® Human Exon 1.0 ST array. The array is a whole genome array and contains only perfect matching (PM) probes, with a small number of generic mismatching probes for the purposes of background correction. Four perfect match probes were designed for each probe set and a probe set can roughly be seen as an exon. There are no probes which span exon-exon junctions (Affymetrix, 2005b). The data comes from the "HuEx-1_0-st-v2" chip type and a custom .cdf file for the annotation is available from the website of the `aroma.affymetrix` (Bengtsson et al., 2008) package.

3 Methodology

The Genome-wide Differential Splicing (GDS) model is situated in a mixed model framework. Alternative splicing detection can now be formulated as variance decomposition in a random effects model with a random intercept per exon taking the gene expression into account. The following can be said. The *between array variability* of an alternatively spliced exon would be higher than the *within array variability* among the exons of the same transcript cluster. A non-alternatively spliced exon would have a between array variability that is at most the within array variability across all exons of the same transcript cluster. These hypotheses can be formulated as a two-stage mixed effect model for alternative splicing detection.

The model is fitted on the observed PM probe intensities levels as:

$$log2(PM_{ijk}) = p_j + c_i + b_{ik} + \epsilon_{ijk}, \tag{1}$$

Here, parameter p_j denotes the effect of probe j, c_i is the overall gene effect of array i and b_{ik} is an exon specific deviation from this overall effect. The background noise $\epsilon_{ijk(j)} \sim N(0, \sigma^2)$ captures the within array variability with σ^2. Differential expression between the arrays is expected to be negated by incorporating the gene effect parameter c_i. By consequence, the exon specific parameters show the deviation of a particular exon from its corresponding gene level. If there is only a small deviation, it can be said that the exon is present in the sample. A large deviation however shows that the exon likely absent. Note that the exon specific deviations are random effects assumed to be normally distributed, $b_{ik} \sim N(\mathbf{0}, \mathbf{D})$. The $K \times K$ covarianc matrix D contains the exon specific signals with τ_k^2 on its diagonal and K the number of exons in a transcript cluster. We term this model the Genome-wide Differential Splicing (GDS) mixed model.

The advantage of a mixed model formulation for alternative splicing detection is the existence of a standard score to quantify the trade-off between signal and noise. We term this score an "exon score" but it is also known as the intra-cluster correlation (ICC) of linear mixed models. The score is defined as the ratio of the exon specific signal to the noise across all exons of a transcript cluster. For a probeset k of a transcript cluster this is formulated as:

$$\rho_k = \tau_k^2/(\sigma^2 + \tau_k^2),$$

where σ^2 is the same for all probesets belonging to the same transcript cluster. It intuitively follows from the definition of the exon score that an equity threshold between exon specific signal

and transcript level noise is 0.5. This could be used as a threshold for identifying an alternatively spliced probeset among the probesets of a transcript cluster. A value of $\rho_k > 0.5$ puts more weight in favor of probeset k to be alternatively spliced. Note that the threshold for the exon score could be adjusted to the relative amount of signal in the data. Given that probeset k has been identified to have substantial between array variability, we propose to use the estimated random effects b_{ik} as an "array score" for identifying arrays in which the alternatively spliced exon is expressed. Tissues with an enrichment of probeset k are expected to have array scores greater than zero as they resisted shrinkage towards the overall gene level effects.

The parameters of the proposed mixed effects model are estimated within the Bayesian framework with vague proper priors since the full conditional posterior distributions for the parameters of interest are known. More information can be found in Van Moerbeke et al. (2016).

4 Data Processing

The processing of the data starts from the raw .CEL files with functions of the `aroma.affymetrix` (Bengtsson et al., 2008) package. Whether you are working on a HPC or on your laptop, to be able to use this package, you have to make sure to have the correct folder structure. In the current working directory two folders have to be set up. The first folder is a rawData folder. This folder contains another folder with the name of your data set. In our case this will be "TissueData". This "TissueData" folder should contain a last folder with as name the chipType used to produce the data (eg "HuEx-1_0-st-v2") and herein the .CEL files should be placed. The second folder is the "annotationData" folder. Herein a folder with name "chipTypes" should be made which contains folders for each chip type with the respective names. In the folder of each chip type the corresponding .cdf file should be saved. Figure 1 shows the structure of the directories with as working directory a folder named "GDS". An R script can now be created in the current working

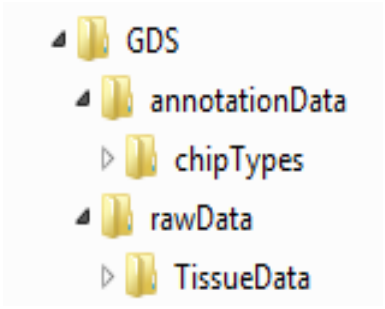

Figure 1: *Folder Structure for the* `aroma.affymetrix` *package*

directory such that the functions have acces to the created folders. The function used to perform the data processing is `DataProcessing` and is a wrapper of several functions of the `aroma.affymetrix` package which is imported by the `GDS` package. To obtain the data to perform the GDS model on, the raw .CEL files are background corrected with the rma background correction and normalization is performed with the quantile normalization. The data is returned as a data frame with one line per probe. There are thus multiple lines per probeset (one for each probe of the probeset) and even more per gene as often more than one exon belongs to the same gene. The first colum contains the gene IDs and the second column the exon IDs. All other columns contain the sample values of the probes. If requested, also a gene and exon level summarization is performed on the data with the rma summarization method. Further, the option is provided to perform the FIRMA model (Purdom et al., 2008) on the data as well. For the tissue data data the function specified in a regular R script is as follows:

```
> DataProcessing(chipType="HuEx-1_0-st-v2",tags="coreR3,A20071112,EP",
+                Name="TissueData",ExonSummarization=TRUE,GeneSummarization=TRUE,
+                FIRMA=TRUE,location="TissueData",verbose=TRUE)
```

The tags parameter refers to a tag that was added to the .cdf file. Make sure to have created a folder named TissueData in your working directory for the results to be saved. This is the loctaion

in the function above. The function as formulated above will create four data frames. One with the backgouncorrected and normalized probe intensities, one with the gene level summarized values, one with the exon level summarized values and a last one with the values of the FIRMA model. If one has a HPC available, we can call upon the previous function via the DataProcessing.R and DataProcessing.pbs scripts available in the doc folder. Be carefull to fill in the requirements for your specific project. After completing the .pbs file, a qsub command on this batch file should get the function running.

If you do not have access to a HPC service, the next processing steps can be skipped and the data is a proper format for the GDS model implemented in the `GDSFunction` function. If you do have access to an HPC service and you wish to use it, some more processing of the data is necessary. The strength of using the HPC server is that calculations can be parallelized. In our case where we run the GDS model gene-by-gene, the server needs a file in which every line correspond to one gene and contains all the information of that gene. This way the cluster can read several lines, etc several genes simultaneously and distribute them over multiple cores who perform the calculations independently. First, a pivot transformation will be conducted to convert the data into a file with one line per gene. All information concerning one gene is thus gathered into this line. The first column of the returned file contains the gene ID, the second column contains the exon IDs of all the exons of that gene. The third colum indicates the number of probes per exon, the fourth contains the values of those probes per sample and the last column contains the sample names.

```
> load("TissueData/TissueData.RData")
> PivotTransformation(Data=TissueData,GeneID=TissueData$GeneID,
+                     ExonID=TissueData$ExonID,savecsv=TRUE,
+                     Name="TissueData_Pivot",location="TissueData")
```

This function is already included in the DataProcessing.R file in the doc folder and thus if this file is used on the HPC service, it is not needed to be called upon separately. Secondly, a final file is created with the begin positions and lengths of the lines of the previous file. The function for this indexing of the lines is not an `R` function but a `python` function. The doc folder contains the `Line_Indexer.py` function and its corresponding .pbs file. The last line is where the function is called.

```
python Line_Indexer.py --output_file TissueData_lineindex.csv TissueData_Pivot.csv
```

After output_file the first name is the name to be given to the transformed file. The second file is the file to be transformed. Make sure that the function is run in the same folder where the file to be transformed is saved. The HPC server will now work from these begin positions, read a line from the first file and process the corresponding gene with the GDS model. The second file is not necessary for the calculations of the HPC server but it makes life easier for the server as it does not need to read in all the information on all the genes. After this final function, the data is ready to be used with the GDS model on the cluster.

5 The GDS Model

The data is now in an adequate format for the GDS model. If you do not have acces to an HPC service, the `GDSFunction` function can be performed on the data frame returned by the `DataProcessing` function. The `GDSFunction` function consists of two subfunctions. The first is the `inigds` which filters out non-informative probesets with the I/NI calls model (Kasim et al., 2010). The option is available whether the filtering should be performed or not. The second function is the `gdsmodel_intern` and performs the actual GDS model, possibly on the output of `inigds`. A list with one element per gene is returned. Per gene, there is list with three elements if `inigds` is performed. Otherwise, two elements are available per gene. The optional element per gene is a data frame with the exons and a logical value indicating whether the exon is informative or not with `TRUE` or `FALSE` respectively. The other two elements contain the exon scores and the array scores of the exons. The function is:

```
> load("TissueData/TissueData.RData")
> GDS_Output=GDSFunction(geneData=TissueData,nsim=5000,geneID=TissueData$GeneID,
+                        exonID=TissueData$ExonID,informativeCalls=TRUE,alpha=0.5)
```

The non-cluster version of the GDS model is recommended to be used only on a small data set or a subset of a larger one. The `GDSFunction_ClusterVersion` is an adaption of the GDS model to be used on a HPC server. The cluster will parallelize the computations. This implies that several genes will be processed simultaneously. The process starts by reading a line of the file created by the python function `Line_Indexer`. This contains a begin position and length of that line of the file made by `PivotTransfornmation`. Next, the corresponing line will be looked for and read into the script. This line enholds all the information of one gene and with some processing, a small data frame of this gene is produced. This subset is then processed by the `GDSFunction_ClusterVersion` function. For each processed gene a .RData file will be saved with the output of the GDS model. We will combine these later. The entire procedure is coded in the GDSFunction_ClusterVersion.R file in the doc folder of this pacakge. A corresponding .pbs file is also available. By submitting the file of the `Line_Indexer` function as the data and the GDSFunction_ClusterVersion.pbs file as a batch file to a worker framework on the HPC server the function should be executed. The computation time depends from data set to data set. For the tissue data, 24 hours should be adequate. The long computational time is due a just a few genes that have a large number of exons. After 24 hours, five gene were not returned and therefore left out of the analysis.

Finally, the indiviudal gene outcomes of the `GDSFunction_ClusterVersion` should be binded together into one larger data set. This is done by the `CreateOutput` function. It is necessary to provide a file of the gene IDs of which the outcomes were produced. The doc folder contains this file for the tissue data. The function will read the gene ID, look for the corresponding file of the GDS model and add it to a list. Eventually it will return a list with one element per gene. The function can be run on the HPC service with the CreateOutput.R and CreateOutput.pbs files in the doc folder. A qsub command on the .pbs file gets the function started.

6 Analysis

If you are interested in the results of a particular gene and/or exon the `Search` function allows you to get that specific data out of the output. Given exon IDs, it is capable of retrieving the corresponding gene IDs as well in case the gene ID is not known beforehand. The function can be run on the GDS ouput but also on more analyzed data frames as for example after performing a test between the samples.

```
> exonID <- c(3252129,3597384,3333718,3735208,2598321,3338589,2605391,2605390,
+             2605386,3025632,2375766,3569827,3569830,2334499,3972987,2516011,
+             2989068,3422189)
> load("GDS_Output.RData")
> Results=Search(WhatToLookFor=data.frame(ExonID=exonID),Data=GDS_Output,
+                AggregateResults=FALSE,NotFound=NULL)
```

In order to identify exons that are alternatively spliced, their exon scores and array scores should be investigated. The array scores can be tested between groups of interest of the samples. If the data consists of paired samples, the mean paired differences can be investigated. We have written a function `ExonTesting` that performs a simple t-test on the array scores. The p-values are adjusted for multiplicty. The function has the option to already filter out probesets that do not pass the exon score threshold and only consider those that do. If also a significance level is specified, non-significant p-values will be left out of the results. A data frame with one line per exon is returned. The columns contain the gene ID, the exon ID, the test statistic, a p-value and an adjusted p-value. If the groups are paired also the mean paired difference is given.

```
> load("TissueData/GDS_Output.RData")
> groupHMPT=c(7,8,9,16,17,18,22,23,24,31,32,33)
> groupOthers=c(1,2,3,4,5,6,10,11,12,13,14,15,19,20,21,25,26,27,28,29,30)
> groups=list(group1=groupHMPT,group2=groupOthers)
> TissueData_ExonTesting=ExonTesting(Data=GDS_Output,Exonthreshold=NULL,groups=groups,
+                                    paired=FALSE,significancelevel=NULL)
```

The `ExonTesting` function was designed to be flexible. Filtering on exon scores is not necessary and neither is filtering on the significance level. Also, ExonTesting.R and ExonTesting.pbs files

are available in the doc folder to be run with an qsub command on the cluster.

The `ExonTesting` function is convenient when you are interested in the socres of all probesets. If the interest lies in the those that are identified as alternatively spliced, we recommend to use the `ASExons` function. If the input of this function is the outcome of the GDS model, the `ExonTesting` function will be performed with a specified threshold on the exon scores and a given significance level. A data frame only containing the probesets that pass the requiremenents on both exon score and significance level will be returned. Further, the data frame is ordered from the highest exon score to the lowest. Otherwise, if the `ExontTesting` function was already performed, the returned data frame can be filtered according to specificed thresholds for the exon score and the significance level with this function. An adjustment for multiplicty after filtering on exon score will be performed as well.

```
> TissueData_ExonTesting_Sign1=ASExons(Data=GDS_Output,Exonthreshold=0.5,groups=groups,
+                                      paired=FALSE,significancelevel=0.05)
> TissueData_ExonTesting_Sign2=ASExons(Data=TissueData_ExonTesting,Exonthreshold=0.5,
+                                      groups=groups,paired=FALSE,significancelevel=0.05)
```

Corresponding files for use on the HPC server are also here available in the doc folder. During the data processing, we have also obtained the results of the FIRMA model. These can be analyzed further with the `FIRMAScores` function. In this function we calculate an all sample FIRMA score as indicated in the FIRMA paper (Purdom et al., 2008). This is the mean paired difference for paired samples. For more than two groups and in a situation were some of these are tested against the other, the all sammple score is the maximum of the minimum of the FIRMA scores of those that belong to one of the groups of interest. Further, a t-test is performed on the FIRMA scores of the groups of interest and the returned p-value is adjusted for multiplicity. If a significance level is specified, only significant probesets are returned. The application of the FIRMA model in the `aroma.affymetrix` package does not allow for filtering. Here, we can specify a character vector of informative exons that remain in the GDS output after filtering with the I/NI calls model or even only those that are have adequate exon scores. The p-value will be adjusted accordingly. A corresponding .R and .pbs script can be found in the doc folder for use with the qsub command.

```
> load("TissueData/FIRMA_Output.RData")
> load("TissueData/GDS_Output.RData")
> groupHMPT=c(7,8,9,16,17,18,22,23,24,31,32,33)
> groupOthers=c(1,2,3,4,5,6,10,11,12,13,14,15,19,20,21,25,26,27,28,29,30)
> groups=list(group1=groupHMPT,group2=groupOthers)
> exons=TissueData_ExonTesting[which(TissueData_ExonTesting$X50.>0.5),]
> TissueData_FIRMA=FIRMAScores(Data=FIRMA_Output,InformativeExons=exons,
+                              groups=groups,paired=FALSE,significancelevel=0.05)
```

If an exon is of particular interest, plots can be made to visualize the provided data to investigate the situation further. The `PlotFunction` makes three plots for one probeset. The first plot illustrates the exon and gene level summarized values together with the observed probe intensities for the probeset of interest. The second plots the density of the array scores. The last plot contains a heatmap of the array scores and FIRMA scores of all the probesets of the gene.

```
> #Top gene 2736322 and   Top exon 2736397
> load("TissueData/TissueData.rda")
> load("TissueData/TissueData_ExonLevelSummarized.RData")
> load("TissueData/TissueData_GeneLevelSummarized.RData")
> load("TissueData/FIRMA_Output.RData")
> load("TissueData/GDS_Output.RData")
> groupHMPT=c(7,8,9,16,17,18,22,23,24,31,32,33)
> groupOthers=c(1,2,3,4,5,6,10,11,12,13,14,15,19,20,21,25,26,27,28,29,30)
> groups=list(group1=groupHMPT,group2=groupOthers)
> PlotFunction(GeneID="2736322",ExonID="2736397",Data=TissueData,
+              GDS_Output=GDS_Output,GeneLevelData=TissueData_GeneLevelSummarized,
+              ExonLevelData=TissueData_ExonLevelSummarized,FIRMA_Data=FIRMA_Output,
+              groups=groups,plottype="new",
```

```
+                location="TissueData/Gene2736322_Exon2736397")
>
```

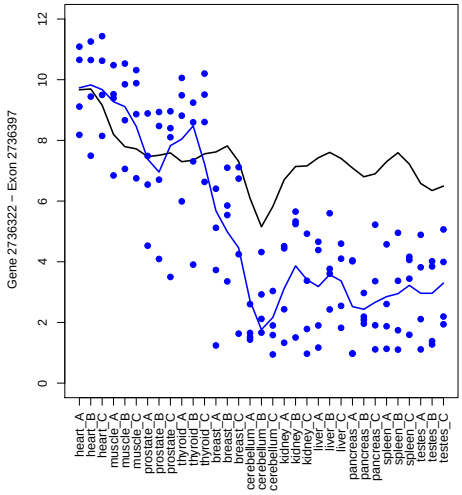

Figure 2: *Gene 2736322 with probeset 2736397*

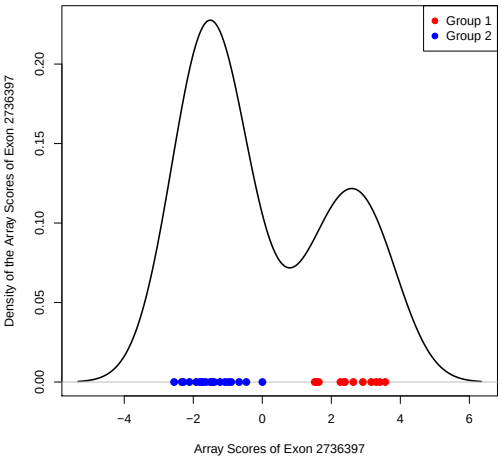

Figure 3: *Density plot for the Array Scores of probeset 2736397*

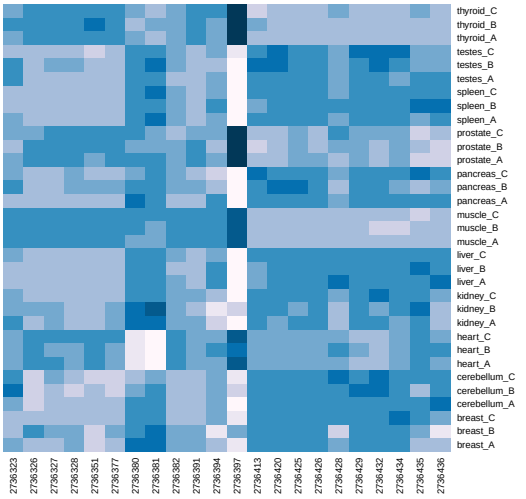

Figure 4: *The array scores of gene 2736322*

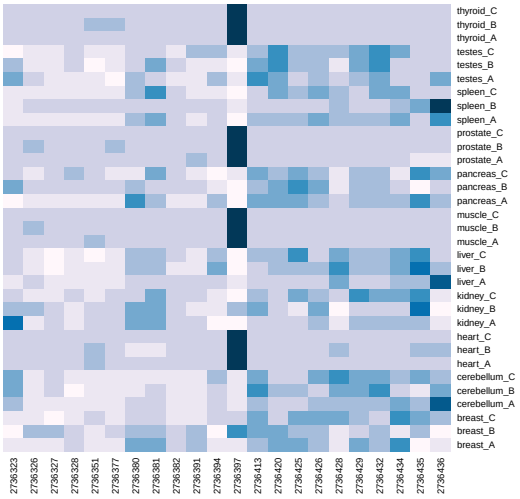

Figure 5: *The array score of gene 2736322*

Next to the GDS and FIRMA model we have also included the SI index of Affymetrix into our package (Clark et al., 2007 and Affymetrix, 2005a) in the `SpliceIndex` function. The function requires the gene and exon level summarized data. The first step is to normalize the exon data by taking the ratio with the gene level data. These values are referred to as the splice indices. If only two groups are specified, the ratio of their splice indices is taken as a measure for alternative splicing. The more the ratio deviates from zero, the more there is an indication of alternative splicing. A t-test is conducted on the splice indices of the two groups to test their difference. If more than two groups are specified, an ANOVA model is fitted on the splice indices to discover with an F-test whether there is a difference between the groups somewhere. If a vector of informative exons is given to the function, only these are considered for the analysis. Finally, the p-values are adjusted for multiplicity and if a significance level is specified only the significant p-valuesare kept in the data frame. The function can be performed on the HPC server with the corresponding SplicIndex.R and SpliceIndex.pbs file in the doc folder.

```
> load("TissueData/TissueData_ExonLevelSummarized.RData")
> load("TissueData/TissueData_GeneLevelSummarized.RData")
> groupHMPT=c(7,8,9,16,17,18,22,23,24,31,32,33)
> groupOthers=c(1,2,3,4,5,6,10,11,12,13,14,15,19,20,21,25,26,27,28,29,30)
> groups=list(group1=groupHMPT,group2=groupOthers)
> SI_Output=SpliceIndex(GeneData=TissueData_ExonLevelSummarized,
+                       ExonData=TissueData_GeneLevelSummarized,
+                       InformativeExons=NULL,groups=groups,
+                       paired=FALSE,significancelevel=NULL)
```

The addition of the SI method was for the completion of the package. We would like to compare the GDS, FIRMA and SI method more in the future. The test-statistics returned by the `SpliceIndex` function did not deviate from the results obtained by the MiDAS algorithm implemented in the Affymetrix command tool program in our experience. We have implemented the function in `R` for flexible and easy accesable use.

References

Affymetrix (2005a), "Alternative transcript analysis methods for exon arrays," *Affymetrix Whitepaper*.

— (2005b), "GeneChip Exon Array Design," *Affymetrix Technical Note*.

Bengtsson, H., Irizarry, R., Carvalho, B., and Speed, T. (2008), "Estimation and assessment of raw copy numbers at the single locus level." *Bioinformatics*, 24, 759–767.

Clark, T., Schweitzer, A., Chen, T., Staples, M., Lu, G., Wang, H., Williams, A., and Blume, J. (2007), "Discovery of tissue-specific exons using comprehensive human exon microarrays," *Genome Biology*, 8, R64.

Kasim, A., Lin, D., Van Sanden, S., Clevert, D., Bijnens, L., Goehlmann, H. W. H., Amaratunga, D., Hochreiter, S., Shkedy, Z., and Talloen, W. (2010), "Informative or noninformative calls for gene expression: a latent variable approach." *Stat Appl Genet Mol Biol*, 9, Article4.

Purdom, E., Simpson, K. M., Robinson, M. D., Conboy, J. G., Lapuk, A. V., and Speed, T. P. (2008), "FIRMA: a method for detection of alternative splicing from exon array data," *Bioinformatics*, 1707–1714.

Van Moerbeke, M.and Kasim, A., Shkedy, Z., and Talloen, W. (2016), "GDS: Genome-wide Differential Splicing Using Exon and HTA Arrays," *BMC Genomics*.

7 Software used

- R version 3.2.2 Patched (2015-08-27 r69201), `x86_64-pc-linux-gnu`
- Locale: `LC_CTYPE=en_US.UTF-8`, `LC_NUMERIC=C`, `LC_TIME=en_US.UTF-8`, `LC_COLLATE=C`, `LC_MONETARY=en_US.UTF-8`, `LC_MESSAGES=en_US.UTF-8`, `LC_PAPER=en_US.UTF-8`, `LC_NAME=C`, `LC_ADDRESS=C`, `LC_TELEPHONE=C`, `LC_MEASUREMENT=en_US.UTF-8`, `LC_IDENTIFICATION=C`
- Base packages: base, datasets, grDevices, graphics, methods, stats, utils
- Loaded via a namespace (and not attached): tools 3.2.2