RDS Assignment Markdown

B198096

2022-11-22

Generating pretty heat maps from gene expression data

Setup

This pipeline is designed to generate a pretty heat map of a subset of Gene expression data from a data file of expression values, using associated Annotation files to label the data for useful interpretation.

Note that before the pipeline begins processing the data I need to import The data into the working directory. I performed this in the terminal command line at the beginning of the project. The command for this is below

Retrieving the files:
cp /shared_files/RDS_assignment_files/s2249132_files.zip .
unzip s2249132_files.zip

All code, both with and without comments, can be accessed here:
https://github.com/B198096-2022/RDS_Assignment

To copy the code to your working directory from a terminal enter:
git clone https://github.com/B198096-2022/RDS_Assignment

##Formatting The Data

The pipeline reads in four files:

data_all.csv contains the actual data
gene_annotation.csv contains the gene names and gene types for all genes analysed
sample_annotation.csv contains information on the sample groupings
genelist_7.txt is a personalized list of the genes that this pipeline will analyse

The files are first read in as tables Then the gene and sample annotations are added to the data table to generate a large, annotated data frame, after which the list of personalized genes is used to extract the desired rows from the data frame to create a smaller data frame which is used for the heat maps. Lastly, the data is log transformed. An error check is done beforehand to ensure that there are no zeros to cause errors for log transformation as described below.

##Checking and Filtering Data

The pipeline contains three functions for checking the data values and correcting any undesired values in the data

zero_test(df, find) is a function that takes a data frame as its first argument It reads through every data point in the data frame and generates a list of every position where there is a zero, printing out this list and returning the list as the output of the function It can also be used to find any specified number or value within the data frame, searching for the input argument 'find'. The default find is 0 but can be passed as any integer or string.

mimimum_test(df, min) is a similar function that takes a data frame as its first argument and reads through every data point in the data frame to check if it is equal to or less than the passed argument "min", which has a default of 1. It generates a list of every position in the data frame that meets this criteria and both prints and returns this list.

change_values <- function(df, find, change) takes a data frame as its first argument, then will find any data points that match the argument 'find' and change them to the value of the argument 'change', which have a default of 0 and 1, respectively. It then returns the corrected data frame. So this function corrects any zero values to ones by default to avoid log transformation errors. But it can be re-used for other error checking.

The pipeline checks whether there are any zeroes in the data and only calls these functions if there are zeros. The data used for this pipeline did not possess any zeros, so the functions were not needed.

##Annotating The Data

The heat maps are annotated to label the gene types and sample treatment groups. This is accomplished by generating two annotation data frames. One is for the genes and the other for the samples. They are generated usingg the information from the annotation files, and are taken as an argument for the pheatmap function.

##Generating the heat maps

There are four heat maps generated by this pipeline. The first two heat maps show the data clustered both by genes and by samples, whereas the third and fourth are only clustered by genes.

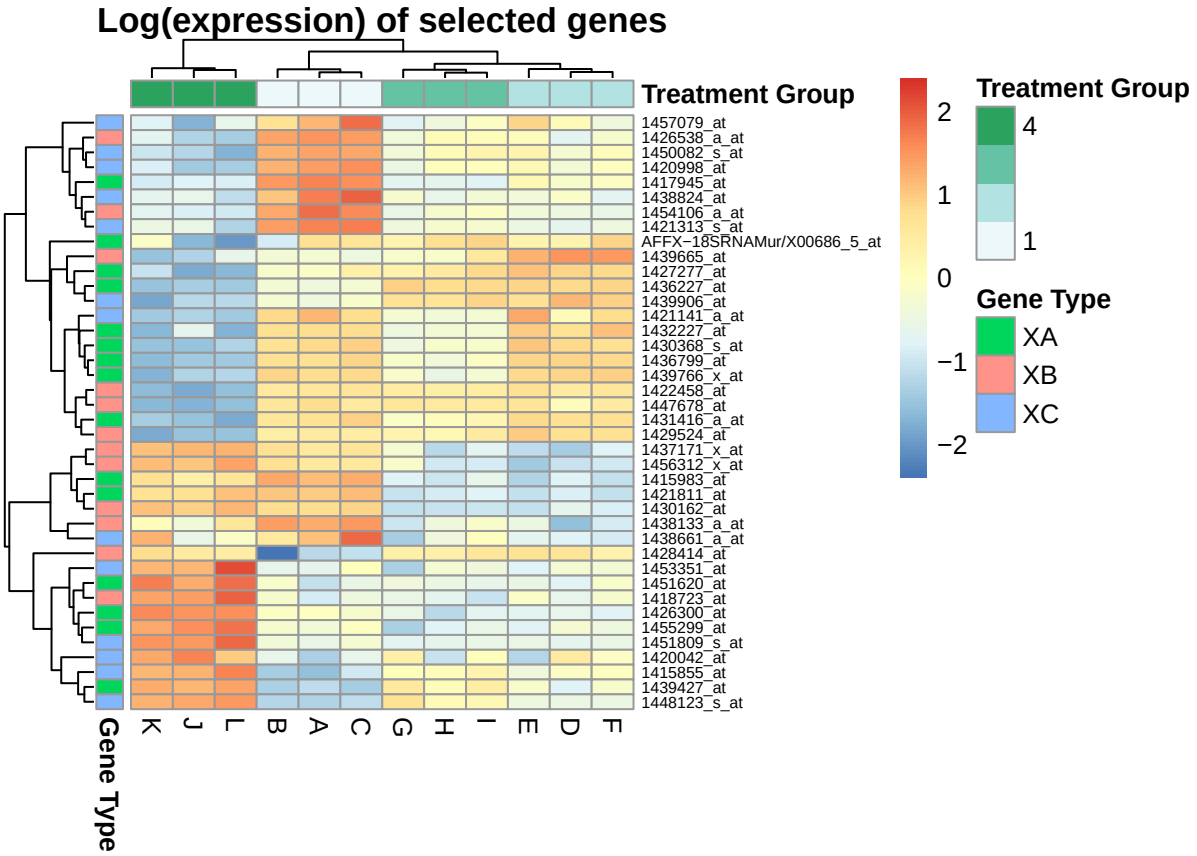
The first and second heat map only differ by the addition of axis labels, which results in a slight shrinking of the heat map that might make it harder for some users to read, so both maps are presented here.

The same is true of maps 3 and 4, with map 4 simply being a labeled version of map 3.

A row font size of 6 was selected because it was the largest font size possible that kept the gene names non-overlapping while allowing a box height that would fit onto the graphical display window for markdown

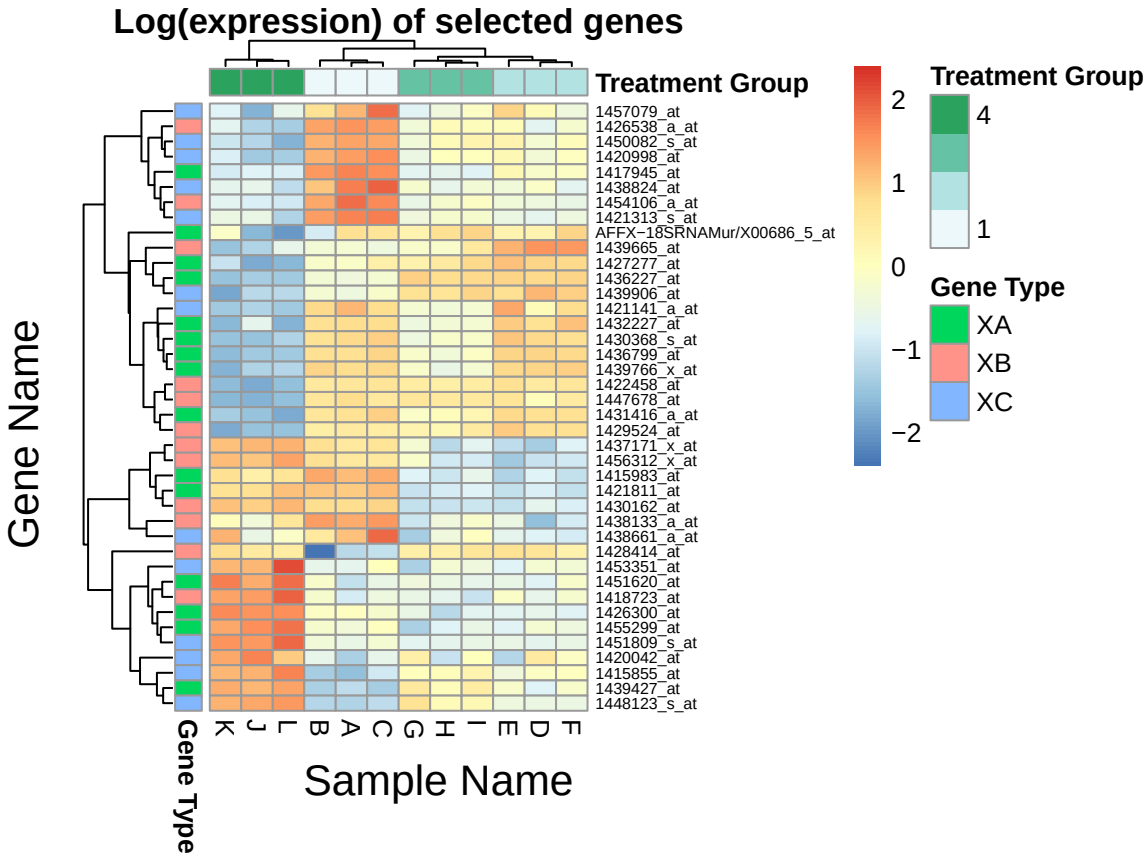
Heat map 1: Clustering by genes and samples

```
pheatmap(labeled_data_matrix, scale='row', fontsize_row = 6,
         annotation_row=genetype_annotate,
         annotation_col=treatment_annotate,
         main = "Log(expression) of selected genes",
         treeheight_col = 10, treeheight_row = 30)
```



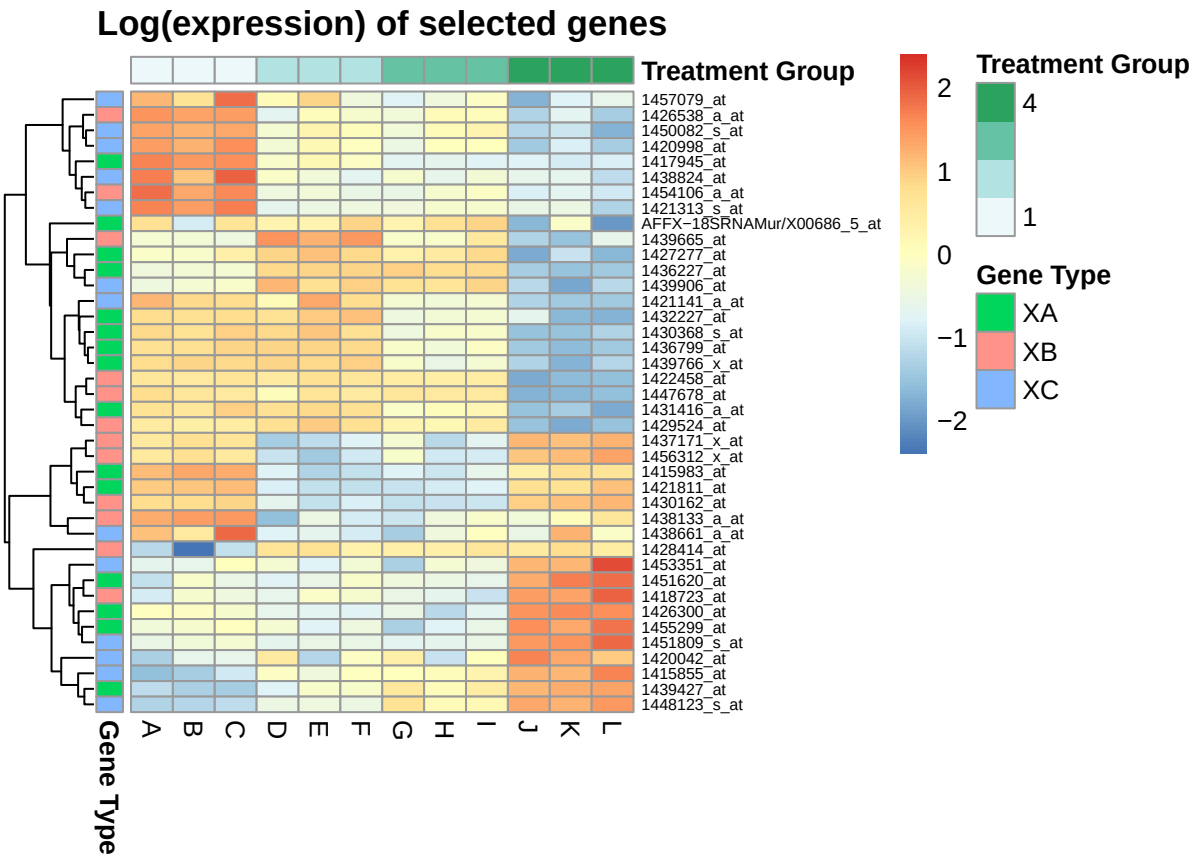
Heat map 2: Clustering by genes and samples (With axis labels)

```
library(grid)
setHook("grid.newpage", function() pushViewport(viewport(x=1,y=1,width=0.9,
                                                         height=1.0, name="vp",
                                                         just=c("right","top"))), action="prepend")
pheatmap(labeled_data_matrix, scale='row', fontsize_row = 6,
         annotation_row=genetype_annotate,
         annotation_col=treatment_annotate,
         main = "Log(expression) of selected genes",
         treeheight_col = 5, treeheight_row = 30)
setHook("grid.newpage", NULL, "replace")
grid.text("Sample Name", x=0.32, y=0.1, gp=gpar(fontsize=16))
grid.text("Gene Name", x=-0.05, rot=90, gp=gpar(fontsize=16))
```



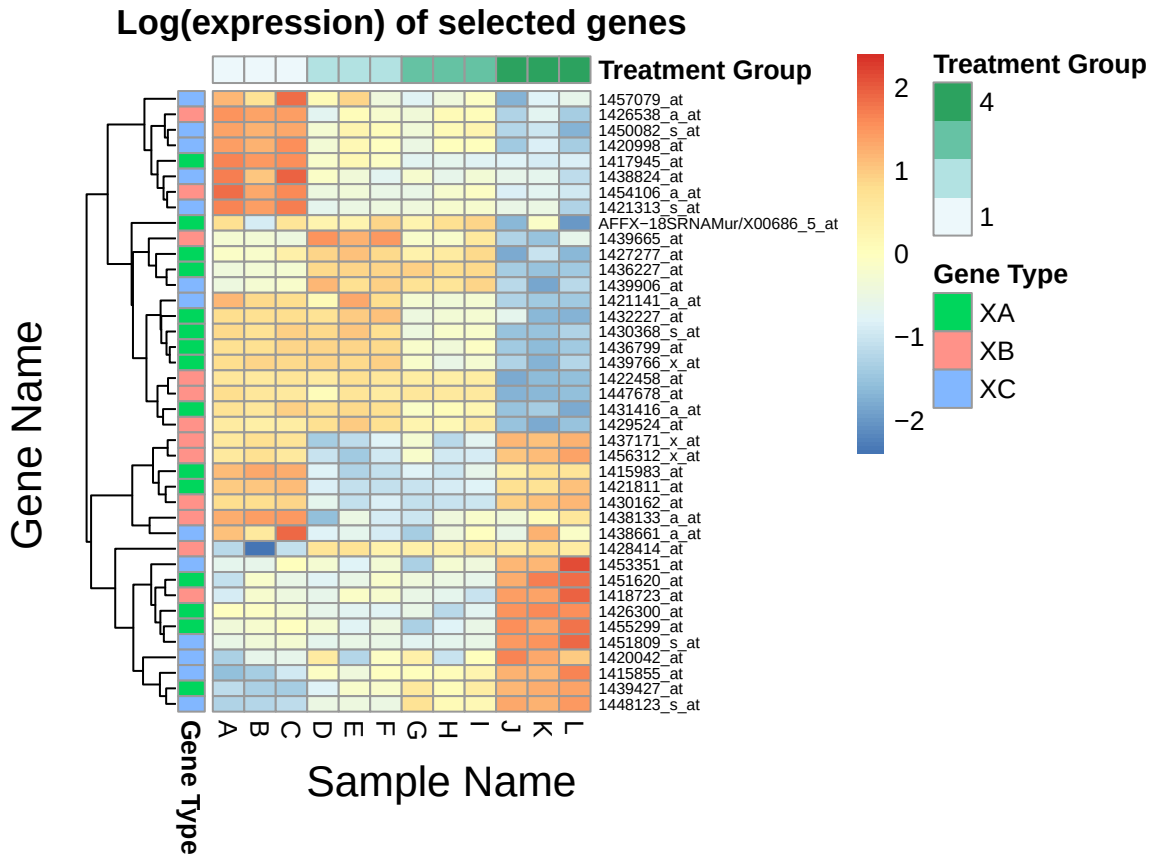
Heat map 3: Clustering by genes only

```
pheatmap(labeled_data_matrix, scale='row', fontsize_row = 6,
         annotation_row=genetype_annotate,
         annotation_col=treatment_annotate,
         main = "Log(expression) of selected genes",
         cluster_cols = FALSE, treeheight_row = 30)
```



Heat map 4: Clustering by genes only (With axis labels)

```
library(grid)
setHook("grid.newpage", function() pushViewport(viewport(x=1,y=1,width=0.9,
                                                         height=1.0, name="vp",
                                                         just=c("right","top"))), action="prepend")
pheatmap(labeled_data_matrix, scale='row', fontsize_row = 6,
         annotation_row=genetype_annotate,
         annotation_col=treatment_annotate,
         main = "Log(expression) of selected genes",
         cluster_cols = FALSE, treeheight_row = 30)
setHook("grid.newpage", NULL, "replace")
grid.text("Sample Name", x=0.32, y=0.1, gp=gpar(fontsize=16))
grid.text("Gene Name", x=-0.05, rot=90, gp=gpar(fontsize=16))
```



#Interpreting the heatmaps

Heat maps are a data visualization tool that utilize a color scale to represent values on a numerical scale.

These heat maps are being used to show gene expression. As indicated in the legend, warmer colors correspond to higher levels of gene expression whereas colder colors indicate lower levels of gene expression.

The data includes gene expression values from four different treatment groups. The experiment is therefore evaluating whether gene expression changes between groups as a result of their differential treatment. This change is observed as a difference in the expression level of a given gene across columns. More importantly, significantly affected genes will show substantially different expression levels between entire treatment groups.

For example, the first eight genes in the heat maps show high expression in treatment group one, middling expression in groups two and three, and low expression in group four. This indicates that treatment one is likely increasing the expression of these genes whereas treatment four is likely decreasing the expression of these genes, with treatments two and three having a mid affect or no affect on the expression of these genes.

The clustering feature of the heat maps puts rows or columns that are more similar in their expression profile closer together. In the above example of the top eight genes it is clear that they get clustered together because they all show a similar pattern of high, mid, then low expression across groups 1, 2/3, and 4, respectively.

The clustering of the samples also shows that treatment groups 2 and 3 have the most similar expression profiles across the set of genes, and that these two groups collectively are more similar to group 1 than group 4, as group 4 has the most disparate expression profile from the rest of the groups.

#The Raw Data Here is the data that was plotted in the heat maps. Note: It is log transformed.

I'm aware that this data puts the report above the page suggestion, so think of this as the supplemental data. Only here to see if you want to look at it.

I also chose to include an extra pair of graphs for view preference and place the graphs on individual pages to keep the pages from looking cluttered, so this artificially increased the page count

```
labeled_data
```

```
##     Type                    LongName         A        B        C        D
## 1     XA                  1436799_at 1.8313566 1.840931 1.884648 1.893870
## 2     XA                  1436227_at 1.6871924 1.734822 1.746053 2.161740
## 4     XA                  1417945_at 2.2716081 2.215295 2.242368 1.631528
## 8     XA                1430368_s_at 1.9299144 1.896536 1.965739 1.933764
## 10    XA                1439766_x_at 1.6883511 1.704857 1.697641 1.715152
## 23    XA                  1432227_at 1.7278696 1.717537 1.730158 1.711048
## 26    XA                  1426300_at 1.3970803 1.391754 1.354466 1.273482
## 29    XA                  1421811_at 1.7523600 1.762313 1.780303 1.360843
## 30    XA                  1439427_at 1.3162415 1.267788 1.261158 1.387205
## 35    XA                  1415983_at 1.8607925 1.899247 1.890432 1.454641
## 44    XA                1431416_a_at 2.0263430 2.002166 2.070685 2.030206
## 49    XA                  1427277_at 1.4049938 1.397590 1.499440 1.605552
## 55    XA AFFX-18SRNAMur/X00686_5_at 2.3548659 2.026778 2.337164 2.262123
## 68    XA                  1455299_at 1.4254080 1.452117 1.490696 1.443411
## 69    XA                  1451620_at 1.1207913 1.291742 1.218511 1.182167
## 71    XB                  1422458_at 2.2309173 2.221095 2.245689 2.211908
## 73    XB                1437171_x_at 1.8779848 1.909484 1.897631 1.512390
## 81    XB                  1428414_at 1.6244327 1.397510 1.640817 1.961785
## 83    XB                1426538_a_at 2.3609072 2.336826 2.345850 1.947933
## 92    XB                1454106_a_at 1.8291092 1.731805 1.786452 1.406972
## 104   XB                1438133_a_at 2.0926613 2.119161 2.130426 1.582922
## 109   XB                  1418723_at 0.9515562 1.077821 1.032099 1.005288
## 113   XB                  1447678_at 2.1435488 2.115857 2.105710 2.019156
## 115   XB                  1430162_at 2.1025855 2.103118 2.123833 1.848056
## 116   XB                  1439665_at 1.4000251 1.392403 1.369293 1.712435
## 118   XB                1456312_x_at 1.8749520 1.914319 1.878080 1.598591
## 127   XB                  1429524_at 1.7915241 1.769419 1.783759 1.834917
## 141   XC                1438661_a_at 1.3991715 1.306900 1.540912 1.071217
## 145   XC                1421313_s_at 2.0249113 1.977041 2.028091 1.621806
## 155   XC                1421141_a_at 1.8369605 1.783993 1.772734 1.657895
## 161   XC                  1415855_at 1.6236835 1.653758 1.726683 1.873526
## 168   XC                  1439906_at 1.7327968 1.767521 1.785214 2.012752
## 173   XC                  1438824_at 1.6453606 1.530237 1.681714 1.336113
## 176   XC                1450082_s_at 2.3144378 2.293374 2.309107 2.042467
## 177   XC                  1457079_at 1.3513668 1.273585 1.457898 1.172509
## 178   XC                1448123_s_at 1.7530620 1.757399 1.776722 1.881177
## 183   XC                1451809_s_at 1.3454822 1.375309 1.395314 1.330298
## 190   XC                  1420998_at 2.2500669 2.212966 2.268437 1.964580
## 191   XC                  1420042_at 1.2965562 1.417099 1.415317 1.597330
## 195   XC                  1453351_at 1.3987078 1.407276 1.508824 1.464704
```

```
##           E        F         G         H         I         J         K
## 1   1.885724 1.877326 1.4963030 1.4291144 1.5233156 1.0342616 0.9545863
## 2   2.179506 2.166644 2.1970081 2.1325469 2.1623193 1.3440787 1.2899737
## 4   1.752639 1.671943 1.4587578 1.4787292 1.4455445 1.4163615 1.3820796
## 8   1.992466 1.910022 1.5616989 1.6348783 1.6338259 1.2472388 1.2491237
## 10  1.694862 1.719120 1.4096273 1.2997842 1.3799758 1.1026593 0.9775801
## 23  1.778947 1.792110 1.4459941 1.4876939 1.4912212 1.4047550 1.1788912
## 26  1.247153 1.244791 1.2844712 1.1466072 1.2512369 1.7370982 1.7595157
## 29  1.302732 1.311255 1.3226422 1.3491937 1.3711290 1.7025981 1.6869572
## 30  1.515727 1.521673 1.6745691 1.5732681 1.6500153 1.8122806 1.8264820
## 35  1.359003 1.398229 1.4599620 1.4162147 1.4948220 1.7038243 1.7861158
## 44  2.055004 2.038236 1.8533665 1.8951540 1.9226095 1.5768573 1.5997964
## 49  1.645149 1.604718 1.4890346 1.5308102 1.6049602 1.0613028 1.2198713
## 55  2.256962 2.386264 2.2611602 2.3540586 2.3860868 1.8724015 2.1781636
## 68  1.354706 1.409042 1.2424580 1.3451599 1.3890640 1.7973748 1.7538152
## 69  1.222310 1.292283 1.2468187 1.2344684 1.2111933 1.5704244 1.6548025
## 71  2.261537 2.237475 2.2104383 2.1995944 2.2065851 1.7767116 1.8150985
## 73  1.556007 1.619984 1.7211810 1.5518954 1.6366519 1.9971082 1.9787373
## 81  1.968356 1.900324 1.9073554 1.9073184 1.9555615 1.9324574 1.9884714
## 83  2.097041 2.039644 2.0180324 2.1039475 2.0904733 1.8438993 1.9534662
## 92  1.423904 1.387570 1.3895169 1.4487637 1.4764650 1.3370110 1.3608980
## 104 1.776663 1.704204 1.6903044 1.7929595 1.8388005 1.8039654 1.8769845
## 109 1.089929 1.072763 1.0121489 0.9992034 0.9325731 1.3623493 1.3602322
## 113 2.117805 2.093844 2.1136900 2.1051315 2.1038967 1.7049822 1.7148608
## 115 1.775287 1.817636 1.7706728 1.7779209 1.7911952 2.1327924 2.1587502
## 116 1.664661 1.708269 1.4198963 1.4131585 1.5438675 1.2178018 1.1780498
## 118 1.534799 1.612940 1.7531019 1.6126154 1.6262181 1.9672088 1.9774470
## 127 1.873344 1.835427 1.7518613 1.7308316 1.7926548 1.4371154 1.3880907
## 141 1.096921 1.056772 0.9691464 1.1392010 1.2091846 1.1125540 1.4199230
## 145 1.654172 1.660861 1.6688406 1.7016932 1.7024881 1.6401690 1.6491480
## 155 1.859377 1.771291 1.5893815 1.5759198 1.5906183 1.4256493 1.3940183
## 161 1.814012 1.880926 1.8922096 1.9004710 1.9330940 2.0917839 2.0850571
## 168 1.939037 1.970608 1.9296155 1.9194502 1.9584011 1.6184437 1.5076671
## 173 1.302756 1.242794 1.3212892 1.2628373 1.3114505 1.2580284 1.2543607
## 176 2.134752 2.097532 2.0289021 2.1099145 2.1308989 1.8824641 1.9243278
## 177 1.304522 1.089401 1.0319415 1.0896807 1.1408262 0.8701196 1.0194067
## 178 1.888523 1.879993 2.0733935 1.9837201 1.9912145 2.1806215 2.1667622
## 183 1.348077 1.344933 1.3216697 1.3291366 1.3446344 1.6790308 1.6874728
## 190 2.046948 2.013949 1.9367976 2.0233443 2.0210601 1.7820132 1.8827247
## 191 1.317988 1.506316 1.5697393 1.3473325 1.5147651 1.7867654 1.7296960
## 195 1.376972 1.458791 1.2937413 1.4653076 1.4432136 1.6979639 1.6977622
##           L
## 1   1.031538
## 2   1.319367
## 4   1.409173
## 8   1.312856
## 10  1.117163
## 23  1.167431
## 26  1.749782
## 29  1.775297
## 30  1.850058
## 35  1.765420
## 44  1.516786
## 49  1.091245
```

```
## 55  1.806079
## 68  1.844344
## 69  1.685232
## 71  1.820079
## 73  2.004434
## 81  1.928080
## 83  1.828131
## 92  1.317028
## 104 1.975869
## 109 1.462699
## 113 1.727864
## 115 2.168696
## 116 1.342740
## 118 2.024819
## 127 1.435591
## 141 1.191764
## 145 1.517004
## 155 1.397106
## 161 2.164230
## 168 1.612690
## 173 1.169245
## 176 1.804616
## 177 1.052358
## 178 2.204355
## 183 1.749376
## 190 1.791001
## 191 1.674849
## 195 1.852259
```