R implementation

Sleiman Bassim, PhD

March 1, 2015

Locally loaded functions:

```
source("~/data/Dropbox/humanR/01funcs.R")
#load("", .GlobalEnv)
lsos(pat="")
```

	Type	Size	PrettySize	Rows	Columns
a.df	tbl_df	6848	6.7 Kb	42	2
a.list	list	6144	6 Kb	2	NA
bagging	function	23520	23 Kb	NA	NA
bagging.clas	function	24280	23.7 Kb	NA	NA
baggingTune	function	23856	23.3 Kb	NA	NA
circos.test	function	55408	54.1 Kb	NA	NA
Cols	function	6576	6.4 Kb	NA	NA
ensemble.mean	function	24232	23.7 Kb	NA	NA
i	integer	48	48 bytes	1	NA
indices.df	integer	4040	3.9 Kb	1000	NA
input.df	data.frame	2432000	2.3 Mb	10027	20
input.subset.df	tbl_df	1497248	1.4 Mb	1000	20
input.summary	function	12808	12.5 Kb	NA	NA
locusRMR	function	11224	11 Kb	NA	NA
model.clas	function	20680	20.2 Kb	NA	NA
model.reg	function	18528	18.1 Kb	NA	NA
modelTune.clas	function	21016	20.5 Kb	NA	NA
modelTune.reg	function	18864	18.4 Kb	NA	NA
net2Bin	function	33240	32.5 Kb	NA	NA
normDataWithin	function	21816	21.3 Kb	NA	NA
olBarplot	function	31992	31.2 Kb	NA	NA
output.df	data.frame	3568	3.5 Kb	46	3
output.df.log	data.frame	7304	7.1 Kb	29	3
output.df.reshape	cast_df	15944	15.6 Kb	21	3
output.df.stand	matrix	640	640 bytes	29	1
output.df.variants	data.frame	7208	7 Kb	29	3
overLapper	function	86584	84.6 Kb	NA	NA
pkgs	character	392	392 bytes	6	NA
predict.regsubsets	function	10208	10 Kb	NA	NA
range_correlation	function	16472	16.1 Kb	NA	NA
regfit	function	9912	9.7 Kb	NA	NA
regfit.int	function	10360	10.1 Kb	NA	NA
rocplot	function	7768	7.6 Kb	NA	NA
set.cor	function	10712	10.5 Kb	NA	NA
set.var	function	12216	11.9 Kb	NA	NA
summarySE	function	30400	29.7 Kb	NA	NA
summarySEwithin	function	39616	38.7 Kb	NA	NA
testing	list	2882440	2.7 Mb	2	NA
top100	tbl_df	70032	68.4 Kb	103	12
vennPlot	function	562632	549.4 Kb	NA	NA
visualizeNet	function	30288	29.6 Kb	NA	NA

1 By-group aggregation

By-group aggregation uses the principle of split-and-conquer or **split-apply-combine** a term coined by Hadley Wickham. R jobs will run in parallel across several cores which reduces the calculation time of the row search.

```
pkgs <- c('rbenchmark', 'doParallel', 'foreach', 'Hmisc', 'dplyr', 'plyr')
lapply(pkgs, require, character.only = TRUE);
```

```
[[1]]
[1] TRUE

[[2]]
[1] TRUE

[[3]]
[1] TRUE

[[4]]
[1] TRUE

[[5]]
[1] TRUE

[[6]]
[1] TRUE
```

Get data from this site here. It is in *.xlsx form. Use `read.xlsx` from the package `xlsx` to read in the table into an R dataframe. Put it in a wrapped dataframe with `dplyr` and saved it in a .Rdata file.

```
load("LoF_Phase1_v3.Rdata", .GlobalEnv)
lsos(pattern="input.*")
```

```
                     Type    Size PrettySize  Rows Columns
input.df        data.frame 2432000     2.3 Mb 10027      20
input.subset.df     tbl_df 1497248     1.4 Mb  1000      20
input.summary     function   12808    12.5 Kb    NA      NA
```

Sampling 1000 rows from the dataset is followed with a grouping and summarizing functions to show detail of the data and the levels of classification.

```
indices.df <- sample(1:nrow(input.df),1000)
```

```
#input.df <- input.df[complete.cases(input.df), ]
input.subset.df <- tbl_df(input.df[indices.df, ])
str(input.subset.df)
```

```
Classes 'tbl_df', 'tbl' and 'data.frame': 1000 obs. of  20 variables:
 $ chr             : Factor w/ 23 levels "1","10","11",..: 2 8 11 13 21 12 17 1 16 1 ...
 $ pos             : num  101180550 48177825 53116811 20033367 145139945 ...
 $ rsID            : Factor w/ 5383 levels ".","rs10009430",..: 1541 1 2426 5056 3447 2223 1 4182 4576 1
 $ ref_allele      : Factor w/ 145 levels "A","AAACC","AAC",..: 107 107 33 33 33 107 107 1 75 68 ...
 $ alt_allele      : Factor w/ 89 levels "A","AAC","AAG",..: 20 1 65 40 65 1 1 40 65 20 ...
 $ ancestral_allele: Factor w/ 125 levels "-",".","A","a",..: 94 94 93 29 29 94 94 3 67 61 ...
 $ effect          : Factor w/ 2 levels "full","partial": 2 1 1 2 1 1 2 2 1 1 ...
 $ type            : Factor w/ 3 levels "frameshift_indel",..: 2 3 3 2 2 3 2 2 3 1 ...
 $ an              : num  2184 2184 2184 2184 2184 ...
 $ ac              : num  1 1 49 561 1 2 1 1 1 67 ...
 $ gene            : Factor w/ 6433 levels "A1BG","A2M","A2ML1",..: 2597 33 6404 1698 2603 4750 797 101
 $ gene_id         : Factor w/ 6433 levels "ENSG00000000419.7",..: 1616 2596 3959 860 5563 213 5310 267
 $ lof_trans       : num  1 4 7 1 2 3 2 1 1 3 ...
 $ all_trans       : num  3 4 7 3 2 3 3 3 1 3 ...
 $ failed_filters  : num  1 0 0 1 1 0 0 0 0 0 ...
 $ filters_failed  : Factor w/ 69 levels "0","all_alt",..: 46 1 1 65 46 1 1 1 1 1 ...
 $ splice_type     : Factor w/ 3 levels ".","acceptor",..: 3 1 1 2 2 1 2 3 1 1 ...
 $ canonical       : Factor w/ 3 levels ".","NO","YES": 2 1 1 3 2 1 3 3 1 1 ...
 $ other_canonical : Factor w/ 3 levels ".","NO","YES": 2 1 1 3 2 1 3 3 1 1 ...
 $ intron_length   : Factor w/ 2266 levels ".","1","100",..: 344 1 1 1177 523 1 2002 2006 1 1 ...
```

```
unique(input.subset.df$effect)
```

```
[1] partial full    <NA>
Levels: full partial
```

```
unique(input.subset.df$chr)
```

```
 [1] 10   16   19   20   8    2    4    1    3    22   5    14   12   6    9    11
[17] 7    15   18   13   21   17   X    <NA>
Levels: 1 10 11 12 13 14 15 16 17 18 19 2 20 21 22 3 4 5 6 7 8 9 X
```

10 Extract subsets of the dataframe depending on the "effect" column.

```
input.summary <- function(x){
```

```
  y <- unique(x)
  sub = list(NULL)
  for(i in 1:length(y)){
    sub[[i]] <- subset(input.subset.df, x == y[i])
  }
  sub
}

testing <- input.summary(input.subset.df$effect)
testing[1]

[[1]]
Source: local data frame [463 x 20]

   chr       pos        rsID ref_allele alt_allele ancestral_allele  effect
1   10 101180550 rs145279669          T          C                T partial
2   20  20033367   rs7508949          C          G                C partial
3    4  42146013           .          T          A                T partial
4    1   1888057 rs191790164          A          G                A partial
5    2 241555861 rs116229176          G          A                G partial
6    2 169923686 rs151018449          C          A                C partial
7    4 170634334           .          C          T                C partial
8    8  69699676 rs150041402          A          C                A partial
9    3  46785355   rs4075012          T          C                C partial
10  19   2340154           .          G         GC               GC partial
.. ...       ...         ...        ...        ...              ...     ...
Variables not shown: type (fctr), an (dbl), ac (dbl), gene (fctr), gene_id (fctr),
  lof_trans (dbl), all_trans (dbl), failed_filters (dbl), filters_failed (fctr),
  splice_type (fctr), canonical (fctr), other_canonical (fctr), intron_length (fctr)

testing[2]

[[1]]
Source: local data frame [536 x 20]

   chr       pos        rsID ref_allele alt_allele ancestral_allele effect
1   16  48177825           .          T          A                T   full
2   19  53116811  rs17855778          C          T                N   full
3    8 145139945 rs187027021          C          T                C   full
4    2  85581474 rs150217356          T          A                T   full
5    3  49842325  rs35389534          G          T                G   full
6    1  40981007           .        CTG          C              CTG   full
7   22  37398102           .          G          A                .   full
8    5 148680794           .          G          A                G   full
9   14  98100029 rs182090492          C          T                C   full
10  12  11461802  rs12829245          G          A                .   full
.. ...       ...         ...        ...        ...              ...    ...
Variables not shown: type (fctr), an (dbl), ac (dbl), gene (fctr), gene_id (fctr),
  lof_trans (dbl), all_trans (dbl), failed_filters (dbl), filters_failed (fctr),
  splice_type (fctr), canonical (fctr), other_canonical (fctr), intron_length (fctr)
```

2 Different grouping and summarizing Functions

There is several functions to be used to extract a specific amount of data. The `tapply, agreggate, ddply` functions:

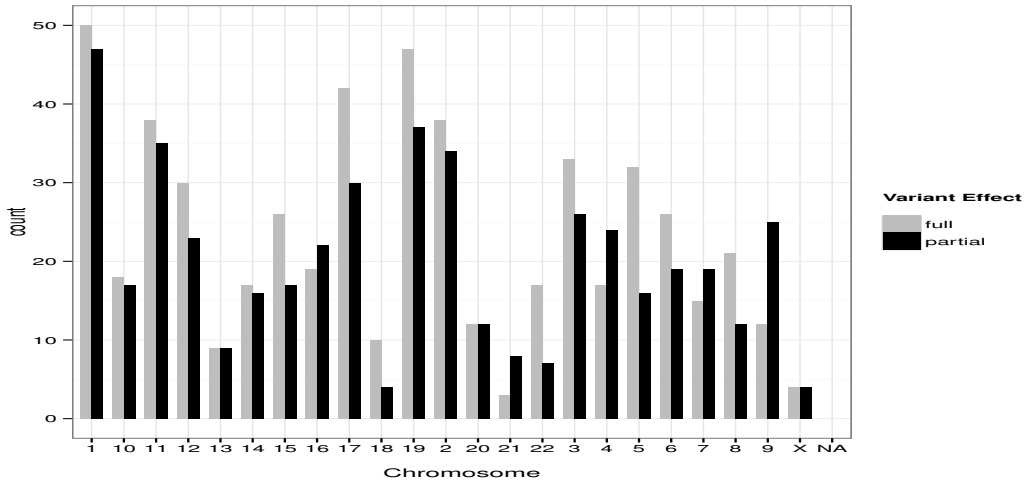
```
tapply(input.subset.df$pos, input.subset.df$effect, mean)

    full  partial
73933120 74055359
```

Extract and visualize a summary of the partial and full effects of variants relative to the nature of the chromosome.

```
require(plyr)
```

```
output.df <- ddply(
    input.subset.df,
    c("effect","chr"),
    function(x){
        c(count = nrow(x))
    })
## Plot
require(ggplot2)
ggplot(output.df) +
    geom_bar(aes(x = chr, y = count, fill = effect),
             stat = 'identity', position = 'dodge', width = .7) +
    scale_fill_manual("Variant Effect\n", values = c("grey", "black"),
                      labels = c('full', 'partial')) +
    labels(x = '\nChromosome', 'Count\n') +
    theme_bw()
```



Visualize a count summary relative to the nature of variants.

```
output.df.variants <- ddply(
```

```
    input.subset.df,
    c('effect','alt_allele'),
    function(x){
    c(count = nrow(x))
    })
head(output.df.variants)
```

```
  effect alt_allele count
1   full          A   225
2   full         AG     1
3   full          C    49
4   full         CA     2
5   full     CACACT     1
6   full         CG     3
```

```
require(reshape)
output.df.reshape <- cast(output.df.variants, alt_allele~effect)
head(output.df.reshape)
```

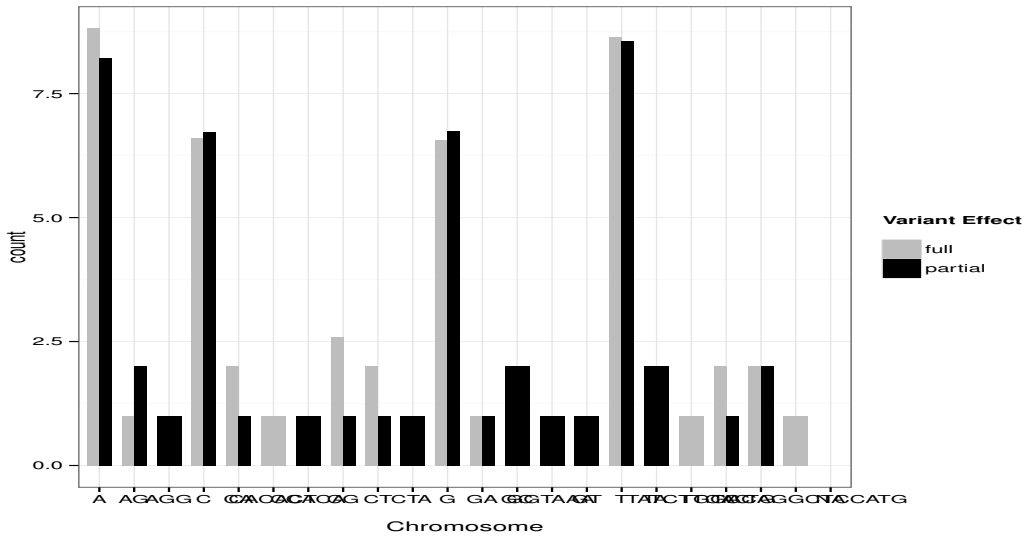
```
  alt_allele full partial NA
1          A  225     148 NA
2         AG    1       2 NA
3        AGG   NA       1 NA
4          C   49      53 NA
5         CA    2       1 NA
6     CACACT    1      NA NA
```

```
output.df.reshape[is.na(output.df.reshape)] <- 0
```

```
Warning in `[<-.factor`(`*tmp*`, thisvar, value = 0):  invalid factor level, NA
generated
```

```
require(vegan)
output.df.stand <- decostand(output.df.variants$count, method="log")
output.df.log <- data.frame(effect = output.df.variants$effect,
                            pos= output.df.variants$alt_allele,
                            count= output.df.stand[,1])

ggplot(output.df.log) +
    geom_bar(aes(x = pos, y = count, fill = effect),
             stat = 'identity', position = 'dodge', width = .7) +
    scale_fill_manual("Variant Effect\n", values = c("grey", "black"),
                      labels = c('full', 'partial')) +
    labs(x = '\nChromosome', 'Count\n') +
    theme_bw()
```



Create a data frame from a list.

```
a.list <- list(x=output.df.reshape[,1], y=output.df.reshape[,2])
system.time(a.df <- tbl_df(do.call("rbind.fill",lapply(a.list, as.data.frame))))

   user  system elapsed
  0.004   0.000   0.002
```

```
registerDoParallel(cores = 2)
benchmark(replications=100, lapply(1:3, sqrt), foreach(i=1:3) %do% sqrt(i))

                          test replications elapsed relative user.self sys.self
2 foreach(i = 1:3) %do% sqrt(i)          100   0.575      575      0.54    0.028
1             lapply(1:3, sqrt)          100   0.001        1      0.00    0.000
  user.child sys.child
2          0         0
1          0         0
```

2.1 More dplyr verbs for data manipulation

Verbs in `dplyr` constitute the essential functions for data manipulation. Five exist and are basic dataframe manipulation functions. **Filter** to choose rows, **select** to choose from columns, **arrange** to order rows, **mutate** and **summarize** to make new columns.

```
input.subset.df %>%
    select(chr, pos, ref_allele, effect) %>%
    filter(ref_allele == "A") %>%
    group_by(chr, effect) %>%
    mutate(pos2 = rank(pos)) %>%
    filter(pos < 8.7*10^6 & pos > 8*10^6 ) %>%
    arrange(chr, effect, pos2)

Source: local data frame [2 x 5]
Groups: chr, effect

  chr     pos ref_allele  effect pos2
1   1 8585816          A partial    9
2  12 8288179          A    full    8
```

3 Other options of parallel computing

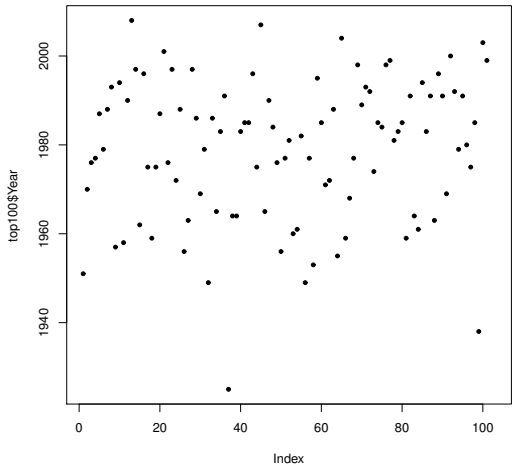
For MySQL try `dbApply()` to aggregate. And `sqldf` will create a temporary database. Or bigmemory.org to load data into memory, or use Hadoop.

4 Top 100 papers cited in Nature

Download data from onedrive in xlsx form.

```
#require(xlsx)
#top100 <- read.xlsx("~/Downloads/WebofSciencetop100.xlsx", sheetIndex = "Sheet1")
load("WebofScienceTop100.Rdata", .GlobalEnv)
top100 <- tbl_df(top100)
plot(top100$Year,top100$Rnk, pch = 20)
abline(lm(top100$Rank~top100$Year), col = "red");

Warning in model.response(mf, "numeric"):  using type = "numeric" with a factor
response will be ignored
Warning in Ops.factor(y, z$residuals):  '-' not meaningful for factors
```

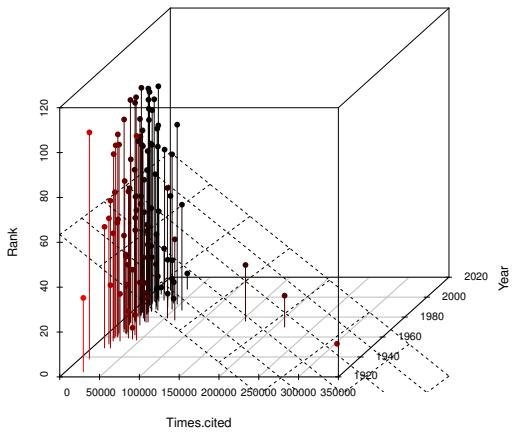


Plot in 3d. `rgl` provides an interactive 3d box.

```
require(scatterplot3d)
scat <- with(top100,
             scatterplot3d(Times.cited,
                           Year, Rank, pch=16,
                           highlight.3d=TRUE,
                           type="h"
                           )
             )
fit <- with(top100, lm(Rank~Times.cited+Year))
```

```
Warning in model.response(mf, "numeric"):  using type = "numeric" with a factor
response will be ignored
Warning in Ops.factor(y, z$residuals):  '-' not meaningful for factors
```

```
scat$plane3d(fit)
```



5 System Information

The version number of R and packages loaded for generating the vignette were:

```
R version 3.1.2 (2014-10-31)
Platform: x86_64-unknown-linux-gnu (64-bit)

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C               LC_TIME=en_US.UTF-8
 [4] LC_COLLATE=en_US.UTF-8     LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C                  LC_ADDRESS=C
[10] LC_TELEPHONE=C             LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

attached base packages:
 [1] splines   grid      parallel  stats     graphics  grDevices utils     datasets
 [9] methods   base

other attached packages:
 [1] scatterplot3d_0.3-35 vegan_2.2-0          permute_0.8-3
 [4] reshape_0.8.5        httr_0.5             rgl_0.95.1158
 [7] plyr_1.8.1           Hmisc_3.14-6         Formula_1.1-2
[10] survival_2.37-7      lattice_0.20-29      doParallel_1.0.8
[13] iterators_1.0.7      foreach_1.4.2        rbenchmark_1.0.0
[16] ggplot2_1.0.0        dplyr_0.3.0.2        knitr_1.8

loaded via a namespace (and not attached):
 [1] acepack_1.3-3.3     assertthat_0.1      cluster_1.15.3      codetools_0.2-9
 [5] colorspace_1.2-4    compiler_3.1.2      DBI_0.3.1           digest_0.6.4
 [9] evaluate_0.5.5      foreign_0.8-61      formatR_1.0         gtable_0.1.2
[13] highr_0.4           labeling_0.3        latticeExtra_0.6-26 lazyeval_0.1.9
[17] magrittr_1.5        MASS_7.3-35         Matrix_1.1-4        mgcv_1.8-3
[21] munsell_0.4.2       nlme_3.1-118        nnet_7.3-8          proto_0.3-10
[25] RColorBrewer_1.0-5  Rcpp_0.11.3         reshape2_1.4        rpart_4.1-8
[29] scales_0.2.4        stringr_0.6.2       tcltk_3.1.2         tools_3.1.2
```