Psychology 350: Special Topics
An introduction to R for psychological research
Function development

William Revelle
Northwestern University
Evanston, Illinois USA

https://personality-project.org/courses/350

May 2024

Outline

A function from scratch

1. Find a use case for your function.
2. Think about what you want it to do?
3. What is the problem you are trying to solve?
4. Does not need to be a new statistic, can just be something that helps you in your work

A function

First create the skeleton of the function.
Think about default values for various options.
Do not "hard code" values. Let the function determine the values.

R code

```
myfunction <- function(a parameter list ) {  #start  from here

Do some basic preprocessing to check that the input is ok

Do the main part of the function

Wrap up what you want to return to the world

return(myresult)   #either one value or a list

} #end of the function
```

`selectBy` as a new function that met a particular need

1. Although core R has an easy way to select a subset of data, this is not obvious for frequent users of the *tidyverse*
2. The [command allows us to select a subset of x: `y <- x[some logical expression, ]`
3. The `subset` command will do this in a more readable fashion: `y <- subset(x, some logical expression)`
4. The `selectBy` and `splitBy` were added to *psychTools* in 2022 in response to a request from a 350 student.
   - Uses `grep` to find symbols, `gsub` to do substitutions
   - `strsplit` to break string into parts , `length` to determine how many times to do things
   - `if ..,else` for basic logical operations
   - `unlist` to convert lists to vectors

selectBy

R code

```
dim(attitude)
 testand <- selectBy(attitude,'rating < 60 & complaints > 50') #logica
testand
test.or <- selectBy(attitude,'rating < 50 | complaints > 84 ' ) #logi
test.or
```

```
dim(attitude)
[1] 30  7
>  testand <- selectBy(attitude,'rating < 60 & complaints > 50') #logical and
> dim(testand)
[1] 6 7
> testand
   rating complaints privileges learning raises critical advance
1      43         51         30       39     61       92      45
6      43         55         49       44     54       49      34
7      58         67         42       56     66       68      35
20     50         58         68       54     64       78      52
23     53         66         52       50     63       80      37
28     48         57         44       45     51       83      38
test.or <- selectBy(attitude,'rating < 50 | complaints > 84 ' ) #logical or
> dim(test.or)
[1] 7 7
> test.or
   rating complaints privileges learning raises critical advance
1      43         51         30       39     61       92      45
6      43         55         49       44     54       49      34
16     81         90         50       72     60       54      36
17     74         85         64       69     79       79      63
24     40         37         42       58     50       57      49
28     48         57         44       45     51       83      38
29     85         85         71       71     77       74      55
```

selectBy

```
#written May 20, 2023
"selectBy" <- function(x,by) {#use a quasi formula input

by <- gsub(" ","", by)   #this removes the spaces in the call

if(grepl("\\|",by)) {  AND <- FALSE
     bb <- unlist(strsplit(by,"\\|"))} else { #note to search for a | we have to escape it!
        AND  <- TRUE
        if(grepl(",",by)) {
              bb <- unlist(strsplit(by,","))} else {
               bb <- unlist(strsplit(by,"&"))} }

    n.by <- length(bb)
    by <- isvalue <- notvalue <- lessthan <- morethan <- matrix(NA,ncol= n.by)

    eq <- grep("=",bb)   #find which operators were used
    lt <- grep("<",bb)   #returns a vector
    gr <- grep(">",bb)
    ne  <- grep("!=",bb)
   #prepare the relevant search parameters
    if(length(eq ) >0)  {temp <- unlist(strsplit(bb[eq],"="))
              by[eq] <- temp[1]
              isvalue[eq] <-as.numeric( temp[2])
              }
    if( length(lt)  >0)  {temp <- unlist(strsplit(bb[lt],"<"))
              by[lt] <- temp[1]
              lessthan[lt] <- as.numeric( temp[2])
              }
     if(length(gr) >0)  {temp <- unlist(strsplit(bb[gr],">"))
              by[gr] <- temp[1]
              morethan[gr] <- as.numeric( temp[2])
              }
```

selectBy (continued)

```
      if(length(ne) >0)  {temp <- unlist(strsplit(bb[eq],"!="))
               by[ne] <- temp[1]
               notvalue[ne] <-as.numeric( temp[2])
               }

#make sure that the variable names are correct
 if(!all(by %in% colnames(x)))  {
            cat("\n Offending variables are ",by[!by %in% colnames(x) ],"\n")
          stop("Variables specified do not match the variables in the data.
               \nFix the names and try again")}
#do this on y which serves as  pointers to x, rather than x  #then combine the pointers for &
y <- matrix(TRUE,nrow=NROW(x), ncol=n.by)

for(i in 1:length(by)) {
        if(!is.na(isvalue[,i])) y[,i] <-  x[,by[i]]==isvalue[i]
        if(!is.na(notvalue[,i])) y[,i] <- (x[,by[i]]!= notvalue[i])
        if(!is.na(lessthan[,i])) y[,i] <- (x[,by[i]]< lessthan[i])
        if(!is.na(morethan[,i])) y[,i] <- (x[,by[i]] >  morethan[i])
}
if(AND ) {y <-apply(y,1,all)} else {y <- apply(y,1,any)}
y[is.na(y) ] <- FALSE
return(x[y,])
}
```

`splitBy` uses the same logic to create dichotomous splits

R code

```
test <- splitBy(attitude, 'rating > 70')
 headTail(test)
```

```
test <- splitBy(attitude, 'rating > 70 ')
> headTail(test)
    rating complaints privileges learning raises critical advance rating2
1       43         51         30       39     61       92      45       0
2       63         64         51       54     63       73      47       0
3       71         70         68       69     76       86      48       1
4       61         63         45       47     54       84      35       0
...    ...        ...        ...      ...    ...      ...     ...     ...
27      78         75         58       74     80       78      49       1
28      48         57         44       45     51       83      38       0
29      85         85         71       71     77       74      55       1
30      82         82         39       59     64       78      39       1
```

Doesn't seem to work with multiple cases. This needs to be fixed.

A collection of simple function to help search for specific information

1. Most use the %in% or `match` function
2. match returns a vector of the positions of (first) matches of its first argument in its second.
3. %in% is a more intuitive interface as a binary operator, which returns a logical vector indicating if there is a match or not for its left operand.
4. %in% is currently defined as "%in%" <- function(x, table) match(x, table, nomatch = 0) > 0
   From the help page for `match`

lookup function

lookup calls match but gives limited output

R code

lookup

```
"lookup" <-
function(x,y,criteria=NULL) {
if (is.null(criteria)) {temp <- match(x,rownames(y))} else {
     temp <- match(x,y[,criteria])}
     if(any(!is.na(temp))) {
 y <- format(y[temp[!is.na(temp)],,drop=FALSE],justify="left")
        } else {y <- NA}
return(y)}

#try match   (returns location of matching

> match("C4",rownames(bfi.dictionary))
[1] 9

lookup("C4",bfi.dictionary)
   ItemLabel                                  Item    Giant3
C4     q_626 Do things in a half-way manner. Stability Conscientiou
   IPIP100
C4    B5:C
```

keys.lookup combines lookup with a dictionary and keys.list

R code

keys.lookup

```
 keys.lookup
function (keys.list, dictionary)
{
    if (is.list(keys.list)) {
        items <- sub("-", "", unlist(keys.list))
        f <- make.keys(items, keys.list)
    }
    keys.list <- fa.sort(f)
    contents <- lookup(rownames(f), y = dictionary)
    contents <- format(contents, format = "left")
    rownames(contents)[rowSums(f) < 0] <- paste0(rownames(contents)[rowSums(f) <
        0], "-")
    return(contents)
}
```

Adequate but not very elegant output

R code

```
keys.lookup(bfi.keys,bfi.dictionary)
```

```
keys.lookup(bfi.keys,bfi.dictionary)
    ItemLabel                                          Item     Giant3              Big6
A1-     q_146  Am indifferent to the feelings of others. Cohesion   Agreeableness
A2      q_1162 Inquire about others' well-being.         Cohesion   Agreeableness
A3      q_1206 Know how to comfort others.               Cohesion   Agreeableness
A4      q_1364 Love children.                            Cohesion   Agreeableness
A5      q_1419 Make people feel at ease.                 Cohesion   Agreeableness
C1      q_124  Am exacting in my work.                   Stability  Conscientiousness
C2      q_530  Continue until everything is perfect.     Stability  Conscientiousness
C3      q_619  Do things according to a plan.            Stability  Conscientiousness
C4-     q_626  Do things in a half-way manner.           Stability  Conscientiousness
C5-     q_1949 Waste my time.                            Stability  Conscientiousness
E1-     q_712  Don't talk a lot.                         Plasticity Extraversion
E2-     q_901  Find it difficult to approach others.     Plasticity Extraversion
E3      q_1205 Know how to captivate people.             Plasticity Extraversion
E4      q_1410 Make friends easily.                      Plasticity Extraversion
E5      q_1768 Take charge.                              Plasticity Extraversion
N1      q_952  Get angry easily.                        Stability  Emotional Stability
N2      q_974  Get irritated easily.                    Stability  Emotional Stability
N3      q_1099 Have frequent mood swings.               Stability  Emotional Stability
N4      q_1479 Often feel blue.                          Stability  Emotional Stability
N5      q_1505 Panic easily.                             Stability  Emotional Stability
O1      q_128  Am full of ideas.                        Plasticity Openness
O2-     q_316  Avoid difficult reading material.        Plasticity Openness
O3      q_492  Carry the conversation to a higher level. Plasticity Openness
O4      q_1738 Spend time reflecting on things.         Plasticity Openness
O5-     q_1964 Will not probe deeply into a subject.    Plasticity Openness
```

`lookupFromKeys` is an example of a more useful function

A generalization of `keys.lookup` to a keys list with better output

1. Typical problem is that we have some set of items from which we want to form scales.
2. We might create a list of `item.keys` specifying which items to score and in which direction.
3. Then, we want to tell other people what the scale measures (or we forget)
4. We create a `dictionary` of the items in our scale (or maybe all the items we are using in our lab).
5. We then want to say which items are in which scales.

The bfi keys

R code

bfi.keys

```
 bfi.keys
$agree
[1] "-A1" "A2"  "A3"  "A4"  "A5"

$conscientiousness
[1] "C1"  "C2"  "C3"  "-C4" "-C5"

$extraversion
[1] "-E1" "-E2" "E3"  "E4"  "E5"

$neuroticism
[1] "N1" "N2" "N3" "N4" "N5"

$openness
[1] "O1"  "-O2" "O3"  "O4"  "-O5"
```

The bfi.dictionary

R code

```
bfi.dictionary
```

```
bfi.dictionary
          ItemLabel                                                Item     Giant3              Big6
A1            q_146  Am indifferent to the feelings of others.          Cohesion       Agreeableness
A2           q_1162          Inquire about others' well-being.          Cohesion       Agreeableness
A3           q_1206                Know how to comfort others.          Cohesion       Agreeableness
A4           q_1364                                 Love children.          Cohesion       Agreeableness
A5           q_1419                  Make people feel at ease.          Cohesion       Agreeableness
C1            q_124                    Am exacting in my work.         Stability   Conscientiousness
C2            q_530       Continue until everything is perfect.         Stability   Conscientiousness
C3            q_619             Do things according to a plan.         Stability   Conscientiousness
C4            q_626           Do things in a half-way manner.          Stability   Conscientiousness
C5           q_1949                                Waste my time.         Stability   Conscientiousness
E1            q_712                             Don't talk a lot.      Plasticity        Extraversion
E2            q_901       Find it difficult to approach others.      Plasticity        Extraversion
E3           q_1205              Know how to captivate people.      Plasticity        Extraversion
E4           q_1410                       Make friends easily.      Plasticity        Extraversion
E5           q_1768                                 Take charge.      Plasticity        Extraversion
N1            q_952                          Get angry easily.         Stability Emotional Stability
N2            q_974                        Get irritated easily.         Stability Emotional Stability
N3           q_1099                  Have frequent mood swings.         Stability Emotional Stability
N4           q_1479                          Often feel blue.         Stability Emotional Stability
N5           q_1505                             Panic easily.         Stability Emotional Stability
O1            q_128                          Am full of ideas.      Plasticity            Openness
O2            q_316          Avoid difficult reading material.      Plasticity            Openness
O3            q_492  Carry the conversation to a higher level.      Plasticity            Openness
O4           q_1738           Spend time reflecting on things.      Plasticity            Openness
O5           q_1964       Will not probe deeply into a subject.      Plasticity            Openness
gender       gender                             males=1, females=2          <NA>                <NA>
education education in HS, fin HS, coll,  coll grad , grad deg          <NA>                <NA>
age             age                                 age in years          <NA>                <NA>
```

The lookupFromKeys function

```
#adjusted 11/15/20 to add correlations if provided
 "lookupFromKeys" <-
function(keys.list,dictionary,n=20,cors = NULL,sort=TRUE,suppress.names=FALSE,digits=2){
n.scales <- length(keys.list)
results <- item.cors <- result.df <- list()
for(i in 1:n.scales) {
  list.name <- names(keys.list[i])
  list.i <- keys.list[[i]]
   keys <- rep(1,length(list.i))[1:(min(n,length(list.i)))]
    neg <- grep("-", list.i[1:(min(n,length(list.i)))])
    keys[neg] <- -1
  select <- sub("-", "", list.i)
  results[[i]] <- lookup(select[1:(min(n,length(list.i)))],dictionary)

 if(!is.null(rownames(results[[i]])[keys < 0]))  rownames(results[[i]])[keys < 0] <-
                                  paste0(rownames(results[[i]])[keys<0],"-")

 if(!is.null(cors)) { item.cors[[i]] <- round(cors[select[1:(min(n,length(select)))],i],
                   digits=digits)
 result.df[[i]] <- data.frame(results[[i]],cors=item.cors[[i]])
 if(sort) {
      ord <- order(abs(item.cors[[i]]),decreasing=TRUE)
      result.df[[i]] <- result.df[[i]][ord,]
 }
 } else {result.df[[i]] <- data.frame(results[[i]])}

   if(suppress.names) names(results[[i]]) <- ""
 # names(results[i]) <- list.name
}
names(result.df) <- names(keys.list)
return(result.df)}
```

using lookupFromKeys

R code

```
lookupFromKeys(bfi.keys, dictionary=bfi.dictionary, n =2) #just show t
```

```
lookupFromKeys(bfi.keys, dictionary=bfi.dictionary, n =3) #just show the first 2 items
$agree
    ItemLabel                                          Item   Giant3          Big6   Little12 Key
A1-     q_146 Am indifferent to the feelings of others. Cohesion Agreeableness Compassion
A2      q_1162         Inquire about others' well-being. Cohesion Agreeableness Compassion

$conscientiousness
   ItemLabel                                      Item    Giant3             Big6    Little12 Key
C1     q_124                Am exacting in my work. Stability Conscientiousness Orderliness
C2     q_530 Continue until everything is perfect. Stability Conscientiousness Orderliness

$extraversion
    ItemLabel                                       Item     Giant3         Big6    Little12 Keying
E1-     q_712                     Don't talk a lot. Plasticity Extraversion Sociability     -1
E2-     q_901 Find it difficult to approach others. Plasticity Extraversion Sociability     -1

$neuroticism
   ItemLabel                   Item    Giant3                Big6 Little12 Keying IPIP100
N1     q_952     Get angry easily. Stability Emotional Stability  Balance      -1    B5:N
N2     q_974 Get irritated easily. Stability Emotional Stability  Balance      -1    B5:N

$openness
    ItemLabel                                  Item     Giant3     Big6  Little12 Keying IPIP100
O1      q_128                 Am full of ideas. Plasticity Openness Intellect       1     B5:O
```

Don't show as many columns of the dictionary

R code

```
lookupFromKeys(bfi.keys, dictionary=bfi.dictionary[1:3], n =2) #just s
```

```
lookupFromKeys(bfi.keys, dictionary=bfi.dictionary[1:3], n =2) #just show the first 2 items
$agree
    ItemLabel                                          Item   Giant3
A1-     q_146 Am indifferent to the feelings of others. Cohesion
A2      q_1162         Inquire about others' well-being. Cohesion

$conscientiousness
   ItemLabel                                     Item    Giant3
C1     q_124                Am exacting in my work. Stability
C2     q_530 Continue until everything is perfect. Stability

$extraversion
    ItemLabel                                    Item     Giant3
E1-     q_712                      Don't talk a lot. Plasticity
E2-     q_901 Find it difficult to approach others. Plasticity

$neuroticism
   ItemLabel                   Item    Giant3
N1     q_952     Get angry easily. Stability
N2     q_974 Get irritated easily. Stability

$openness
    ItemLabel                                 Item     Giant3
O1      q_128                Am full of ideas. Plasticity
O2-     q_316 Avoid difficult reading material. Plasticity
```

Improving the function

1. It would be helpful to know the correlations of the items with the scales.
2. Find the correlations using `scoreItems` and then combine that information into our function

First use `scoreItems` and then use that information

R code

```
scores <- scoreItems(bfi.keys,bfi)
names(scores)
dim(scores$item.cor)
```

```
scores <- scoreItems(bfi.keys,bfi)
 names(scores)
 [1] "scores"          "missing"         "alpha"           "av.r"            "sn"
           "n.items"         "item.cor"        "cor"
 [9] "corrected"       "G6"              "item.corrected" "response.freq"   "raw"
              "ase"             "med.r"           "keys"
[17] "Call"

dim(scores$item.cor)
[1] 25  5
```

Combine that information with the item names and the keying information, sort in absolute order of correlations

R code

```
lookupFromKeys(bfi.keys, dictionary=bfi.dictionary[1:3],
                        cors=scores$item.cor, n =2)
```

```
lookupFromKeys(bfi.keys, n=2,dictionary=bfi.dictionary[1:2],cors=scores$item.cor)
$agree
    ItemLabel                                          Item  cors
A2     q_1162 Inquire about others' well-being.          0.73
A1-    q_146  Am indifferent to the feelings of others. -0.58

$conscientious
   ItemLabel                                     Item cors
C2     q_530 Continue until everything is perfect. 0.70
C1     q_124 Am exacting in my work.               0.64

$extraversion
    ItemLabel                                      Item  cors
E2-     q_901 Find it difficult to approach others. -0.78
E1-     q_712 Don't talk a lot.                     -0.72

$neuroticism
   ItemLabel                  Item cors
N1     q_952 Get angry easily.     0.80
N2     q_974 Get irritated easily. 0.78

$openness
    ItemLabel                                  Item  cors
O2-     q_316 Avoid difficult reading material. -0.65
O1      q_128 Am full of ideas.                  0.61
```

Show all items per scale

```
> lookupFromKeys(bfi.keys,dictionary=bfi.dictionary[1:2],cors=scores$item.cor)
$agree
    ItemLabel                                        Item  cors
A3     q_1206 Know how to comfort others.                   0.76
A2     q_1162 Inquire about others' well-being.             0.73
A5     q_1419 Make people feel at ease.                     0.69
A4     q_1364 Love children.                                0.65
A1-    q_146  Am indifferent to the feelings of others. -0.58

$conscientious
    ItemLabel                                    Item  cors
C4-    q_626  Do things in a half-way manner.       -0.74
C5-    q_1949 Waste my time.                        -0.72
C2     q_530  Continue until everything is perfect.  0.70
C3     q_619  Do things according to a plan.         0.66
C1     q_124  Am exacting in my work.                0.64

$extraversion
    ItemLabel                                    Item  cors
E2-    q_901  Find it difficult to approach others. -0.78
E4     q_1410 Make friends easily.                   0.74
E1-    q_712  Don't talk a lot.                     -0.72
E3     q_1205 Know how to captivate people.          0.68
E5     q_1768 Take charge.                           0.64

$neuroticism
   ItemLabel                         Item cors
N3    q_1099 Have frequent mood swings. 0.81
N1    q_952  Get angry easily.          0.80
N2    q_974  Get irritated easily.      0.78
N4    q_1479 Often feel blue.           0.71
N5    q_1505 Panic easily.              0.68
```

Another improvement

1. Wouldn't it be nice to sort the items by their absolute loadings.
2. To do this, we add another feature to `lookupFromItems`: the sort option
3. We make this default to be TRUE because this is so useful.
4. How we do this is discussed in the accompanying Rmd file

Sort by absolute value of the correlation

```
$agree
    ItemLabel                                      Item   Giant3  cors
A3     q_1206                Know how to comfort others. Cohesion  0.76
A2     q_1162           Inquire about others' well-being. Cohesion  0.73
A5     q_1419                   Make people feel at ease. Cohesion  0.69
A4     q_1364                               Love children. Cohesion  0.65
A1-     q_146 Am indifferent to the feelings of others. Cohesion -0.58

$conscientiousness
    ItemLabel                                    Item    Giant3  cors
C4-     q_626         Do things in a half-way manner. Stability -0.74
C5-    q_1949                    Waste my time. Stability -0.72
C2      q_530 Continue until everything is perfect. Stability  0.70
C3      q_619          Do things according to a plan. Stability  0.66
C1      q_124                 Am exacting in my work. Stability  0.64

$extraversion
    ItemLabel                                    Item     Giant3  cors
E2-     q_901 Find it difficult to approach others. Plasticity -0.78
E4     q_1410                  Make friends easily. Plasticity  0.74
E1-     q_712                     Don't talk a lot. Plasticity -0.72
E3     q_1205           Know how to captivate people. Plasticity  0.68
E5     q_1768                          Take charge. Plasticity  0.64

$neuroticism
   ItemLabel                        Item    Giant3 cors
N3    q_1099 Have frequent mood swings. Stability 0.81
N1     q_952          Get angry easily. Stability 0.80
N2     q_974      Get irritated easily. Stability 0.78
N4    q_1479           Often feel blue. Stability 0.71
N5    q_1505              Panic easily. Stability 0.68

$openness
```

A generic search function

R code

lookupItems

```
 lookupItems
function (content = NULL, dictionary = NULL, search = c("Item",
    "Content", "item"))
{
    location <- which(colnames(dictionary) %in% search)
    value <- grep(content, dictionary[, location])
    dictionary[value, , drop = FALSE]
}
```

1. Elaborate `lookupItems` with progressively more sophisticated options.
2. First, introduce the concept of a dictionary that has the relevant items.
3. Then extend these options to different kinds of input.

Search for a key word in a list of items

R code

```
lookupItems("life",psychTools::spi.dictionary) #find those items with
lookupItems("liberal",psychTools::spi.dictionary) #find those items wi
```

```
lookupItems("life",psychTools::spi.dictionary) #find those items with "life" in the item
       item_id                                  item item_scale resp_type   B5           L27
q_1281  q_1281          Like a leisurely lifestyle.       IPIP       reg <NA> EasyGoingness
q_2765  q_2765               Am happy with my life.       IPIP       reg <NA>     WellBeing
q_1248  q_1248           Laugh my way through life.       IPIP       reg <NA>         Humor
q_755    q_755 Enjoy examining myself and my life.       IPIP       reg <NA> Introspection
q_1052  q_1052         Have a slow pace to my life.       IPIP       reg <NA> EasyGoingness
q_1371  q_1371                             Love life.       IPIP       reg <NA>     WellBeing
>

lookupItems("liberal",psychTools::spi.dictionary) #find those items with "life" in the item
       item_id                                                 item item_scale resp_type   B5
q_1825  q_1825 Tend to vote for liberal political candidates.       IPIP       reg <NA>
               L27
```

item.lookup combines factor loadings and means

R code

```
ds <- describe(bfi)

item.lookup(f5, ds$mean,bfi.dictionary[,2,drop=FALSE])
```

```
item.lookup(f5, ds$mean,bfi.dictionary[,2,drop=FALSE])
             MR2   MR1   MR3   MR5   MR4 means                                  Item
N1          0.78  0.09  0.00 -0.12 -0.03  2.41 Get angry easily.
N2          0.76  0.03  0.01 -0.09  0.02  4.80 Get irritated easily.
N3          0.73 -0.06 -0.03  0.06  0.02  4.60 Have frequent mood swings.
N5          0.53 -0.15  0.01  0.21 -0.17  4.70 Panic easily.
N4          0.50 -0.36 -0.13  0.10  0.09  4.56 Often feel blue.
E4          0.00  0.67  0.02  0.18 -0.07  4.50 Make friends easily.
E2          0.15 -0.64 -0.02 -0.03 -0.06  4.37 Find it difficult to approach others.
E1         -0.03 -0.54  0.10 -0.07 -0.09  4.30 Don't talk a lot.
E3          0.08  0.49  0.00  0.16  0.29  2.55 Know how to captivate people.
E5          0.12  0.41  0.27  0.03  0.22  3.30 Take charge.
C2          0.16 -0.05  0.66  0.04  0.04  2.97 Continue until everything is perfect.
C4          0.18  0.06 -0.62 -0.01 -0.04  3.14 Do things in a half-way manner.
C3          0.04 -0.05  0.56  0.08 -0.06  4.00 Do things according to a plan.
C5          0.20 -0.12 -0.56  0.01  0.10  4.42 Waste my time.
C1          0.07 -0.02  0.55 -0.04  0.15  4.42 Am exacting in my work.
A2          0.01  0.07  0.08  0.63  0.03  2.93 Inquire about others' well-being.
A3          0.00  0.23  0.03  0.58  0.03  3.51 Know how to comfort others.
A1          0.20  0.19  0.07 -0.51 -0.05  3.22 Am indifferent to the feelings of others.
```

fa.lookup sorts factor loadings and shows item content

R code

```
fa.lookup(f5,bfi.dictionary[,2,drop=FALSE])
```

```
fa.lookup(f5,bfi.dictionary[,2,drop=FALSE])
            MR2   MR1   MR3   MR5   MR4   h2  com                                  Item
N1         0.78  0.09  0.00 -0.12 -0.03 0.61 1.08 Get angry easily.
N2         0.76  0.03  0.01 -0.09  0.02 0.58 1.03 Get irritated easily.
N3         0.73 -0.06 -0.03  0.06  0.02 0.56 1.03 Have frequent mood swings.
N5         0.53 -0.15  0.01  0.21 -0.17 0.37 1.70 Panic easily.
N4         0.50 -0.36 -0.13  0.10  0.09 0.48 2.15 Often feel blue.
E4         0.00  0.67  0.02  0.18 -0.07 0.55 1.17 Make friends easily.
E2         0.15 -0.64 -0.02 -0.03 -0.06 0.51 1.13 Find it difficult to approach others.
E1        -0.03 -0.54  0.10 -0.07 -0.09 0.32 1.17 Don't talk a lot.
E3         0.08  0.49  0.00  0.16  0.29 0.45 1.93 Know how to captivate people.
E5         0.12  0.41  0.27  0.03  0.22 0.39 2.58 Take charge.
C2         0.16 -0.05  0.66  0.04  0.04 0.44 1.15 Continue until everything is perfect.
C4         0.18  0.06 -0.62 -0.01 -0.04 0.46 1.20 Do things in a half-way manner.
C3         0.04 -0.05  0.56  0.08 -0.06 0.31 1.09 Do things according to a plan.
C5         0.20 -0.12 -0.56  0.01  0.10 0.43 1.43 Waste my time.
C1         0.07 -0.02  0.55 -0.04  0.15 0.33 1.21 Am exacting in my work.
A2         0.01  0.07  0.08  0.63  0.03 0.47 1.07 Inquire about others' well-being.
A3         0.00  0.23  0.03  0.58  0.03 0.49 1.31 Know how to comfort others.
```

lmCorLookup looks up and displays items from lmCor

R code

```
lc <- lmCor(x=1:25,y=28,data=bfi) #do the regressions
lmCorLookup(lc,bfi.dictionary[,2:3]) #display the coefficients
```

```
lc <- lmCor(x=1:25,y=28,data=bfi)
> lmCorLookup(lc,bfi.dictionary[,2:3])
$age
    coef                                        Item     Giant3
E4 -0.14                     Make friends easily. Plasticity
A4  0.13                           Love children.   Cohesion
A5  0.12                Make people feel at ease.   Cohesion
A1 -0.11 Am indifferent to the feelings of others.   Cohesion
N4  0.10                          Often feel blue.  Stability
E5  0.10                             Take charge. Plasticity
E2 -0.10     Find it difficult to approach others. Plasticity
C4 -0.08           Do things in a half-way manner.  Stability
E3 -0.08             Know how to captivate people. Plasticity
C2 -0.07     Continue until everything is perfect.  Stability
N3 -0.07               Have frequent mood swings.  Stability
N5 -0.06                           Panic easily.  Stability
C1  0.05                  Am exacting in my work.  Stability
O5 -0.05     Will not probe deeply into a subject. Plasticity
```

the `lookupFromKeys` function calls several other basic functions

grep Find the location of an object inside another object

sub Replace an object with another object inside a longer vector

gsub Replace each occasion of an object with another object

match returns a vector of the positions of (first) matches of its first argument in its second.

%in% returns a logical vector indicating if there is a match or not for its left operand.

order returns a permutation which rearranges its first argument into ascending or descending order, breaking ties by further arguments. sort.list does the same, using only one argument.

lookup uses match to find where x is in y (*psych*)

grep

Pattern Matching and Replacement

Description

grep, grepl, regexpr, gregexpr, regexec and gregexec search for matches to argument pattern within each element of a character vector: they differ in the format of and amount of detail in the results.

```
grep(pattern, x, ignore.case = FALSE, perl = FALSE, value = FALSE,
     fixed = FALSE, useBytes = FALSE, invert = FALSE)
```

R code

```
set.seed(42)  #so you get the same results
  x <- sample(10,20,replace=TRUE)
 x  #show the values
 grep(10,x)
```

```
set.seed(42)  #so you get the same results
>   x <- sample(10,20,replace=TRUE)
>  x  #show the values
 [1] 10 10  3  9  7  6  8  2  7  8  5  8 10  3  5 10 10  2  5  6
>  grep(10,x)
[1]  1  2 13 16 17
```

sub

```
sub(pattern, replacement, x, ignore.case = FALSE, perl = FALSE,
    fixed = FALSE, useBytes = FALSE)

gsub(pattern, replacement, x, ignore.case = FALSE, perl = FALSE,
     fixed = FALSE, useBytes = FALSE)
```

R code

```
sub(10,-999,x)
#x was a vector
x.string <- "This is a very long and convoluted sentence.
    We want to search it for all occasions of `a` and then change them
sub('a','A', x.string)  #just the first one is changed
gsub('a','A', x.string) #they are all changed
```

```
sub(10,-999,x)
 [1] "-999" "-999" "3"    "9"    "7"    "6"    "8"    "2"    "7"    "8"    "5"    "8"    "-99
[1] "This is A very long and convoluted sentence.  We want to search it for all occasions of

 [1] "This is A very long And convoluted sentence.  We wAnt to seArch it for All occAsions of
```

match and %in%

'match' match returns a vector of the positions of (first) matches of its first argument in its second.
%in% is a more intuitive interface as a binary operator, which returns a logical vector indicating if there is a match or not for its left operand.

```
match(x, table, nomatch = NA_integer_, incomparables = NULL)
x %in% table
```

R code

```
x
match(10,x)
10 %in% x
x %in% 10
```

```
 x
 [1] 10 10  3  9  7  6  8  2  7  8  5  8 10  3  5 10 10  2  5  6
> match(10,x)   #yes, 10 is in x
[1] 1
> 10 %in% x     #it is still in x
[1] TRUE
> x %in% 10   #and now we get a vector of where in x it is
 [1]  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE
```

order and sort and dfOrder

'order' returns a permutation which rearranges its first argument into ascending or descending order, breaking ties by further arguments. sort.list does the same, using only one argument.

R code

```
x
ord <- order(x)
ord.decreasing <- order(x,decreasing=TRUE)
ord
x[ord]
x[ord.decreasing]
```

```
 x
 [1] 10 10  3  9  7  6  8  2  7  8  5  8 10  3  5 10 10  2  5  6
> ord <- order(x)
> ord.decreasing <- order(x,decreasing=TRUE)
> ord
 [1]  8 18  3 14 11 15 19  6 20  5  9  7 10 12  4  1  2 13 16 17
> x[ord]
 [1]  2  2  3  3  5  5  5  6  6  7  7  8  8  8  9 10 10 10 10 10
> x[ord.decreasing]
 [1] 10 10 10 10 10  9  8  8  8  7  7  6  6  5  5  5  3  3  2  2
```

`order` of one column in a data.frame

Use `order` to give a vector of *pointers* to a data.frame. Then reorder the data.frame using these pointers.

R code

```
ds <- describe(attitude)  #find the descriptive statistics
ord <- order(ds[,"mean")  #rank order of the means
ord #show the order
ds[ord, ] #show data frame in sorted order
```

```
ds <- describe(attitude)  #find the descriptive statistics
> ord <- order(ds[,"mean"])  #rank order of the means
> ord #show the order
[1] 7 3 4 1 5 2 6
> ds[ord, ] #show data frame in sorted order
           vars  n  mean    sd median trimmed   mad min max range  skew kurtosis   se
advance       7 30 42.93 10.29   41.0   41.83  8.90  25  72    47  0.85     0.47 1.88
privileges    3 30 53.13 12.24   51.5   52.75 10.38  30  83    53  0.38    -0.41 2.23
learning      4 30 56.37 11.74   56.5   56.58 14.83  34  75    41 -0.05    -1.22 2.14
rating        1 30 64.63 12.17   65.5   65.21 10.38  40  85    45 -0.36    -0.77 2.22
raises        5 30 64.63 10.40   63.5   64.50 11.12  43  88    45  0.20    -0.60 1.90
complaints    2 30 66.60 13.31   65.0   67.08 14.83  37  90    53 -0.22    -0.68 2.43
```

sort 2 or more columns using `dfOrder`

`order` just works on vectors, but `dfOrder` will sort data.frames on multiple columns

R code

```
ds <- describe(ability)
ds.ordered <- dfOrder(ds,cs(n, mean))
ds.ordered
```

```
ds <- describe(ability)
ds.ordered <- dfOrder(ds,cs(n, mean))
ds.ordered
          vars    n mean   sd median trimmed mad min max range  skew kurtosis   se
letter.58    8 1438 0.47 0.50      0    0.46   0   0   1     1  0.12    -1.99 0.01
letter.33    6 1438 0.61 0.49      1    0.63   0   0   1     1 -0.43    -1.82 0.01
reason.17    3 1440 0.74 0.44      1    0.80   0   0   1     1 -1.08    -0.84 0.01
letter.7     5 1441 0.63 0.48      1    0.67   0   0   1     1 -0.56    -1.69 0.01
reason.4     1 1442 0.68 0.47      1    0.72   0   0   1     1 -0.75    -1.44 0.01
letter.34    7 1455 0.64 0.48      1    0.68   0   0   1     1 -0.59    -1.65 0.01
rotate.3    13 1456 0.20 0.40      0    0.13   0   0   1     1  1.48     0.19 0.01
rotate.6    15 1456 0.31 0.46      0    0.27   0   0   1     1  0.80    -1.35 0.01
reason.19    4 1456 0.64 0.48      1    0.68   0   0   1     1 -0.60    -1.64 0.01
matrix.45    9 1458 0.55 0.50      1    0.56   0   0   1     1 -0.20    -1.96 0.01
matrix.55   12 1459 0.39 0.49      0    0.36   0   0   1     1  0.45    -1.80 0.01
rotate.8    16 1460 0.19 0.39      0    0.12   0   0   1     1  1.55     0.41 0.01
rotate.4    14 1460 0.22 0.42      0    0.15   0   0   1     1  1.34    -0.21 0.01
reason.16    2 1463 0.73 0.45      1    0.78   0   0   1     1 -1.02    -0.96 0.01
matrix.47   11 1465 0.64 0.48      1    0.67   0   0   1     1 -0.57    -1.67 0.01
matrix.46   10 1470 0.57 0.50      1    0.59   0   0   1     1 -0.28    -1.92 0.01
```

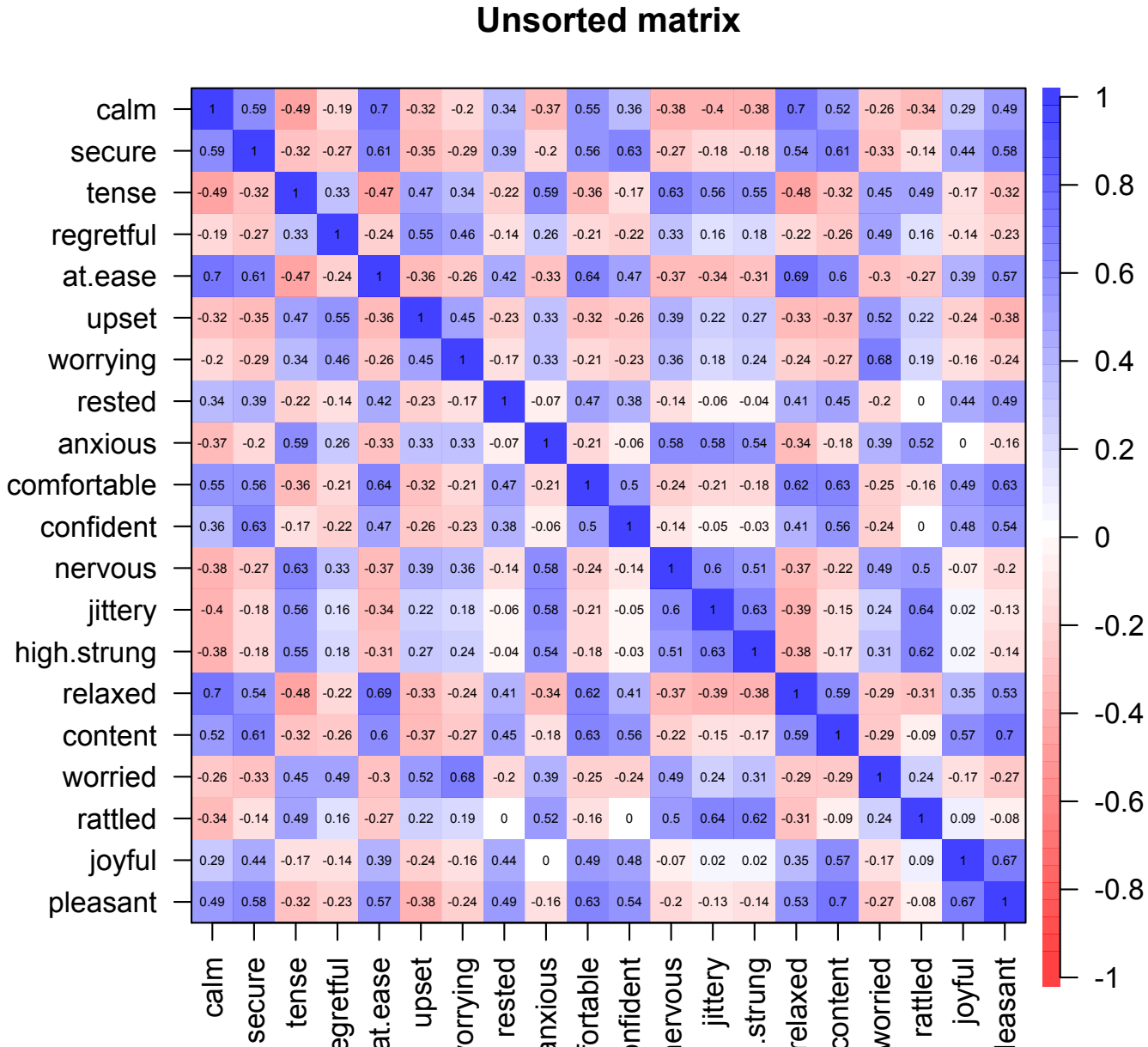
matSort to sort a matrix by some specified order

R code

```
R <- lowerCor(sai[4:23])
```

```
             calm  secur tense rgrtf at.es upset wrryn restd anxis cmfrt cnfdn nervs jttry hgh
calm         1.00
secure       0.59  1.00
tense       -0.49 -0.32  1.00
regretful   -0.19 -0.27  0.33  1.00
at.ease      0.70  0.61 -0.47 -0.24  1.00
upset       -0.32 -0.35  0.47  0.55 -0.36  1.00
worrying    -0.20 -0.29  0.34  0.46 -0.26  0.45  1.00
rested       0.34  0.39 -0.22 -0.14  0.42 -0.23 -0.17  1.00
anxious     -0.37 -0.20  0.59  0.26 -0.33  0.33  0.33 -0.07  1.00
comfortable  0.55  0.56 -0.36 -0.21  0.64 -0.32 -0.21  0.47 -0.21  1.00
confident    0.36  0.63 -0.17 -0.22  0.47 -0.26 -0.23  0.38 -0.06  0.50  1.00
nervous     -0.38 -0.27  0.63  0.33 -0.37  0.39  0.36 -0.14  0.58 -0.24 -0.14  1.00
jittery     -0.40 -0.18  0.56  0.16 -0.34  0.22  0.18 -0.06  0.58 -0.21 -0.05  0.60  1.00
high.strung -0.38 -0.18  0.55  0.18 -0.31  0.27  0.24 -0.04  0.54 -0.18 -0.03  0.51  0.63  1.
relaxed      0.70  0.54 -0.48 -0.22  0.69 -0.33 -0.24  0.41 -0.34  0.62  0.41 -0.37 -0.39 -0.
content      0.52  0.61 -0.32 -0.26  0.60 -0.37 -0.27  0.45 -0.18  0.63  0.56 -0.22 -0.15 -0.
worried     -0.26 -0.33  0.45  0.49 -0.30  0.52  0.68 -0.20  0.39 -0.25 -0.24  0.49  0.24  0.
rattled     -0.34 -0.14  0.49  0.16 -0.27  0.22  0.19  0.00  0.52 -0.16  0.00  0.50  0.64  0.
joyful       0.29  0.44 -0.17 -0.14  0.39 -0.24 -0.16  0.44  0.00  0.49  0.48 -0.07  0.02  0.
pleasant     0.49  0.58 -0.32 -0.23  0.57 -0.38 -0.24  0.49 -0.16  0.63  0.54 -0.20 -0.13 -0.
```

Corplot of the unsorted matrix



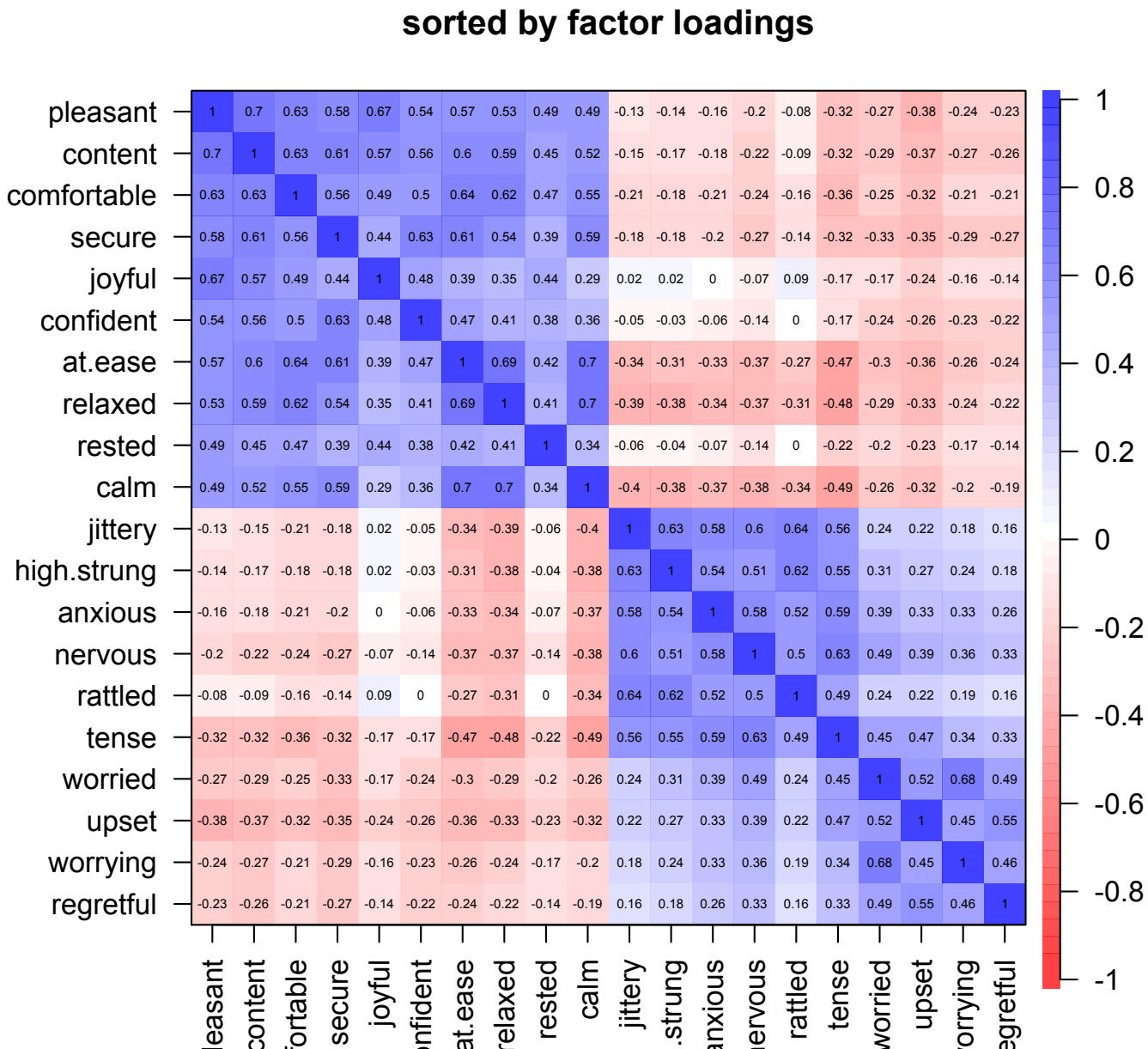
Sorted matrix

R code

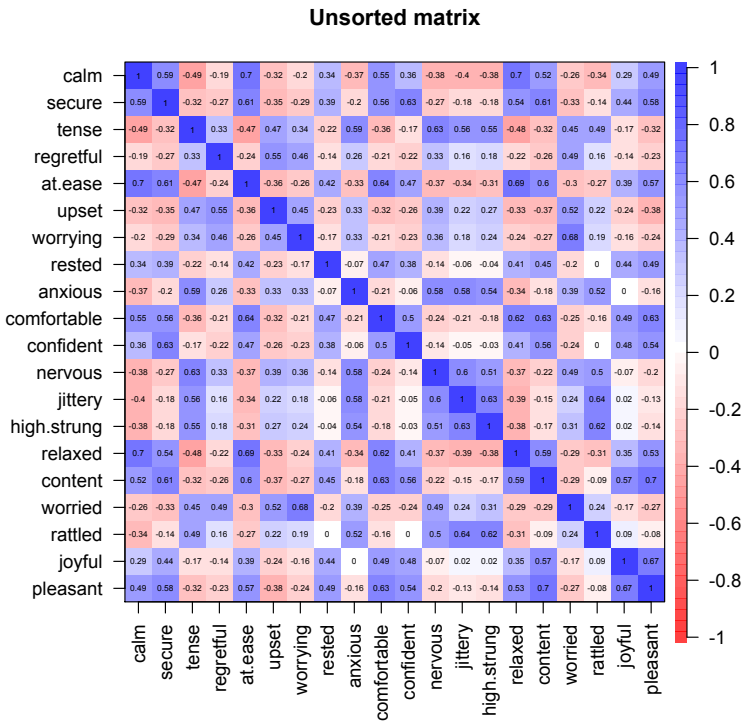
```
f2 <- fa(R,2); Rs <- matSort(R,f2);     lowerMat(Rs)
```

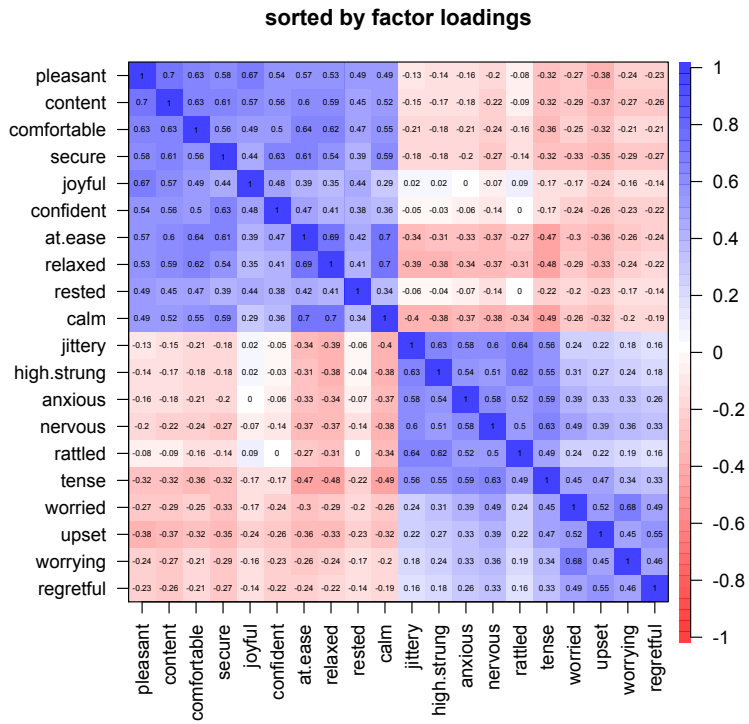
```
lowerMat(Rs)
            at.es relxd calm  cntnt cmfrt secur plsnt tense nervs upset worrd cnfdn anxis jtt
            plsnt cntnt cmfrt secur joyfl cnfdn at.es relxd restd calm  jttry hgh.s anxis ner
pleasant     1.00
content      0.70  1.00
comfortable  0.63  0.63  1.00
secure       0.58  0.61  0.56  1.00
joyful       0.67  0.57  0.49  0.44  1.00
confident    0.54  0.56  0.50  0.63  0.48  1.00
at.ease      0.57  0.60  0.64  0.61  0.39  0.47  1.00
relaxed      0.53  0.59  0.62  0.54  0.35  0.41  0.69  1.00
rested       0.49  0.45  0.47  0.39  0.44  0.38  0.42  0.41  1.00
calm         0.49  0.52  0.55  0.59  0.29  0.36  0.70  0.70  0.34  1.00
jittery     -0.13 -0.15 -0.21 -0.18  0.02 -0.05 -0.34 -0.39 -0.06 -0.40  1.00
high.strung -0.14 -0.17 -0.18 -0.18  0.02 -0.03 -0.31 -0.38 -0.04 -0.38  0.63  1.00
anxious     -0.16 -0.18 -0.21 -0.20  0.00 -0.06 -0.33 -0.34 -0.07 -0.37  0.58  0.54  1.00
nervous     -0.20 -0.22 -0.24 -0.27 -0.07 -0.14 -0.37 -0.37 -0.14 -0.38  0.60  0.51  0.58  1.
rattled     -0.08 -0.09 -0.16 -0.14  0.09  0.00 -0.27 -0.31  0.00 -0.34  0.64  0.62  0.52  0.
tense       -0.32 -0.32 -0.36 -0.32 -0.17 -0.17 -0.47 -0.48 -0.22 -0.49  0.56  0.55  0.59  0.
worried     -0.27 -0.29 -0.25 -0.33 -0.17 -0.24 -0.30 -0.29 -0.20 -0.26  0.24  0.31  0.39  0.
upset       -0.38 -0.37 -0.32 -0.35 -0.24 -0.26 -0.36 -0.33 -0.23 -0.32  0.22  0.27  0.33  0.
worrying    -0.24 -0.27 -0.21 -0.29 -0.16 -0.23 -0.26 -0.24 -0.17 -0.20  0.18  0.24  0.33  0.
regretful   -0.23 -0.26 -0.21 -0.27 -0.14 -0.22 -0.24 -0.22 -0.14 -0.19  0.16  0.18  0.26  0.
```

Corplot of the sorted matrix



Compare these two for clarity





Programming is stringing together functions

1. Break down the problem into parts
2. Use a skeleton style

R code

```
myfunction <- function(a parameter list) {
preliminary code to go here
main part here
wrap up here
}
```

3. Perhaps write the guts of the function first. What are you actually trying to do?
   - What functions already exist to do part of what you want to do?
   - Can you use them directly, or do you need to modify them?
4. Then add the preliminary code with various default values, check for good input
5. Figure out how to return useful information.
   - Do you want to report a single object
   - Or return a list of objects
6. Run, debug, run again.

Checking the results

1. Do the results meet a ‘sanity check‘.
2. Do the results match what you want.
3. Are they right? How can you tell?