Selecting and filtering

|  |
| --- |
| **Data TRansformation: Selecting & Filtering** |
| Selecting and filtering= assessing data (elements or subsequences) of an object | |
| **1. Vectors:**  **Summary:**  x[2], x[-2], x[c(2,3,4)], x[x\*2>10], x[‘a‘], x[[‘a‘]]  **1.1 Subset to a new vector** 1) Selecting with **integer Index/Indices:** <Vektorname>[<Indizes>]   * for example:     „gebe den Wert des Vektors wieder, für den gilt, dass er an 2. Stelle im Vektor auftritt“  negative index excludes the respective value   * define “Index”-variable to use for filtering:     **2) Filtering with rules/relational operators:**  <Vectorname>[<Vectorname><Rule>]   * for example: * relational expression results logical vector, that R then uses to filter out values for which vector has value TRUE * **beachte: innerhalb der Regeln muss nochmals auf den Vektor bezuggenommen werden**; ansonsten ist die Regel nicht eindeutig definiert   **3) Select with boolean variables/logical vectors:**   * for example:     **4) Select with character strings/vector: <name of vector>[”<name of element>”]**   * for example:       **1.2 Assess values only:**  [[„<name of element>“]]   * for example:   **# note that [[]] can only extract one value of a vector element at a time**    **1.3 Definieren des subset-/filter-Vektors als eigenständiger, neuer Vektor:**   * mit [] (behält Namen einzelner Elemente des Vektors bei):      * mit [[]] (Namen werden nicht beibehalten) and only one element:     **2. Subsetting matrices:**  **Summary:**  m1[2,3], m1[1,], m1[,1], m1[1:2,], m1[,1:2], m1[,3,drop=false], m1[1],m1[’r1’]  **Subsettting to a vector:**  **1) Select with 2-dimensional Index:** <Matrixname>[<Row Index>,<Column Index>]  for example    # access element in first row and second column     * same as w/ vectors: neg. numbers exclude rows or columns respectively     note: if one wants to keep the data structure *matrix* when accessing a complete column, rather than getting a vector:  use > drop = FALSE  for example:      **Subsetting to a matrix:** for example    # access elements in row 1-2 and column 2-3    **2) select with one-dimensional Index:** <Matrixname>[<Index>]   * as matrix is a collection of vectors, we can also use one-dimensional accessor * resulting subset is always a vector * gem. der Vektorschreibweise, setzt **R bei Matrizen Indizes spaltenweise**       **3) filter with rules/relational operators:** <Matrixname>[<Regel>]     * note: auch hier gilt, dass innerhalb der Regel nochmals auf die Matrix Bezug genommen werden muss   **4) select with character strings/vector for named matrix (e.g. dimnames):**    **5) extracting values:** <Matrixname>[[<Row Index>, <Column Index>]]    **3. array:** <array>[<row index>,<column index>, <layer index>]   * **principle for subsetting a matrix also apply to subsetting an array**   for example:    **1) Select with multidimensional Index:**  only specifying row index: yields all columns and layers of a row    **- note: R directly coerces to matrix**  only specifying column index: yields all rows and layers of that column  only specifying layer index: yields all rows and columns in that layer    specifying with multiple indizes:    **2) select w/ one-dimensional index:**   * treats array as vector       **3) select with rules:**    **4) select with character strings for named array**     * **note:** when selecting w/ character strings we have to define all dimensions; otherwise, we will get following error     **4. lists:**   * list access is overall **very similar to vector access**   **3.1 Subset to a new list:**  **1) select with integer Index/Indices:** <name of list>[<Index des Listenelementes>]    # access 1st element from list    # access 1st-2nd element from list    **2) select with boolean variables/logical vectors:**    **3) select with character strings/vectors** [<name of list element>]      **3.1 Extracting values:  🡪 getting a vector**  **Using $**  > <name of list>$<name of list element>[<Index/Rule>]  for example:    **# note that $ can only extract one list element at a time**  **also: no “ ” needed**      **Extracting with rules**    🡪 again: name of list has to be mentioned in rule as well  **Using** [[“<name of list element>”]]  **# note that [[]] can only extract one list element at a time**    **Using** [[<name of list element/index list element>]] [[<Index>]] 🡪 the first double bracket defines index of list element; **the second defines the index within the element**  **# note that double-[[]] can only extract one element of a list element at a time (see error message)**    Creating new vector with extracted values:  For example:    **4. Data frames**  **4.1) Subset a data frame as a list**   * note: subsetting data frames by treating them as list, does not support row-selection (see data frames as a matrix) * the same principles **as for lists** apply:     **1) select with integer Index/Indices:**    # access ‚list elements‘ 1-2      **2) select with character strings/vectors**     * **note:** even if we only choose one columne, subsetting a dataframe as a list,  still results in a dataframe not in a vector   **3) select with boolean variables**    **4.2) Subset a dataframe as a matrix**   * supports both, column and row selection * the same principles **as with matrix selection** apply   **1) select with two-dimensional integer Index/Indices** <name of df>[<Row Index>,<Column Index>]      **2) character strings/vectors (i.e. row- and columnnames)**       * note: as with matrices, we will get a vector, not a data.frame anymore, if we choose only one column and subset data.frame as matrix using two-dimensional integer index * keep the result as a data frame when assessing only one column:   > <name of df> [<# of rows>,] [<name of column>] (dadurch bleiben 2 dimensionen erhalten) s   * alternative: drop=false> <name of df> [<number of rows>,”<name of column>”, drop = FALSE] * z.B.:   **3) filter with rules/relational operators**  > ‘name data frame’[name data frame‘$‘Spaltenname‘*Regel***,]**  **🡪** relational expression results in logical vector; the latter is then used to filter the data frame 🡪 the comma at the end indicates that all columns shall be included in filtered data frame  🡪 this command DOES NOT work without the comma  **🡪** again name of dataframe has to be mentioned in rule again  another example w/ explicetly stated columns:    **4.3 Extract values w/ Index or Character Strings:**    **# note that $ and [[]] can only extract one data frame element at a time**  #> <name data frame>$<Spaltenname>   * **note: $ uses partial matching; hence one could type $i and R would autocomplete it to id, if (!) there does not exist any other variable starting with i** | |