Combining Data Frames

- Merge two data frames by common columns or row names, or do other versions of database *join* operations.
- General form:

  merge(*frame1*, *frame2*, *all* = FALSE)

- *frame*n – data frames being joined
- *all*
  - FALSE – Only rows with matching data from data frames are included
  - TRUE – nonmatching rows added to output

Combining Data Frames

Country	CPI
New Zealand	9.5
Denmark	9.4
Finland	9.4
Sweden	9.3

Country	Capital
Austria	Vienna
Denmark	Copenhagen
Finland	Helsinki
Iceland	Reykjavik

Example:

merge(cpis, capitals, all=TRUE)

Strings/ Character Arrays

- Character arrays are vectors of strings
- Use single (') or double (") quotes to mark strings, but don’t mix:

```
x <- 'good'
y <- "no'
z <- "it's working"
```

String Functions

- Create a vector of character strings
  s <- c('apple','bee','cars','danish','egg')
- Get the number of characters in each string
  nchar(s)
- Convert all letters to upper case
  toupper(s)

String Functions

- Extract or replace substrings in a character vector.
- General forms:

  substr(*x, start, stop*)

  substr(*x, start, stop*) <- *value*

  - *x* – a character vector
  - *start* – first element of substring
  - *stop* – last element of substring
  - *value* – character vector to replace original values

Substring Function

- Examples:
- Extract the first to third characters
  substr(s,1,3)
- Replace first and second characters
  substr(s,1,2) <- ‘BU’

String Functions

- Replace (substitute) specific values
- sub – replaces first occurrence in string
- gsub – replaces all occurrences (globally)
- General form:

  sub(*pattern, replacement, x*)

  - *pattern* – string to be replaced
  - *replacement* – new values for matched pattern
  - *x* – character vector where matches are sought

Substitute Functions

- Examples:
- Replace first 'e' with '_ '
  sub('e', '_', s)
- Globally replace every 'e' with '_ '
  gsub('e', '_', s)

Working with Text

- Find characters in a string
- grep – return indices or values of strings where matches are found
- grepl – (logic) return TRUE or FALSE to indicate where matches are found

General Form of grep Functions

grep(*pattern*, *x*, *ignore.case* = FALSE, *value* = FALSE)

grepl(*pattern*, *x*, *ignore.case* = FALSE)

- *pattern* – what you are looking for (add ^ to search only at beginning)
- *x* – vector being searched
- *ignore.case* – control case sensitivity
- *value* – returns indices of matches instead of actual value

Examples of grep Functions

- Search s for instances of 'e'
  grep('e', s)
- Return list of TRUE/FALSE instead of indices
  grepl('e', s)
- Search using a regular expression
  grep('^e', s)
- Return values instead of indices
  grep("Z", cpi$Country, value=TRUE)

Reordering Values

- sort – returns actual values in desired order
- order – returns vector of indices in new order
- order is most useful when working with tabular data like data frames

General Form of sort/order

order(..., *na.last* = TRUE, *decreasing* = FALSE)

- … a sequence of numeric, complex, character or logical vectors, all of the same length, or a classed **R** object.
- *na.last*= – where to put missing values (NA removes them)
- *decreasing*= – increasing order by default

Reordering Values - Examples

- Show country names in ascending order
  sort(cpi$Country)
- Return index values for descending order
  order(cpi$Country, decreasing=TRUE)
- Use ordered index values with other data
  cpi[order(cpi$AvgCPI),c(1,12)]

Text Output Functions

- Functions to concatenate vectors
- General Forms:

cat(... , *file* = "", *sep* = " ")

paste(..., *sep* = " ", *collapse* = NULL)

  - … - R objects to be concatenated
  - *file* – path and name of optional output file (cat)
  - *sep* – character string to separate elements or terms
  - *collapse* – optional string to separate results (paste)

Text Output Functions

- Blackslash characters allow you to generate control characters, importantly:
  - newline: \n
  - tab: \t
  - Example: cat ("5\t9\n\n")

Comparison of cat and paste

cat

- converts its arguments to character vectors
- concatenates them to a single character vector
- appends sep= string(s) to each element and then outputs them
- No linefeeds are output unless requested by "\n“
- writes to file if specified

paste

- vectors are concatenated term-by-term
- recycles as needed
- terms separated by sep
- elements separated by collapse value
- may be used for assignment
- often combined with cat

cat/paste Examples

- cat

```
cat(Country, Capital, sep=',')
Austria,Denmark,Finland,Iceland,Vienna,Cope
nhagen,Helsinki,Reykjavik
```

- paste

```
paste(Country, Capital, sep=',')
[1] "Austria,Vienna"
"Denmark,Copenhagen" "Finland,Helsinki"
[4] "Iceland,Reykjavik"
```

Importing Data

- See help on read.table for more info
- R recommends converting Excel, etc. to delimited text if possible
- Know thy data
- Use \\ or / rather than \ in Windows path
- Remember to use the entire file path and not just the file name.
- read.csv – for comma separated data
- read.delim – for tab delimited data

General Form of read Function

read.csv(*file*, *header* = TRUE, *sep* = ",", *quote*="\"", *dec*=".")

- *file* – file path or URL to data
- *header* – does first line contain field names
- *sep* – field separator character
- *quote* – set of quoting characters
- *dec* – character used for decimal points

Example of read function

- Read csv-like file that uses pipe as delimiter and * for missing data
  read.csv("c:/data/pipedata.csv", sep="|", na.strings="*")