Data Management in R

Alena van Bömmel
Matthew Huska
Slides from Uwe Ziegenhagen

http://www.uweziegenhagen.de
Agenda

- Data frames
- Reading and Writing Data
- Summary Statistics
Data Frames

Remember:
- matrices can only have one data type
- data frames: can have a different data type in each column
Data Frame Example

```r
a <- c(10, 20, 15, 43, 76, 41, 25, 46)  # numeric
# Factor 'sex'
b <- factor(c("m", "f", "m", "f", "m", "f", "m", "f"))
# siblings, numeric
c <- c(2, 5, 8, 3, 6, 1, 5, 6)
myframe <- data.frame(a, b, c)
myframe
colnames(myframe) <- c("Age", "Sex", "Siblings")
```
Factor variables

- Factor variables: categorical variables (numeric or string)
- Advantages:
  - implemented correctly in statistical modeling
  - very useful in many different types of graphics
  - correct number of degrees of freedom
Adressing Components

```r
1 myframe [,1]
2 myframe [,"Age"]
3 myframe$Age
4 myframe[3,3] <- 2 # change value
5 myframe [,,-2] # all vars except 2nd
```
Subsetting Data Frames

```r
> subset(myframe, Age > 30) # 4 entries
   Age Sex Siblings
 4 43 f   3
 5 76 m   6
 6 41 f   1
 8 46 f   6

> mean(subset(myframe, Sex == "m", Age))
   Age
31.5

# Males over 20
> subset(myframe, Sex == "m" & Age > 20)
   Age Sex Siblings
 5 76 m   6
 7 25 m   5
```
Data Frames - Column labels

```r
> myframe <- cbind(myframe, "Income (USD)" =
c(1700, 2100, 2300, 2050, 2800, 1450, 3400, 2000))
> names(myframe)[names(myframe) == "Income (USD)"] <- "IncomeUSD"
```
### Search and Replace

```r
> grepl("In", names(myframe))
[1] FALSE FALSE FALSE TRUE
> names(myframe) <- sub("In","Out",names(myframe))  # first match replacement
> names(myframe) <- gsub("In","Out",names(myframe))  # all matches replacement
> myframe
Age   Sex Siblings Outcome
1 10    m      2      1700
2 20    f      5      2100
```

Data Management in R
Sorting and Deleting

```r
> myframe[order(myframe$Age),]
   Age Sex Siblings
1   10  m    2
2   15  m    8
3   20  f    5
4   25  m    5
5   41  f    1
6   43  f    3
...
> myframe$Age <- NULL
> myframe
   Sex Siblings
1   m    2
2   f    5
3   m    8
...
```

Data Management in R
Deleting and Sorting

First sort by Sex, then if there are any ties sort them by Age:

```r
> myframe[order(myframe$Sex, partial = myframe$Age),]
Age  Sex Siblings Outcome USD
2    20  f      5      2100
6    41  f      1      1450
4    43  f      3      2050
8    46  f      6      2000
1    10  m      2      1700
3    15  m      8      2300
7    25  m      5      3400
5    76  m      6      2800
```
Data Management

- Sources of data:
  - comma-separated format (CSV), Note: German Excel separates columns using ";;"
  - tab-separated format (TXT, DAT, TSV)
  - experiment-specific types (CEL, FASTA, bam, bed)

- R has 100 built-in data sets: `objects(package: datasets)`
- many packages include their own sample data sets
Reading & Writing Data

```r
> data <- read.table("filename", header=TRUE, 
    as.is=TRUE)
# guesses the type of variable for each column
# header with column names is available
> colnames(data)  # variable names
> str(data)  # show structure of dataframe
> head(data)  # show the first few rows
> tail(data)  # show the last few rows
```

Note: If you don’t specify `as.is=TRUE`, then strings in the input file will always be interpreted as factors. This can lead to some unexpected bugs if it isn’t what you mean to do.
Reading & Writing Data

If you don’t specify the right separator it will read whole line as one column

```r
> data <- read.table("file", sep=" "")
> dim(data)
[1] 50 1
> data <- read.table("file", sep="\t")
> dim(data)
[1] 50 5
```
Be careful when reading output from Excel!

```r
# reading CSV
# decimal sep '.', column sep ','
data <- read.csv("file") #
# decimal sep ',', column sep ';'
data2 <- read.csv2("file")
# direct import from Excel
data <- read.table(file="clipboard")
# tab-delimited
datat <- read.delim("file")
```
Reading & Writing Data

```r
x <- read.csv("example.csv")
dim(x)
names(x)
x

# write to file
write.csv(x, file="test.csv")
```
Summary statistics

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Example Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>mean(x)</code></td>
<td>mean</td>
<td><code>mean(x)</code></td>
</tr>
<tr>
<td><code>median(x)</code></td>
<td>median</td>
<td><code>median(x)</code></td>
</tr>
<tr>
<td><code>var(x)</code></td>
<td>sample variance</td>
<td><code>var(x)</code></td>
</tr>
<tr>
<td><code>sd(x)</code></td>
<td>sample standard deviation</td>
<td><code>sd(x)</code></td>
</tr>
<tr>
<td><code>cov(y)</code></td>
<td>covariance of matrix y</td>
<td><code>cov(y)</code></td>
</tr>
<tr>
<td><code>quantile(x,p)</code></td>
<td>sample quantile</td>
<td><code>quantile(x,p)</code></td>
</tr>
<tr>
<td><code>min(x)</code></td>
<td>minimum of x</td>
<td><code>min(x)</code></td>
</tr>
<tr>
<td><code>max(x)</code></td>
<td>maximum of x</td>
<td><code>max(x)</code></td>
</tr>
<tr>
<td><code>range()</code></td>
<td>range of x</td>
<td><code>range()</code></td>
</tr>
<tr>
<td><code>skewness(x)</code></td>
<td>skewness</td>
<td><code>skewness(x)</code></td>
</tr>
<tr>
<td><code>kurtosis(x)</code></td>
<td>kurtosis</td>
<td><code>kurtosis(x)</code></td>
</tr>
</tbody>
</table>

* can remove missing values (NA’s) using parameter `na.rm=T`