

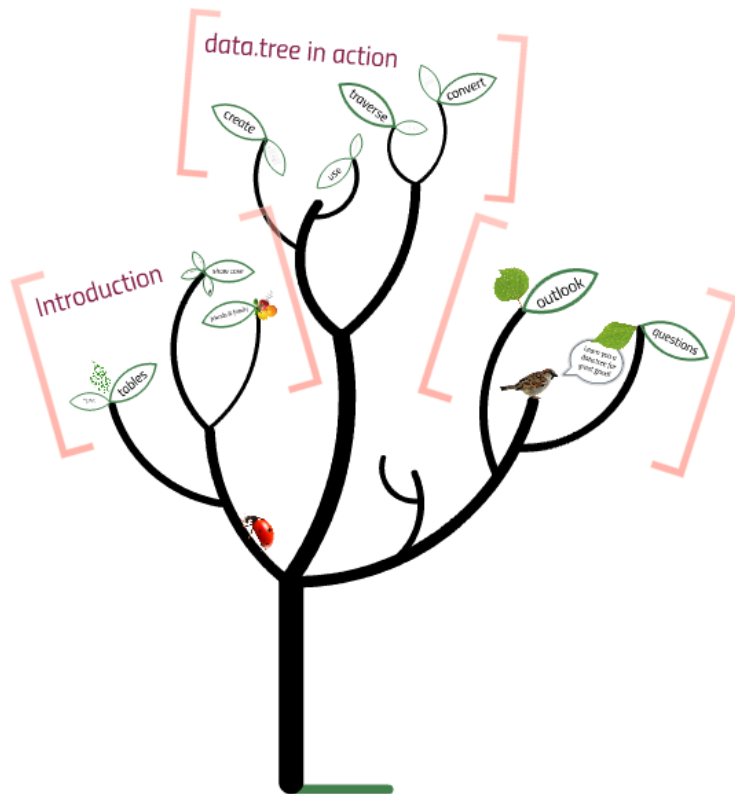
data.tree

A better way to manage
hierarchical data

useR! 2015

<http://github.com/gluc/useR15>

christoph.glur@ipub.com



data.tree

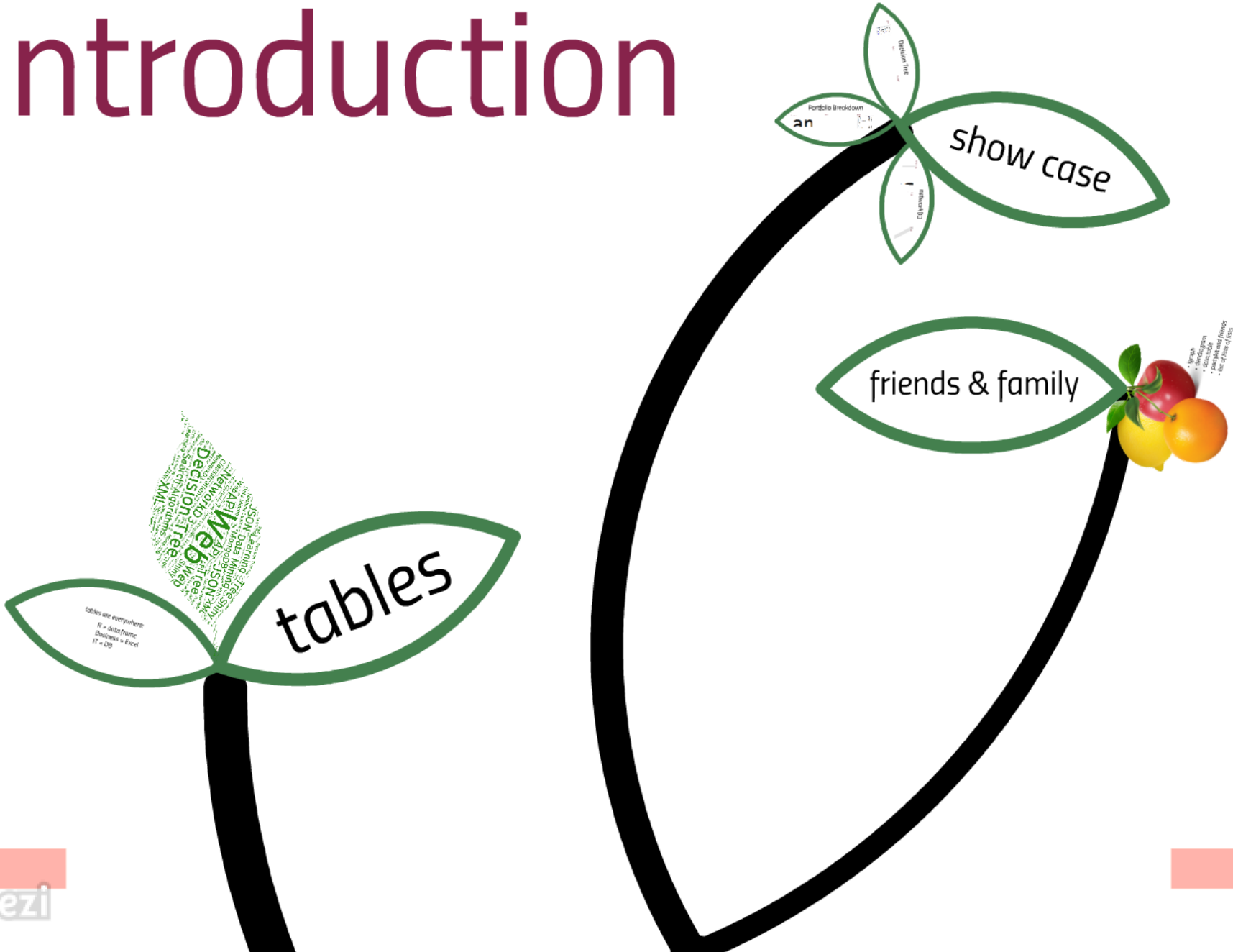
A better way to manage
hierarchical data

useR! 2015

<http://github.com/gluc/useR15>

christoph.glur@ipub.com

Introduction



tables are everywhere:

R = data.frame

Business = Excel

IT = DB





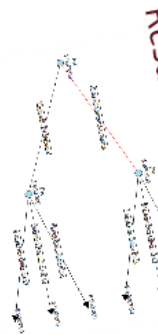
...where:

show case

Decision Tree

Portfolio Breakdown

networkD3



Portfolio Breakdown

Data in csv:

Code	Name	Asset Class	Weight	Asset Class	Asset Class	Asset Class	Asset Class	Asset Class
1	Equity	Equity	100.00%					
2	Domestic	Domestic	80.00%					
3	International	International	20.00%					
4	US Large Cap	US Large Cap	50.00%					
5	US Mid Cap	US Mid Cap	20.00%					
6	US Small Cap	US Small Cap	10.00%					
7	International Large Cap	International Large Cap	10.00%					
8	International Mid Cap	International Mid Cap	5.00%					
9	International Small Cap	International Small Cap	5.00%					
10	Fixed Income	Fixed Income	20.00%					
11	Government	Government	10.00%					
12	Corporate	Corporate	8.00%					
13	High Yield	High Yield	2.00%					
14	Real Estate	Real Estate	10.00%					
15	Commodities	Commodities	2.00%					
16	Alternatives	Alternatives	10.00%					
17	Hedge	Hedge	10.00%					
18	Private Equity	Private Equity	10.00%					
19	Private Debt	Private Debt	10.00%					
20	Impact	Impact	10.00%					
21	ESG	ESG	10.00%					
22	Climate	Climate	10.00%					
23	Human Rights	Human Rights	10.00%					
24	Social	Social	10.00%					
25	Environment	Environment	10.00%					

Code:


```

1 # Import libraries
2 import pandas as pd
3
4 # Read the CSV file
5 csv_data = pd.read_csv('portfolio_data.csv')
6
7 # Filter the data for Equity
8 equity_data = csv_data[csv_data['Asset Class'] == 'Equity']
9
10 # Filter the data for Fixed Income
11 fixed_income_data = csv_data[csv_data['Asset Class'] == 'Fixed Income']
12
13 # Filter the data for Real Estate
14 real_estate_data = csv_data[csv_data['Asset Class'] == 'Real Estate']
15
16 # Filter the data for Commodities
17 commodities_data = csv_data[csv_data['Asset Class'] == 'Commodities']
18
19 # Filter the data for Alternatives
20 alternatives_data = csv_data[csv_data['Asset Class'] == 'Alternatives']
21
22 # Filter the data for Hedge
23 hedge_data = csv_data[csv_data['Asset Class'] == 'Hedge']
24
25 # Filter the data for Private Equity
26 private_equity_data = csv_data[csv_data['Asset Class'] == 'Private Equity']
27
28 # Filter the data for Private Debt
29 private_debt_data = csv_data[csv_data['Asset Class'] == 'Private Debt']
30
31 # Filter the data for Impact
32 impact_data = csv_data[csv_data['Asset Class'] == 'Impact']
33
34 # Filter the data for ESG
35 esg_data = csv_data[csv_data['Asset Class'] == 'ESG']
36
37 # Filter the data for Climate
38 climate_data = csv_data[csv_data['Asset Class'] == 'Climate']
39
40 # Filter the data for Human Rights
41 human_rights_data = csv_data[csv_data['Asset Class'] == 'Human Rights']
42
43 # Filter the data for Social
44 social_data = csv_data[csv_data['Asset Class'] == 'Social']
45
46 # Filter the data for Environment
47 environment_data = csv_data[csv_data['Asset Class'] == 'Environment']
48
49 # Print the filtered data
50 print(equity_data)
51 print(fixed_income_data)
52 print(real_estate_data)
53 print(commodities_data)
54 print(alternatives_data)
55 print(hedge_data)
56 print(private_equity_data)
57 print(private_debt_data)
58 print(impact_data)
59 print(esg_data)
60 print(climate_data)
61 print(human_rights_data)
62 print(social_data)
63 print(environment_data)
  
```

Result:

Code	Name	Weight	Asset Class	Asset Class	Asset Class	Asset Class	Asset Class
1	portfolio	100.00%					
2	Equity	100.00%					
3	Domestic	80.00%					
4	International	20.00%					
5	US Large Cap	50.00%					
6	US Mid Cap	20.00%					
7	US Small Cap	10.00%					
8	International Large Cap	10.00%					
9	International Mid Cap	5.00%					
10	International Small Cap	5.00%					
11	Fixed Income	20.00%					
12	Government	10.00%					
13	Corporate	8.00%					
14	High Yield	2.00%					
15	Real Estate	10.00%					
16	Commodities	2.00%					
17	Alternatives	10.00%					
18	Hedge	10.00%					
19	Private Equity	10.00%					
20	Private Debt	10.00%					
21	Impact	10.00%					
22	ESG	10.00%					
23	Climate	10.00%					
24	Human Rights	10.00%					
25	Social	10.00%					
26	Environment	10.00%					

Data in csv:

	A	B	C	D	E	F	G	H	I	J
1	ISIN	Name	Ccy	Type	Duration	Weight	AssetCategory	AssetClass	SubAssetClass	
2	LI0015327682	LGT Money Mar	CHF	Fund		0.03	Cash	CHF		
3	LI0214880598	CS (Lie)  ney	EUR	Fund		0.06	Cash	EUR		
4	LI0214880689	CS (Lie) Money	USD	Fund		0.02	Cash	USD		
5	LU0243957825	Invesco Euro Co	EUR	Fund	5.1	0.12	Fixed Income	EUR	Sovereign and Corporate Bonds	
6	LU0408877412	JPM Euro Gov SI	EUR	Fund	2.45	0.065	Fixed Income	EUR	Sovereign and Corporate Bonds	
7	LU0376989207	Aberdeen Globa	EUR	Fund	6.8	0.03	Fixed Income	EUR	Emerging Markets Bonds	
8	GB00B42R2118	Threadneedle E	EUR	Fund	3.4	0.045	Fixed Income	EUR	High Yield Bonds	
9	LU0292585030	AXA IM FIIS US S	USD	Fund	1.6	0.025	Fixed Income	USD	High Yield Bonds	
10	CH0011037469	Syngenta AG	CHF	Stock		0.01	Equities	Switzerland		
11	DE0008490145	DWS Zurich Inve	EUR	Fund		0.05	Equities	Switzerland		
12	NL0000303600	ING Grope NV	EUR	Stock		0.01	Equities	Euroland		
13	IE00B60SWX25	Source EURO ST	EUR	ETF		0.08	Equities	Euroland		
14	FR0000120271	TOTAL	EUR	Stock		0.014	Equities	Euroland		
15	DE0008404005	Allianz SE	EUR	Stock		0.013	Equities	Euroland		
16	IT0000072618	Intesa Sanpaolo	EUR	Stock		0.01	Equities	Euroland		
17	BE0003793107	Anheuser-Busch	EUR	Stock		0.018	Equities	Euroland		
18	US4581401001	Intel Corp.	USD	Stock		0.01	Equities	US		
19	US0378331005	Apple Corp	USD	Stock		0.03	Equities	US		
20	US4370761029	Home Depot Inc	USD	Stock		0.015	Equities	US		
21	US5949181045	Microsoft Corp.	USD	Stock		0.014	Equities	US		
22	US7427181091	Procter & Gamb	USD	Stock		0.012	Equities	US		

Code:

```
1 library(data.tree)
2
3 #read from file
4 pfodf <- read.csv('../userR15/data/portfolio.csv', stringsAsFactors = FALSE)
5 pfodf
6
7 #convert to data.tree
8 pfodf$pathString <- paste("portfolio", pfodf$AssetCategory, pfodf$AssetClass, pfodf$SubAssetClass, pfodf$ISIN, sep = "/")
9 pfo <- as.Node[ pfodf ]
10
11 #Calculate breakdown
12 pfo$Get(Aggregate, "weight", sum, assign = "weight")
13 pfo$Get(function(x) x$Weight / x$parent$Weight, traversal = "post-order", assign = "WoP")
14 pfo$Get(attribute = Aggregate,
15         function(x) x$Weight * x$Duration / x$parent$Weight,
16         fun = function(x) sum(x, na.rm = TRUE),
17         traversal = "post-order",
18         assign = "Duration")
19
20 #Formatters
21 pfo$formatters$WoP <- function(x) FormatPercent(x, digits = 1)
22 pfo$formatters$Weight <- FormatPercent
23 pfo$formatters$Duration <- function(x) {
24   if (x != 0) res <- FormatFixedDecimal(x, digits = 1)
25   else res <- ""
26   return (res)
27 }
28
29 #Print
30 print(pfo,
31       "weight",
32       "WoP",
33       "Duration",
34       filterFun = function(x) !x$isLeaf)
```



Result:

	LevelName	Weight	WoP	Duration
1	portfolio	100.00 %		0.8
2	--Cash	11.00 %	11.0 %	
3	--CHF	3.00 %	27.3 %	
4	--EUR	6.00 %	54.5 %	
5	*--USD	2.00 %	18.2 %	
6	--Fixed Income	28.50 %	28.5 %	3.0
7	--EUR	26.00 %	91.2 %	3.1
8	--Sovereign and Corporate Bonds	18.50 %	71.2 %	2.4
9	--Emerging Markets Bonds	3.00 %	11.5 %	6.8
10	*--High Yield Bonds	4.50 %	17.3 %	3.4
11	*--USD	2.50 %	8.8 %	1.6
12	*--High Yield Bonds	2.50 %	100.0 %	1.6
13	--Equities	40.00 %	40.0 %	
14	--Switzerland	6.00 %	15.0 %	
15	--Euroland	14.50 %	36.2 %	
16	--US	8.10 %	20.2 %	
17	--UK	0.90 %	2.2 %	
18	--Japan	3.00 %	7.5 %	
19	--Australia	2.00 %	5.0 %	
20	*--Emerging Markets	5.50 %	13.7 %	
21	*--Alternative Investments	20.50 %	20.5 %	
22	--Real Estate	5.50 %	26.8 %	
23	*--Eurozone	5.50 %	100.0 %	
24	--Hedge Funds	10.50 %	51.2 %	
25	*--Commodities	4.50 %	22.0 %	

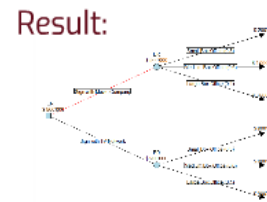
Decision Tree

Data:

```
yaml
1 name: berry_lined
2 type: decision
3 sign: after_wash_country:
4 type: choice
5 small_box_office:
6 type: terminal
7 p: 0.8
8 payoff: 200000
9 Medium_Box_office:
10 type: terminal
11 p: 0.9
12 payoff: 1000000
13 Large_Box_office:
14 type: terminal
15 p: 0.1
16 payoff: 1000000
17 sign: with_TV_Network:
18 type: choice
19 small_box_office:
20 type: terminal
21 p: 0.2
22 payoff: 600000
23 Medium_Box_office:
```

Code:

```
python
1 # Import the necessary libraries
2 from decision_tree import *
3
4 # Create the decision tree
5 dt = DecisionTree()
6
7 # Add the root node
8 dt.add_node(1, name="berry_lined", type="decision",
9            sign="after_wash_country",
10            children=[(2, "small_box_office", "type: terminal",
11                    "p: 0.8", "payoff: 200000"),
12                    (3, "Medium_Box_office", "type: terminal",
13                    "p: 0.9", "payoff: 1000000"),
14                    (4, "Large_Box_office", "type: terminal",
15                    "p: 0.1", "payoff: 1000000")],
16            value=0)
17
18 # Add the child nodes
19 dt.add_node(5, name="with_TV_Network", type="choice",
20            sign="with_TV_Network",
21            children=[(6, "small_box_office", "type: terminal",
22                    "p: 0.2", "payoff: 600000"),
23                    (7, "Medium_Box_office", "type: terminal",
24                    "p: 0.8", "payoff: 1000000")],
25            value=0)
26
27 # Print the decision tree
28 dt.print_tree()
```



ckdown

Result:

```
python
1 # Import the necessary libraries
2 from decision_tree import *
3
4 # Create the decision tree
5 dt = DecisionTree()
6
7 # Add the root node
8 dt.add_node(1, name="berry_lined", type="decision",
9            sign="after_wash_country",
10            children=[(2, "small_box_office", "type: terminal",
11                    "p: 0.8", "payoff: 200000"),
12                    (3, "Medium_Box_office", "type: terminal",
13                    "p: 0.9", "payoff: 1000000"),
14                    (4, "Large_Box_office", "type: terminal",
15                    "p: 0.1", "payoff: 1000000")],
16            value=0)
17
18 # Add the child nodes
19 dt.add_node(5, name="with_TV_Network", type="choice",
20            sign="with_TV_Network",
21            children=[(6, "small_box_office", "type: terminal",
22                    "p: 0.2", "payoff: 600000"),
23                    (7, "Medium_Box_office", "type: terminal",
24                    "p: 0.8", "payoff: 1000000")],
25            value=0)
26
27 # Print the decision tree
28 dt.print_tree()
```

yaml

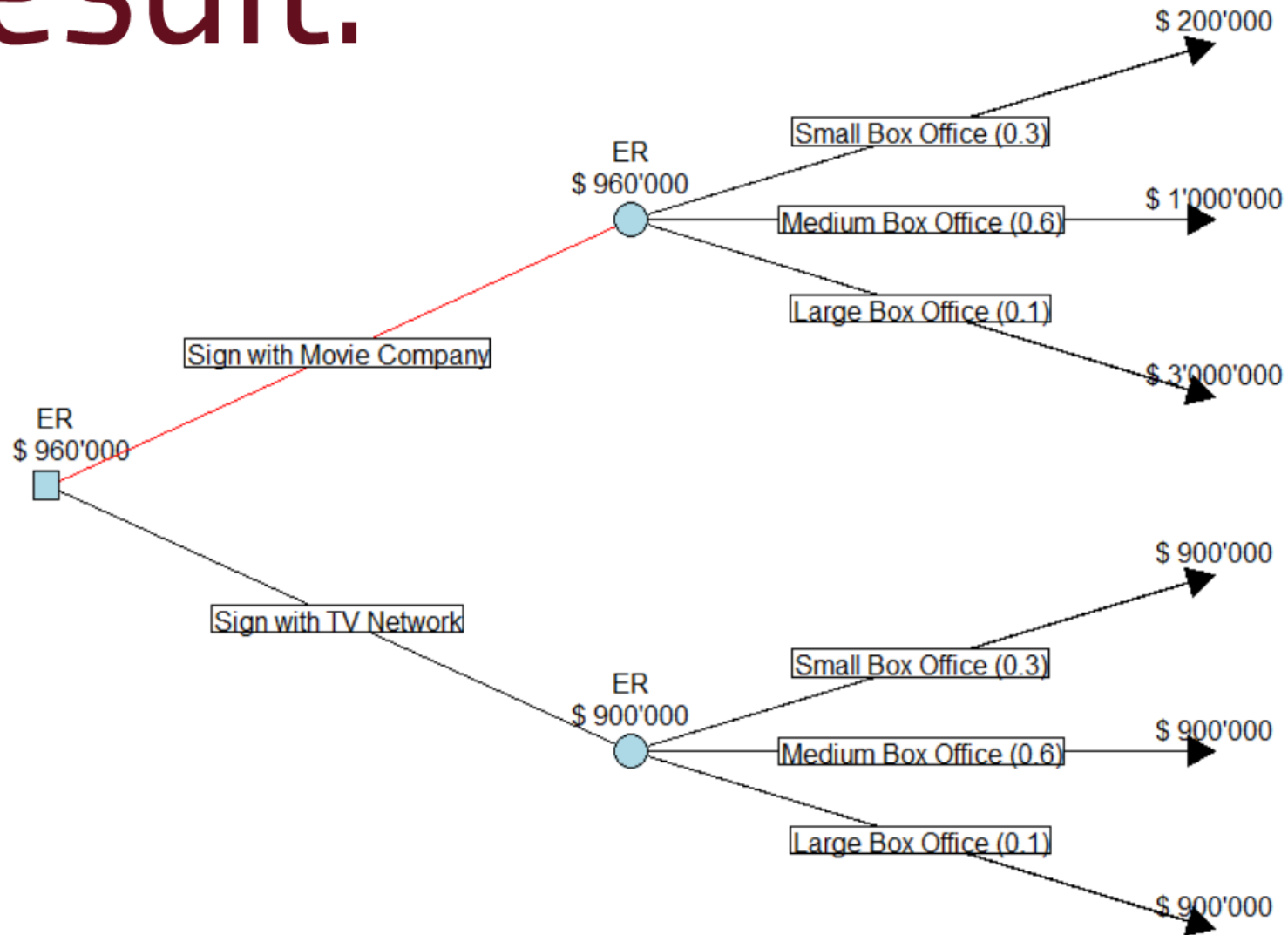
Data:

```
1 name: Jenny Lind
2 type: decision
3 Sign with Movie Company:
4   type: chance
5   Small Box office:
6     type: terminal
7     p: 0.3
8     payoff: 200000
9   Medium Box Office:
10    type: terminal
11    p: 0.6
12    payoff: 1000000
13   Large Box Office:
14     type: terminal
15     p: 0.1
16     payoff: 3000000
17 Sign with TV Network:
18   type: chance
19   Small Box office:
20     type: terminal
21     p: 0.3
22     payoff: 900000
23   Medium Box Office:
```

Code:

```
1 library(data.tree)
2 library(yaml)
3
4 #load from file
5 fileName <- '../useR15/data/jennyind.yaml'
6 j1 <- yaml.load_file(fileName)
7 j1 <- as.Node(j1)
8
9 #calculate decision tree
10
11 payoff <- function(x) {
12   if (x$type == 'terminal') res <- x$payoff
13   else if (x$type == 'chance') res <- x$Aggregate(function(node) node$payoff * node$prob, sum)
14   else if (x$type == 'decision') res <- x$Aggregate("payoff", max)
15   return (res)
16 }
17
18 j1$Get(payoff, traversal = "post-order", assign = "payoff")
19
20 decision <- function(x) {
21   if (x$type == 'decision') {
22     po <- sapply(x$children, function(child) child$payoff)
23     res <- names(po[po == x$payoff])
24   } else res <- NULL
25   return (res)
26 }
27
28 j1$Get(decision, assign = "decision")
29
30 #Plot the decision tree with ape
31
32 library(ape)
33 j1$Revert()
34 j1p <- as.phylo(j1)
35 par(mar=c(1,1,1,1))
36 plot(j1p, show.tip.label = FALSE, type = "cladogram")
37
38
39 nodeLabel <- function(x) {
40   po <- paste0('$ ', format(x$payoff, scientific = FALSE, big.mark = ''))
41   if (x$type == 'terminal') return (po)
42   return (paste0('E', po))
43 }
44
45 for (node in j1$leaves) edges(GetPhyloNr(node$parent, "node"), GetPhyloNr(node, "node"), arrows = 2, type = "triangle", angle = 60)
46
47 for (node in j1$Get(function(x) x)) {
48   if (node$type == 'decision') {
49     nodeLabels(nodeLabel(node), GetPhyloNr(node, "node"), frame = 'none', adj = c(0.3, -0.5))
50   } else if (node$type == 'chance') {
51     if (node$name == node$parent$decision) edges(GetPhyloNr(node$parent, "node"), GetPhyloNr(node, "node"), col = "red")
52     nodeLabels(" ", GetPhyloNr(node, "node"), frame = "circle")
53     nodeLabels(nodeLabel(node), GetPhyloNr(node, "node"), frame = 'none', adj = c(0.5, -0.5))
54     edgeLabels(node$name, GetPhyloNr(node, "edge"), bg = "white")
55   } else if (node$type == 'terminal') {
56     tipLabels(nodeLabel(node), GetPhyloNr(node, "node"), frame = "none", adj = c(0.5, -0.6))
57     edgeLabels(paste0(node$name, " (", node$prob, ")"), GetPhyloNr(node, "edge"), bg = "white")
58   }
59 }
60
61 nodeLabels(" ", GetPhyloNr(j1, "node"), frame = "rect")
```

Result:





networkD3

Data in csv:

```
1 # csv file with 10 columns: source, target, weight, source_label, target_label, weight_label, source_color, target_color, weight_color, weight_size
2 source,target,weight,source_label,target_label,weight_label,source_color,target_color,weight_color,weight_size
3 100,100,0.1,100,100,0.1,100,100,0.1,100
4 100,100,0.1,100,100,0.1,100,100,0.1,100
5 100,100,0.1,100,100,0.1,100,100,0.1,100
6 100,100,0.1,100,100,0.1,100,100,0.1,100
7 100,100,0.1,100,100,0.1,100,100,0.1,100
8 100,100,0.1,100,100,0.1,100,100,0.1,100
9 100,100,0.1,100,100,0.1,100,100,0.1,100
10 100,100,0.1,100,100,0.1,100,100,0.1,100
11 100,100,0.1,100,100,0.1,100,100,0.1,100
12 100,100,0.1,100,100,0.1,100,100,0.1,100
13 100,100,0.1,100,100,0.1,100,100,0.1,100
14 100,100,0.1,100,100,0.1,100,100,0.1,100
15 100,100,0.1,100,100,0.1,100,100,0.1,100
16 100,100,0.1,100,100,0.1,100,100,0.1,100
17 100,100,0.1,100,100,0.1,100,100,0.1,100
18 100,100,0.1,100,100,0.1,100,100,0.1,100
19 100,100,0.1,100,100,0.1,100,100,0.1,100
20 100,100,0.1,100,100,0.1,100,100,0.1,100
```

Code:

```
1 # Import the data from the csv file
2 data = pd.read_csv('data.csv')
3 # Create a network graph
4 graph = NetworkX.from_edgelist(data[['source', 'target', 'weight']])
5 # Create a D3.js network visualization
6 networkD3 = NetworkXtoD3.from_networkx(graph)
```

Result:



Data in csv:

	A	B	C	D	E	F	G	H
1	session	start	end	sessionName	room	seats	speaker	presentation
2	Session 1	01.07.2015 10:30	01.07.2015 12:00	Kaleidoscope 1	Aalborghallen	790	Federico Marini	flowcatchR: A use
3	Session 1	01.07.2015 10:30	01.07.2015 12:00	Kaleidoscope 1	Aalborghallen	790	Jonathan Clayden	Image processing
4	Session 1	01.07.2015 10:30	01.07.2015 12:00	Kaleidoscope 1	Aalborghallen	790	Carel F. W. Peeters	rags2ridges: Ridge
5	Session 1	01.07.2015 10:30	01.07.2015 12:00	Kaleidoscope 1	Aalborghallen	790	Henrik Tobias Madsen	dgRaph: Discrete
6	Session 1	01.07.2015 10:30	01.07.2015 12:00	Ecology	GÃstesalen	149	Costas Varsos	Optimized R func
7	Session 1	01.07.2015 10:30	01.07.2015 12:00	Ecology	GÃstesalen	149	David L Miller	Building ecologica
8	Session 1	01.07.2015 10:30	01.07.2015 12:00	Ecology	GÃstesalen	149	Andrew Dolman	Simulating ecolog
9	Session 1	01.07.2015 10:30	01.07.2015 12:00	Ecology	GÃstesalen	149	Marcel Austenfeld	Graphical User
10	Session 1	01.07.2015 10:30	01.07.2015 12:00	Networks	Musiksalen	160	Gergely Daroczi	fbRads: Analyzing
11	Session 1	01.07.2015 10:30	01.07.2015 12:00	Networks	Musiksalen	160	Peter MeiÃner	Web scraping wit
12	Session 1	01.07.2015 10:30	01.07.2015 12:00	Networks	Musiksalen	160	Antonio Rivero Ostoic	multiplex: Analyzi
13	Session 1	01.07.2015 10:30	01.07.2015 12:00	Networks	Musiksalen	160	Gabor Csardi	What's new in igr
14	Session 1	01.07.2015 10:30	01.07.2015 12:00	Reproducibility	Det lille Teate	224	Karthik Ram	rOpenSci: A suite
15	Session 1	01.07.2015 10:30	01.07.2015 12:00	Reproducibility	Det lille Teate	224	Michael Lawrence	Enhancing reprod
16	Session 1	01.07.2015 10:30	01.07.2015 12:00	Reproducibility	Det lille Teate	224	Joshua R. Polanin & Emily A.	A Review of Meta
17	Session 1	01.07.2015 10:30	01.07.2015 12:00	Reproducibility	Det lille Teate	224	David Smith	Simple reproducib
18	Session 1	01.07.2015 10:30	01.07.2015 12:00	Interfacing	Rudiosalen	216	Kasper D. Hansen	Some lessons rele
19	Session 1	01.07.2015 10:30	01.07.2015 12:00	Interfacing	Rudiosalen	216	Karl Millar	CXXR: Modernizir
20	Session 1	01.07.2015 10:30	01.07.2015 12:00	Interfacing	Rudiosalen	216	Matt P. Dziubinski	Naturally Sweet R
21	Session 1	01.07.2015 10:30	01.07.2015 12:00	Interfacing	Rudiosalen	216	Dan Putler	Linking R to the S
22	Session 2	01.07.2015 13:30	01.07.2015 15:00	Kaleidoscope 2	Aalborghallen	790	Przemyslaw Biecek	archivist: Tools fo

Code:

```
1 library(data.tree)
2
3 userRdf <- read.csv('../user15/data/user15.csv', stringsAsFactors = FALSE, encoding = "UTF-8")
4 userRdf$pathString <- paste("user", userRdf$session, userRdf$room, userRdf$speaker, sep="|")
5 userRtree <- as.Node(userRdf, pathDelimiter = "|")
6
7 library(networkD3)
8 network <- treeNetwork(userRtree$ToList(mode = "explicit", unname = TRUE))
9 saveNetwork(network, "network.html", selfcontained = TRUE)
10
```

Result:





friends & family



- `igraph`
- `dendrogram`
- `data.table`
- `partykit` and friends
- `list` of lists of lists



- igraph
- dendrogram
- data.table
- partykit and friends
- list of lists of lists

data.tree in action



Create a tree programmatically

Tree structure:

- level 1/root: useR! 2015 Sessions
- level 2: session (name, start, end)
- level 3: room (name, seats)
- level 4: presentation (speaker, presentation name)

```
library(data.tree)
```

```
useR <- Node$new("UseR! Conference 2015")
  session1 <- useR$AddChild("Session 1")
    aalborghallen <- session1$AddChild("Aalborghallen")
      aalborghallen$seats <- 790
        p1 <- aalborghallen$AddChild("Federico Marini")
        p2 <- aalborghallen$AddChild("Jonathan Clayden")
        #... etc.
      gaestsalen <- session1$AddChild("Gaestsalen")
        gaestsalen$seats <- 149
          p1 <- gaestsalen$AddChild("Costas Varsos")
          # ... etc
```

```
p2 <- gaestsalen$AddChild("Jonathan Clayton")  
#... etc.  
gaestsalen <- session1$AddChild("Gaestsalen")  
gaestsalen$seats <- 149  
p1 <- gaestsalen$AddChild("Costas Varsos")  
#... etc.
```

Normally, you would do this in an algorithm.

See:

```
vignette("ID3")
```

OO-Style method calling

```
user$ToList()
```

But you can still use “classical” R generics if you prefer. The following is equivalent:

```
as.list(user)
```

Reference Semantics

Nodes exhibit reference semantics:

```
session1$day <- "Wed"  
useR$Find("Session 1")$day
```

```
## [1] "Wed"
```

Equivalent operation with lists doesn't work:

```
useRlist <- list()  
  session1list <- list()  
  useRlist$session1 <- session1list  
  session1list$day <- "Wed"  
useRlist$session1$day
```

```
## NULL
```


Basic actives on nodes

```
useR$isLeaf
```

```
## [1] FALSE
```

```
useR$isRoot
```

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

```
useR$depth
```

```
## [1] 4
```

```
useR$depth
```

```
## [1] 4
```

```
useR$count
```

```
## [1] 6
```

```
useR$totalCount
```

```
## [1] 164
```

```
useR$leafCount
```

```
## [1] 127
```



Navigation

Navigate to a specific node:

```
glur <- useR$Find("Session 2",  
                 "Det lille Teater",  
                 "Christoph Glur")
```

...and navigate relatively to it:

```
glur$parent$name
```

```
## [1] "Det lille Teater"
```

```
glur$root$name
```

```
## [1] "useR"
```

Attributes

Find attributes:

```
glur$fields
```

```
## [1] "presentation" "speaker"
```

```
glur$fieldsAll
```

```
## [1] "presentation" "speaker"
```

... and access them:

```
glur$presentation
```

```
## [1] "A better way to manage hierarchical data"
```

```
glur$presentation
```

```
## [1] "A better way to manage hierarchical data"
```

Add new ones if you wish:

```
glur$package <- 'data.tree'  
glur$fields
```

```
## [1] "package" "presentation" "speaker"
```

```
glur$root$fieldsAll
```

```
## [1] "end" "session" "sessionName"  
## [4] "start" "room" "seats"  
## [7] "presentation" "speaker" "package"
```



Create a deep copy

```
useR2 <- useR$Clone()
```

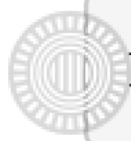
Prune

```
myPrune <- function(x) {  
  return(x$level!=3 || x$room == 'Radiosalen')  
}
```

```
useR2$Prune(pruneFun = myPrune)  
useR2$totalCount
```

```
## [1] 39
```

```
head(useR2$ToDataFrame(), n = 10)
```



```
useR2$totalCount
```

```
## [1] 39
```

```
head(useR2$ToDataFrame(), n = 10)
```

```
##                                     levelName
## 1  useR
## 2  |--Session 1
## 3  |    °--Radiosalen
## 4  |          |--Kasper D. Hansen
## 5  |          |--Karl Millar
## 6  |          |--Matt P. Dziubinski
## 7  |          °--Dan Putler
## 8  |--Session 2
## 9  |    °--Radiosalen
## 10 |          |--E. James Harner
```

```
## 7 | °--Dan Putler
## 8 | --Session 2
## 9 | °--Radiosalen
## 10 | |--E. James Harner
```

Sort

```
useR$Sort("speaker")
print(useR$Find("Session 2", "Det lille Teater"))
```

```
##                               levelName
## 1 Det lille Teater
## 2 |--Christoph Glur
## 3 |--Filip Schouwenaars
## 4 |--Hadley Wickham
## 5 |--Indrajit Roy, Michael Lawrence
## 6 °--Tony Fischetti
```


data.frame

Load in the tabular data from csv:

```
library(data.tree)
useRdf <- read.csv('../data/useR15.csv', stringsAsFactors = FALSE, encoding = "UTF-8")
head(useRdf[,c("session", "start", "sessionName", "room", "seats", "speaker")])
```

##	session	start	sessionName	room	seats	speaker
## 1	Session 1	01.07.2015 10:30	Kaleidoscope 1	Aalborghallen	790	Federico Marini
## 2	Session 1	01.07.2015 10:30	Kaleidoscope 1	Aalborghallen	790	Jonathan Clayden
## 3	Session 1	01.07.2015 10:30	Kaleidoscope 1	Aalborghallen	790	Carel F. W. Peeters
## 4	Session 1	01.07.2015 10:30	Kaleidoscope 1	Aalborghallen	790	Henrik Tobias Madsen
## 5	Session 1	01.07.2015 10:30	Ecology	Gæstesalen	149	Costas Varsos
## 6	Session 1	01.07.2015 10:30	Ecology	Gæstesalen	149	David L Miller

Convert it to a data.tree

```
#1. the pathString defines the hierarchy:
useRdf$pathString <- paste("useR", useRdf$session, useRdf$room, useRdf$speaker, sep="|")

#2. advanced and optional: define which attributes go to which level in the tree
#By default, they go to the leaf
cols <- list(NULL, #root
             c('session', 'start', 'end', 'sessionName'), #session
             c('room', 'seats') #room
            )

#3. convert to tree
```



```
## 2 Session 1 01.07.2015 10:30 Kaleidoscope 1 Aalborghallen 790 Jonathan Clayden
## 3 Session 1 01.07.2015 10:30 Kaleidoscope 1 Aalborghallen 790 Carel F. W. Peeters
## 4 Session 1 01.07.2015 10:30 Kaleidoscope 1 Aalborghallen 790 Henrik Tobias Madsen
## 5 Session 1 01.07.2015 10:30 Ecology Gæstesalen 149 Costas Varsos
## 6 Session 1 01.07.2015 10:30 Ecology Gæstesalen 149 David L Miller
```

Convert it to a data.tree

```
#1. the pathString defines the hierarchy:
useRdf$pathString <- paste("useR", useRdf$session, useRdf$room, useRdf$speaker, sep="|")

#2. advanced and optional: define which attributes go to which level in the tree
#By default, they go to the leaf
cols <- list(NULL, #root
             c('session', 'start', 'end', 'sessionName'), #session
             c('room', 'seats') #room
            )

#3. convert to tree
useRtree <- as.Node(useRdf, pathDelimiter = "|", colLevels = cols)

print(useRtree,
      "start", "seats",
      filterFun = function(x) x$level < 4)
```

```
##          levelName          start seats
## 1  useR
## 2  |--Session 1          01.07.2015 10:30  NA
## 3  |  |--Aalborghallen          790
## 4  |  |--Gæstesalen          149
## 5  |  |--Musiksalen          160
## 6  |  |--Det lille Teater          224
## 7  |  °--Radiosalen          216
```



```
print(useRtree,
      "start", "seats",
      filterFun = function(x) x$level < 4)
```

```
##           levelName           start seats
## 1  useR
## 2  |--Session 1           01.07.2015 10:30   NA
## 3  |  |--Aalborghallen
## 4  |  |--Gæstesalen           149
## 5  |  |--Musiksalen           160
## 6  |  |--Det lille Teater     224
## 7  |  °--Radiosalen           216
## 8  |--Session 2           01.07.2015 13:30   NA
## 9  |  |--Aalborghallen
## 10 |  |--Gæstesalen           149
## 11 |  |--Musiksalen           160
## 12 |  |--Det lille Teater     224
## 13 |  °--Radiosalen           216
## 14 |--Session 3           01.07.2015 16:00   NA
## 15 |  |--Aalborghallen
## 16 |  |--Gæstesalen           149
## 17 |  |--Musiksalen           160
## 18 |  |--Det lille Teater     224
## 19 |  °--Radiosalen           216
## 20 |--Session 4           02.07.2015 10:30   NA
## 21 |  |--Radiosalen           216
## 22 |  |--Aalborghallen
## 23 |  |--Gæstesalen           149
## 24 |  |--Musiksalen           160
## 25 |  °--Det lille Teater     224
## 26 |--Session 5           02.07.2015 13:00   NA
## 27 |  |--Aalborghallen           790
```



traversal

```
session2 <- useR$Find("Session 2")
session2$Set(preo = 1:session2$totalCount)
print(session2, "preo")
```

```
##                               levelName preo
## 1   Session 2                               1
## 2   |--Aalborghallen                       2
## 3   |   |--Przemyslaw Biecek               3
## 4   |   |--Joseph B. Rickert              4
## 5   |   |--Richard M. Heiberger           5
## 6   |   °--Rasmus Bååth                   6
## 7   |--Gæstesalen                          7
## 8   |   |--Johannes Breidenbach           8
## 9   |   |--Ivan Kasanický                 9
## 10  |   |--Jakob W. Messner               10
## 11  |   °--Helle Sørensen                 11
## 12  |--Musiksalen                          12
## 13  |   |--Anders Ellern Bilgrau         13
```



Compare this with post-order and level (breadth first):

```
session2$Set(posto = 1:session2$totalCount,  
             traversal = "post-order")  
session2$Set(lev = 1:session2$totalCount,  
            traversal = "level")  
print(session2, "preo", "posto", "lev")
```

##		levelName	preo	posto	lev
##	1	Session 2	1	28	1
##	2	--Aalborghallen	2	5	2
##	3	--Przemyslaw Biecek	3	1	7
##	4	--Joseph B. Rickert	4	2	8
##	5	--Richard M. Heiberger	5	3	9
##	6	°--Rasmus Bååth	6	4	10
##	7	--Gæstesalen	7	10	3
##	8	--Johannes Breidenbach	8	6	11
##	9	--Ivan Kasanický	9	7	12
##	10	--Jakob W. Messner	10	8	13
##	11	°--Helle Sørensen	11	9	14
##	12	--Musiksalen	12	15	4



You can also collect data instead of setting data:

```
datamanagement <- session2$Find("Det lille Teater")  
datamanagement$Get("name")
```

```
## [1] "Det lille Teater"  
## [2] "Filip Schouwenaars"  
## [3] "Tony Fischetti"  
## [4] "Hadley Wickham"  
## [5] "Christoph Glur"  
## [6] "Indrajit Roy, Michael Lawrence"
```

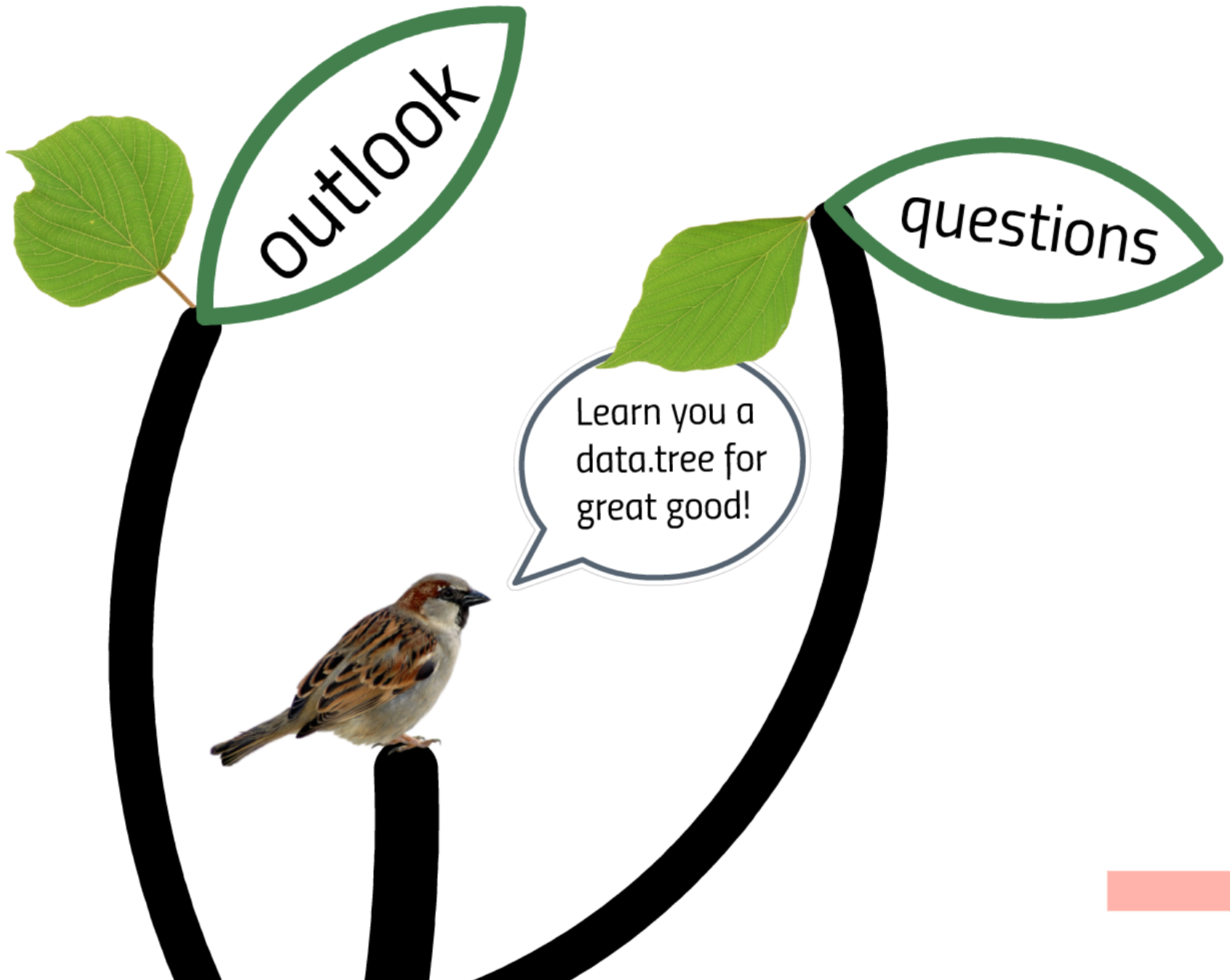
Works also on functions:

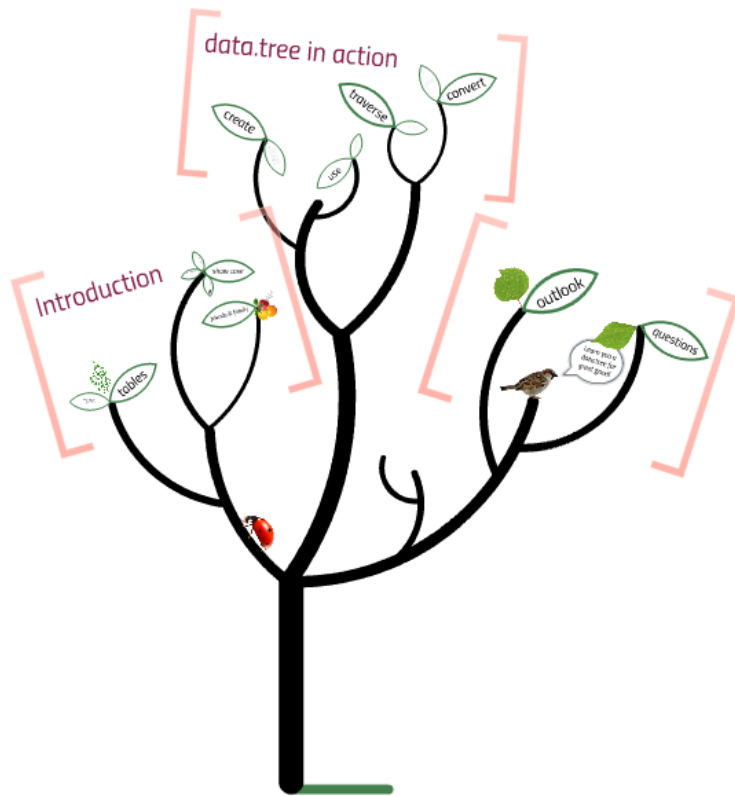
```
toUpperName <- function(node) {  
  toupper(node$name)  
}  
  
datamanagement$Get(toUpperName)
```

```
## [1] "DET LILLE TEATER"  
## [2] "FILIP SCHOUWENAARS"  
## [3] "TONY FISCHETTI"  
## [4] "HADLEY WICKHAM"  
## [5] "CHRISTOPH GLUR"  
## [6] "INDRAJIT ROY, MICHAEL LAWRENCE"
```

And on actives, with pruning and filtering, and and and:

```
useR$Get("totalCount",  
        traversal = "level",  
        pruneFun = function(x) {  
            x$level != 3 || x$room == "Det lille Teater"  
        },  
        filterFun = function(x) !x$isLeaf)
```





data.tree

A better way to manage
hierarchical data

useR! 2015

<http://github.com/gluc/useR15>

christoph.glur@ipub.com