Lab Session

Quantitative Drama Analytics

Benjamin Krautter Nils Reiter

Digitale Analyse großer Textkorpora, Ruhr-Universität Bochum

17. Dezember 2021

INSTITUT FÜR
DIGITAL HUMANITIES
UNIVERSITÄT ZU KÖLN

CRETA
CENTER FOR REFLECTED TEXT ANALYTICS

Section 1

What you really need to know about R

R Basics

- R is a programming language
  - Mostly used for statistical data analysis ("data science")
  - First version: 1993, current stable release: 4.1
  - Website: https://www.r-project.org
  - Open source
- Three important concepts we need to talk about
  - Objects/Types
  - Variables
  - Functions

R Basics

Objects and Types

- Objects live in the computer memory (or on disk)
- Objects represent the things we want to analyse (e.g., dramatic texts, words, or numbers)
- Variables are used to operate on objects (similar to Python)
- An object has one or more types

R Basics

Objects and Types

- Objects live in the computer memory (or on disk)
- Objects represent the things we want to analyse (e.g., dramatic texts, words, or numbers)
- Variables are used to operate on objects (similar to Python)
- An object has one or more types
- The type of an object determines what we can do with it
  - E.g., a knife allows other operations than a fork

R Basics

Objects and Types

- Objects live in the computer memory (or on disk)
- Objects represent the things we want to analyse (e.g., dramatic texts, words, or numbers)
- Variables are used to operate on objects (similar to Python)
- An object has one or more types
- The type of an object determines what we can do with it
  - E.g., a knife allows other operations than a fork
- Types: Numbers, character sequences ("strings"), lists, tables, ...
  - Numbers allow arithmetic operations
    - E.g., summation: `sum(3,5)` (evaluates to 8, equivalent to 3+5)
  - Character sequences allow character-based operations
    - E.g., conversion to lower case: `tolower("ABC")` (evaluates to `"abc"`)

R Basics

Objects and Types

- Objects live in the computer memory (or on disk)
- Objects represent the things we want to analyse (e.g., dramatic texts, words, or numbers)
- Variables are used to operate on objects (similar to Python)
- An object has one or more types
- The type of an object determines what we can do with it
  - E.g., a knife allows other operations than a fork
- Types: Numbers, character sequences ("strings"), lists, tables, ...
  - Numbers allow arithmetic operations
    - E.g., summation: `sum(3,5)` (evaluates to 8, equivalent to 3+5)
  - Character sequences allow character-based operations
    - E.g., conversion to lower case: `tolower("ABC")` (evaluates to `"abc"`)
- "evalutes to": result of the operation/function

R Basics

Objects and Types

Type	Example	Description
Numeric	`5`	A numeric value
Character	`"Bochum"`	A sequence of characters (note the double quotes!)
Logical	`TRUE/FALSE`	A truth value
Vector	`c(5,4,1)`	Sequence of objects *of the same type*
List	`list(5,"Hd",TRUE)`	Sequence of objects (Python list/dictionary)
Matrix		Table of objects *of the same type*
Data frame		Table of objects

R Basics

Objects and Types

In R, there are no single-value variables: Everything is a vector!

- Entering `5` creates a numeric vector of length 1
- Entering `"Bla"` creates a character vector of length 1

(In this way, R is different from other programming languages)

```
5
# Creates a vector consisting of the numbers 1 to 50
1:50
```

R Basics

Variables

- We usually do not interact with the objects directly
  - Because they are not known in advance (but loaded from files)
- Variables
  - A way to *name* objects
  - Used as a placeholder for objects
  - The actual operation takes place on the objects (R takes care of this)
- Creating a variable a: `a <- 3` (think of this as an arrow)

```
> a <- 3
> b <- 5
> a + b
[1] 8
>
```

R Basics

Functions

- "Mini programs": A collection of instructions that you can use as a single instruction
- Input: Functions take *arguments* as input
- Output: Functions return an object (that stores the result of the instructions)

R Basics

Functions

- "Mini programs": A collection of instructions that you can use as a single instruction
- Input: Functions take *arguments* as input
- Output: Functions return an object (that stores the result of the instructions)
- Functions have a name (typically lower case) and can be recognized by the round parentheses
  `function(argument1, argument2, argument3, ...)`
- The return value of a function can be stored in a variable
  `variable <- function(arg1, arg2, ...)`

R Basics

Functions

- "Mini programs": A collection of instructions that you can use as a single instruction
- Input: Functions take *arguments* as input
- Output: Functions return an object (that stores the result of the instructions)
- Functions have a name (typically lower case) and can be recognized by the round parentheses
  `function(argument1, argument2, argument3, ...)`
- The return value of a function can be stored in a variable
  `variable <- function(arg1, arg2, ...)`
- Some functions not only return a value, but also do something (e.g., display a plot)
- Pipeline: Multiple functions operating in succession

Subsection 1

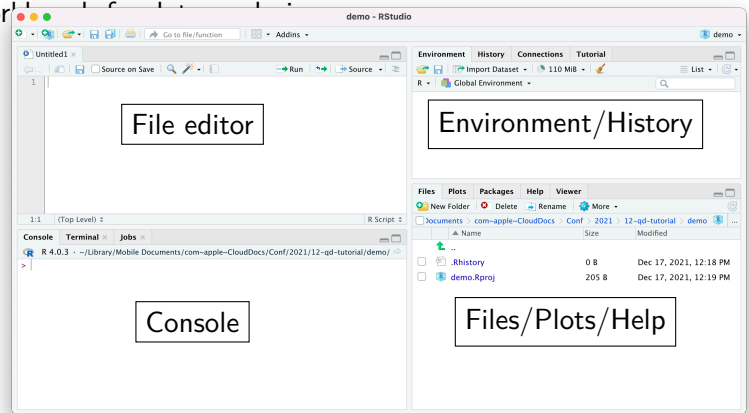
RStudio

RStudio

- An integrated development environment (IDE) for R
- Capable workbench for data analysis

RStudio

- An integrated development environment (IDE) for R
- Capable wor



RStudio

Four Panes

- Console: Where you enter R code and get the result immediately
- Environment: Shows the objects currently in memory
  - History: Previous commands
- Plots: Shows plots
  - Help: Documentation for packages/functions
- Editor/Code: Allows editing R code and inspecting tables

We will focus on the console and plot area

Section 2

DramaAnalysis

Outline

- Introduction/Installation and Overview
- Three areas for you to play with
  1. Global character statistics
  2. Word fields
  3. Copresence and network analysis

Introduction

- R Package: A collection of functions and/or data sets
  - Similar to libraries in Python
- Function: Mini program
- DramaAnalysis: Functions for drama analysis (surprise!)
  - Today: Third iteration, extensive rewrite
- Philosophy: Construction kit with building blocks
  - No graphical user interface, but interactive use through console

Installation

Code

```
install.packages("DramaAnalysis")
library(DramaAnalysis)  # no quotes

 # additional package
library(magrittr)
```

Installation

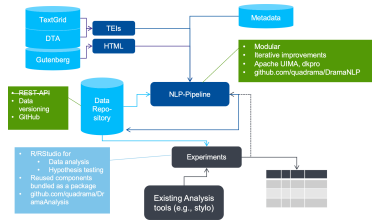
Code

```
install.packages("DramaAnalysis")
library(DramaAnalysis)  # no quotes

 # additional package
library(magrittr)
```

Installation

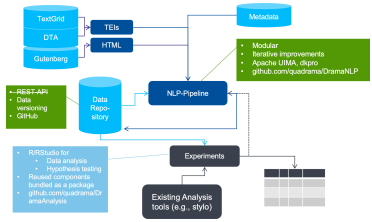
Data



- Dramatic texts are initially stored as TEI/XML files
  - From https://dracor.org
- Language processing (e.g., identification of parts of speech) takes place in a UIMA pipeline: https://github.com/quadrama/DramaNLP
- Output of the pipeline: Several CSV files (i.e., tables) for each play (meta data, character data, ...)
- CSV files are analysed in R

Installation

Data



- Dramatic texts are initially stored as TEI/XML files
  - From https://dracor.org
- Language processing (e.g., identification of parts of speech) takes place in a UIMA pipeline: https://github.com/quadrama/DramaNLP
- Output of the pipeline: Several CSV files (i.e., tables) for each play (meta data, character data, ...)
- CSV files are analysed in R

Two corpora today:

```
installData("gdc") # German literary canon, GerDraCor
# or
installData("shakedracor") # English Shakespeare plays
```

Installation

Data

The function `installData()`

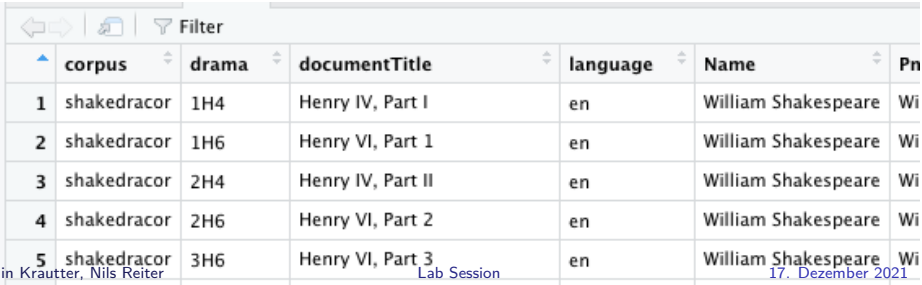
- Clones a git repository from `github.com/quadrama` into a local directory
- Allows easy update of data files
- German literary canon (`qd`)
  - TextGrid → GerDraCor → QuaDramA
- English Shakespeare plays (`shakedracor`)
  - Folger → DraCor → QuaDramA
- Two demo plays included in the package
  - Including manual coreference annotation
  - Lessings *Emilia Galotti* and *Miss Sara Sampson* (German)

Inspecting data

```
# Collect all play ids into a vector
loadAllInstalledIds() %>%
  # Extract metadata for each play,
  # put it into a table
  loadMeta()          %>%
  # Have RStudio display a nice table
  View()
```

Inspecting data

```
# Collect all play ids into a vector
loadAllInstalledIds() %>%
  # Extract metadata for each play,
  # put it into a table
  loadMeta()          %>%
  # Have RStudio display a nice table
  View()
```



Filter

	corpus	drama	documentTitle	language	Name	Pn
1	shakedracor	1H4	Henry IV, Part I	en	William Shakespeare	Wi
2	shakedracor	1H6	Henry VI, Part 1	en	William Shakespeare	Wi
3	shakedracor	2H4	Henry IV, Part II	en	William Shakespeare	Wi
4	shakedracor	2H6	Henry VI, Part 2	en	William Shakespeare	Wi
5	shakedracor	3H6	Henry VI, Part 3	en	William Shakespeare	Wi

Loading a play

- We first have to load plays into the environment
- Each play has an associated id
- Select one and create a variable to store the id (less typing in the future)

Loading a play

- We first have to load plays into the environment
- Each play has an associated id
- Select one and create a variable to store the id (less typing in the future)

```
# General form: collection colon play
# (allows comparison across collections)
myId <- "shakedracor:Rom"

play <- loadDrama(myId)
```

Online help

- Each function is documented
- Entering question mark followed by the function name opens the help view
  - `?loadDrama`
- Documentation
  - What does the function do?
  - What arguments does it expect, which default values are defined?
  - What does it return?
  - Usage example

Online help

- Each function is documented
- Entering question mark followed by the function name opens the help view
  - `?loadDrama`
- Documentation
  - What does the function do?
  - What arguments does it expect, which default values are defined?
  - What does it return?
  - Usage example

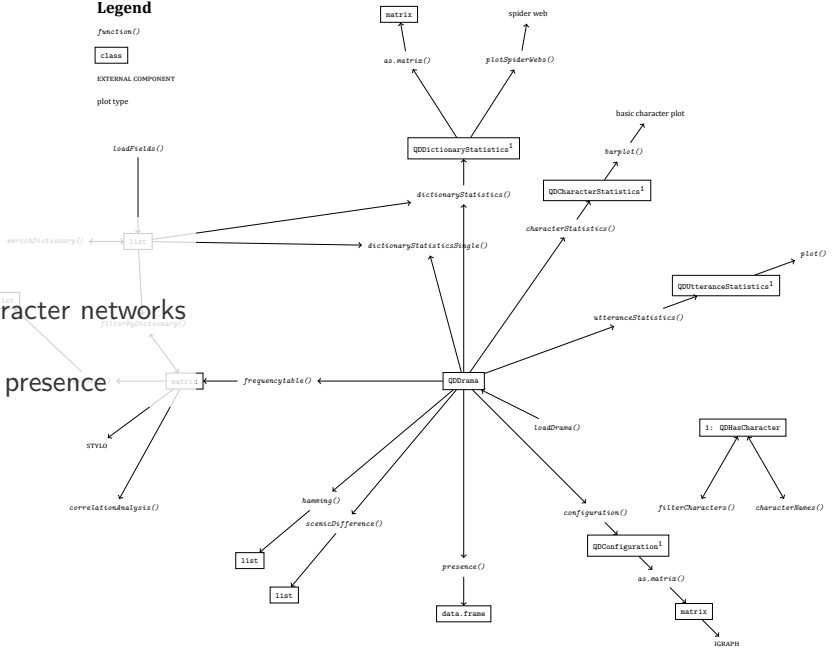
Package documentation: https://quadrama.github.io/DramaAnalysis/3.0.0
Tutorial: https://quadrama.github.io/DramaAnalysis/tutorial/3/index.html

Use cases

1. Basics
2. Stage presence and character networks

Break

3. Collections and passive presence



Subsection 1

Basics

Function characterStatistics I

Returns a table (in R: `data.frame`) with

- `corpus`: The collection id
- `drama`: The play id
- `character`: the character id
- `tokens`: Number of tokens (for this character)
- `types`: Number of different tokens (for this character)
- `utterances`: Number of utterances (for this character)
- `utteranceLengthMean`: Mean utterance length
- `utteranceLengthSd`: Utterance length standard deviation
- `firstBegin`: Starting position of the first utterance
- `lastEnd`: End position of the last utterance

Function characterStatistics II

The function `View()` can be used to get browsable table in RStudio:

```
cs <- characterStatistics(play)
View(cs)
```

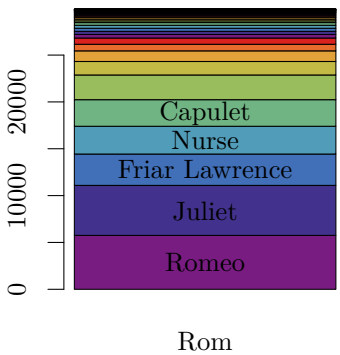
Plotting characterStatistics I

"Take the output of the function, replace character ids with names and make bar plot"

```
# load a play
play <- loadDrama("shakedracor:Rom")

# call the function
characterStatistics(play) %>%
  # replace character ids by character names
  characterNames(play) %>%
  # plot them stacked
  barplot()
```

Plotting characterStatistics II



Subsection 2

Stage presence and character networks

Stage Presence

Ingredients

	1	2	3	4	5	6	7	8	9	10
Dirne	1	0	0	0	0	0	0	0	0	1
Soldat	1	1	0	0	0	0	0	0	0	0
Stubenmädchen	0	1	1	0	0	0	0	0	0	0
Junger Herr	0	0	1	1	0	0	0	0	0	0
Junge Frau	0	0	0	1	1	0	0	0	0	0
Ehemann	0	0	0	0	1	1	0	0	0	0
Süßes Mädel	0	0	0	0	0	1	1	0	0	0
Dichter	0	0	0	0	0	0	1	1	0	0
Schauspielerin	0	0	0	0	0	0	0	1	1	0
Graf	0	0	0	0	0	0	0	0	1	1

- Configuration: A matrix showing who is on stage when
- Function: `configuration()`
- Package `igraph`

Stage Presence – Function `configuration()` I

```
play <- loadDrama("shakedracor:Rom")
conf <- configuration(play)
```

Table with columns

- `corpus`, `drama`, `character`
- One column per segment, filled with the number of words spoken by a character

Stage Presence – Function `configuration()` II

	corpus	drama	character	1	2	3	4	5
1	shakedracor	Rom	SERVANTS.CAPULET.Sampson_Rom	329	0	0	0	0
2	shakedracor	Rom	SERVANTS.CAPULET.Gregory_Rom	183	0	0	0	0
3	shakedracor	Rom	SERVANTS.MONTAGUE.Abram_Rom	34	0	0	0	0
4	shakedracor	Rom	Benvolio_Rom	778	176	484	0	0
5	shakedracor	Rom	Tybalt_Rom	205	0	129	0	0
6	shakedracor	Rom	CITIZENS_Rom	23	0	0	0	0
7	shakedracor	Rom	Capulet_Rom	957	0	1024	762	96
8	shakedracor	Rom	LadyCapulet_Rom	368	0	505	208	53
9	shakedracor	Rom	Montague_Rom	259	0	27	0	93
10	shakedracor	Rom	LadyMontague_Rom	33	0	0	0	0
11	shakedracor	Rom	PrinceEscalus_Rom	200	0	147	0	356
12	shakedracor	Rom	Romeo_Rom	1390	1620	1244	0	1503
13	shakedracor	Rom	Paris_Rom	39	0	37	288	305
14	shakedracor	Rom	SERVANTS.CAPULET.X.1_Rom	174	0	0	0	0
15	shakedracor	Rom	Nurse_Rom	806	994	788	375	0
16	shakedracor	Rom	Juliet_Rom	256	1633	2166	1136	141
17	shakedracor	Rom	SERVANTS.CAPULET.X.2_Rom	45	0	0	0	0
18	shakedracor	Rom	Mercutio_Rom	695	1220	723	0	0
19	shakedracor	Rom	SERVANTS.CAPULET.0.1_Rom	90	0	0	61	0
20	shakedracor	Rom	SERVANTS.CAPULET.0.2_Rom	26	0	0	20	0
21	shakedracor	Rom	SERVANTS.CAPULET.0.3_Rom	29	0	0	0	0
22	shakedracor	Rom	Cousin_Rom	28	0	0	0	0
23	shakedracor	Rom	SERVANTS.CAPULET.0_Rom	6	0	0	0	0
29	shakedracor	Rom	FriarLawrence_Rom	0	851	852	793	852
30	shakedracor	Rom	SERVANTS.CAPULET.Peter_Rom	0	55	0	264	0
35	shakedracor	Rom	Petruchio_Rom	0	0	4	0	0

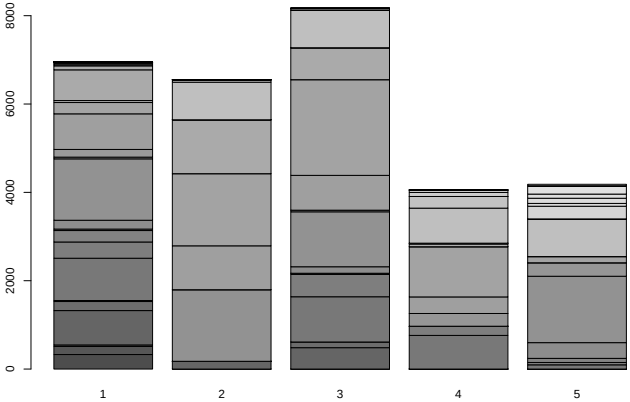
Stage Presence – Function `configuration()` III

```
## 36 shakedracor   Rom                CITIZENS.0.1_Rom    0    0   38    0    0
## 53 shakedracor   Rom              MUSICIANS.0.1_Rom     0    0    0   91    0
## 55 shakedracor   Rom              MUSICIANS.0.2_Rom     0    0    0   47    0
## 56 shakedracor   Rom              MUSICIANS.0.3_Rom     0    0    0    9    0
## 58 shakedracor   Rom SERVANTS.MONTAGUE.Balthasar_Rom    0    0    0    0  287
## 59 shakedracor   Rom                  Apothecary_Rom    0    0    0    0   64
## 60 shakedracor   Rom                   FriarJohn_Rom    0    0    0    0  117
## 63 shakedracor   Rom            SERVANTS.PARIS.1_Rom    0    0    0    0   94
## 65 shakedracor   Rom                WATCHMEN.0.1_Rom    0    0    0    0  175
## 66 shakedracor   Rom                WATCHMEN.0.2_Rom    0    0    0    0   15
## 67 shakedracor   Rom                WATCHMEN.0.3_Rom    0    0    0    0   32
```

Plotting Stage Presence I

```
configuration(play) %>%
  characterNames(play) %>%
  as.matrix() %>%
  barplot()
```

Plotting Stage Presence II



Stage Presence I

Arguments for ‘configuration()‘

- `onlyPresence`: If `TRUE`, table only contains if a character is present
- `segment`: Specifies if act or scene boundaries should be used

```
c <- configuration(play, onlyPresence=TRUE, segment="Act") %>%
  characterNames(play)
```

Stage Presence II

Arguments for 'configuration()'

	corpus	drama	character	1	2	3	4	5
1	shakedracor	Rom	Sampson	TRUE	FALSE	FALSE	FALSE	FALSE
2	shakedracor	Rom	Gregory	TRUE	FALSE	FALSE	FALSE	FALSE
3	shakedracor	Rom	Abram	TRUE	FALSE	FALSE	FALSE	FALSE
4	shakedracor	Rom	Benvolio	TRUE	TRUE	TRUE	FALSE	FALSE
5	shakedracor	Rom	Tybalt	TRUE	FALSE	TRUE	FALSE	FALSE
6	shakedracor	Rom	Three Or Four Citizens	TRUE	FALSE	FALSE	FALSE	FALSE
7	shakedracor	Rom	Capulet	TRUE	FALSE	TRUE	TRUE	TRUE
8	shakedracor	Rom	Lady Capulet	TRUE	FALSE	TRUE	TRUE	TRUE
9	shakedracor	Rom	Montague	TRUE	FALSE	TRUE	FALSE	TRUE
10	shakedracor	Rom	Lady Montague	TRUE	FALSE	FALSE	FALSE	FALSE
11	shakedracor	Rom	Escalus	TRUE	FALSE	TRUE	FALSE	TRUE
12	shakedracor	Rom	Romeo	TRUE	TRUE	TRUE	FALSE	TRUE
13	shakedracor	Rom	Paris	TRUE	FALSE	TRUE	TRUE	TRUE
14	shakedracor	Rom	Servants.capulet.x.1_rom	TRUE	FALSE	FALSE	FALSE	FALSE
15	shakedracor	Rom	Nurse	TRUE	TRUE	TRUE	TRUE	FALSE
16	shakedracor	Rom	Juliet	TRUE	TRUE	TRUE	TRUE	TRUE
17	shakedracor	Rom	Servants.capulet.x.2_rom	TRUE	FALSE	FALSE	FALSE	FALSE

Stage Presence III

Arguments for 'configuration()'

18	shakedracor	Rom	Mercutio	TRUE	TRUE	TRUE	FALSE	FALSE
19	shakedracor	Rom	Servants.capulet.0.1_rom	TRUE	FALSE	FALSE	TRUE	FALSE
20	shakedracor	Rom	Servants.capulet.0.2_rom	TRUE	FALSE	FALSE	TRUE	FALSE
21	shakedracor	Rom	Servants.capulet.0.3_rom	TRUE	FALSE	FALSE	FALSE	FALSE
22	shakedracor	Rom	Cousin_rom	TRUE	FALSE	FALSE	FALSE	FALSE
23	shakedracor	Rom	Servants.capulet.0_rom	TRUE	FALSE	FALSE	FALSE	FALSE
29	shakedracor	Rom	Friar Lawrence	FALSE	TRUE	TRUE	TRUE	TRUE
30	shakedracor	Rom	Peter	FALSE	TRUE	FALSE	TRUE	FALSE
35	shakedracor	Rom	Petruchio	FALSE	FALSE	TRUE	FALSE	FALSE
36	shakedracor	Rom	Citizens.0.1_rom	FALSE	FALSE	TRUE	FALSE	FALSE
53	shakedracor	Rom	Musicians.0.1_rom	FALSE	FALSE	FALSE	TRUE	FALSE
55	shakedracor	Rom	Musicians.0.2_rom	FALSE	FALSE	FALSE	TRUE	FALSE
56	shakedracor	Rom	Musicians.0.3_rom	FALSE	FALSE	FALSE	TRUE	FALSE
58	shakedracor	Rom	Balthasar	FALSE	FALSE	FALSE	FALSE	TRUE
59	shakedracor	Rom	Apothecary	FALSE	FALSE	FALSE	FALSE	TRUE
60	shakedracor	Rom	Friar John	FALSE	FALSE	FALSE	FALSE	TRUE
63	shakedracor	Rom	Servants.paris.1_rom	FALSE	FALSE	FALSE	FALSE	TRUE
65	shakedracor	Rom	Watchmen.0.1_rom	FALSE	FALSE	FALSE	FALSE	TRUE
66	shakedracor	Rom	Watchmen.0.2_rom	FALSE	FALSE	FALSE	FALSE	TRUE

Stage Presence IV

Arguments for ‘configuration()‘

67	shakedracor	Rom	Watchmen.0.3_rom	FALSE	FALSE	FALSE	FALSE	TRUE

Total Absolute Stage Presence

```
c <- configuration(play, onlyPresence=TRUE, segment="Scene") %>%
  filterCharacters(play, by="rank", n=10) %>%
  characterNames(play)
v <- rowSums(c)
names(v) <- c$character
```

```
##       Benvolio        Capulet   Lady Capulet        Escalus          Romeo
##              7              9             10              3             14
##          Paris          Nurse         Juliet       Mercutio Friar Lawrence
##              5             11             11              4              7
```

Total Relative Stage Presence (= Active Presence)

- Absolute numbers not useful to compare across plays
- Scaling by dividing by the total number of segments

```
c <- configuration(play, onlyPresence=TRUE, segment="Scene") %>%
  filterCharacters(play, by="rank", n=10) %>%
  characterNames(play)
v <- rowSums(c) / length(unique(play$segments$begin.Scene))
names(v) <- c$character
```

```
##       Benvolio        Capulet   Lady Capulet        Escalus          Romeo
##           0.28           0.36           0.40           0.12           0.56
##          Paris          Nurse         Juliet      Mercutio Friar Lawrence
##           0.20           0.44           0.44           0.16           0.28
```

Character Network I

Step 1: Create an adjacency matrix

- Graph: Mathematical concept $G = (V, E)$
  - (V: vertices/nodes, E: edges)
  - No visual concept
- Adjacency matrix: Data structure to represent graphs
- Rows and columns represent vertices
- Cells indicate the connection strength between vertices

Table: Adjacency matrix

	A	B	C
A		5	1
B	5		3
C	1	3	

Character Network II

Step 1: Create an adjacency matrix

```
c <- configuration(play,
                   onlyPresence = TRUE,
                   segment = "Scene") %>%
  filterCharacters(play) %>%
  characterNames(play)
mat <- as.matrix(c)

# multiply the matrix with its inverse to create the adjacency matrix
adjMatrix <- mat %*% t(mat)

# add character names
rownames(adjMatrix) <- c$character
colnames(adjMatrix) <- c$character
```

Character Network I

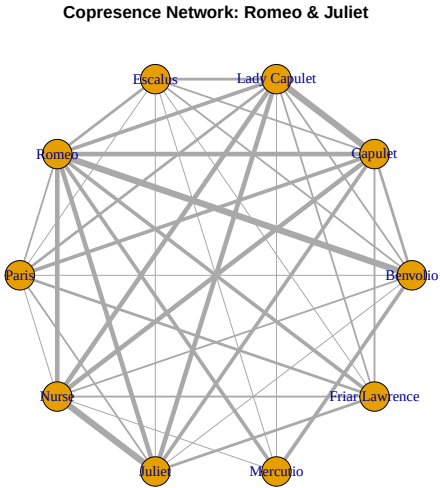
Step 2: Create graph and plot it (with library 'igraph')

```
library(igraph)
# convert the adjacency matrix to a graph object
g <- graph_from_adjacency_matrix(adjMatrix,
                                 weighted=TRUE,
                                 mode="undirected",
                                 diag=FALSE)

# plot it
plot.igraph(g,
     layout=layout_in_circle,
     main="Copresence Network: Romeo & Juliet",
     edge.width=E(g)$weight)
```

Character Network II

Step 2: Create graph and plot it (with library 'igraph')



Hands-On-Session

- Now it is your turn
- Run `loadMeta(loadAllInstalledIds())` to see which plays are there
- Pick a play
- Do some network analysis!
- This works best and is the most fun in small groups

(do not be afraid, you can not break anything)

Subsection 3

Collections and Passive Presence

Collections

- So far: Single plays
- A collection is a set of plays
  - Technically: A vector of ids
- Three relevant functions
  - `installCollectionData()`
  - `loadSets()`
  - `loadSet(...)`

Collections

- So far: Single plays
- A collection is a set of plays
  - Technically: A vector of ids
- Three relevant functions
  - `installCollectionData()`
  - `loadSets()`
  - `loadSet(...)`

```
installCollectionData()
loadSets()
```

Collections

- So far: Single plays
- A collection is a set of plays
  - Technically: A vector of ids
- Three relevant functions
  - `installCollectionData()`
  - `loadSets()`
  - `loadSet(...)`

```
installCollectionData()
loadSets()
```

	size
allegorisches_drama	1
ammenmaerchen	1
arabische_fantasia	1
arbeiterdrama	1
aufklaerung	14
bardiet	1
bauernkomoedie	1
bauernschwank	1
biedermeier	2
buehnenfestspiel	1
buehnenspass	1
buehnenstück	1
buehnenweihfestspiel	1
buergerliches_drama	1
buergerliches_familiengemaelde	1
buergerliches_trauerspiel	11
burleske	1
charakterlustspiel	1
deutsches_lustspiel	1

Collections: Loading a Set

```
# load ids
sud.ids <- loadSet("sturm_und_drang")

# inspect meta data
View(loadMeta(sud.ids))

# load plays as a single QDDrama object
sud.plays <- loadDrama(sud.ids)
```

Character Presence I

- Before: Calculating presence via `configuration()`, and dividing by number of scenes
- Simpler: `presence()` function
  - actives/passives: Number of scenes with active/passive presence
  - presence $= \frac{\text{actives}-\text{passives}}{\text{scenes}}$

```
# load Schillers' Kabale und Liebe
play <- loadDrama("gdc:tz39.0")

# calculate presence for some characters
pres <- presence(play) %>%
  filterCharacters(play) %>%
  characterNames(play)
```

Character Presence II

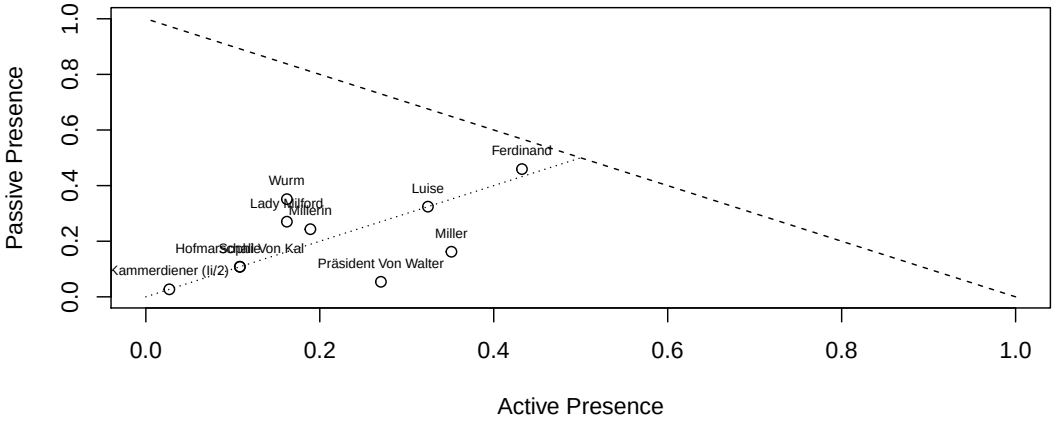
	corpus	drama	character	scenes	actives	passives	presence
5	gdc	tz39.0	Ferdinand	37	16	17	-0.0270270
6	gdc	tz39.0	Hofmarschall Von Kalb	37	4	4	0.0000000
7	gdc	tz39.0	Kammerdiener (Ii/2)	37	1	1	0.0000000
11	gdc	tz39.0	Lady Milford	37	6	10	-0.1081081
12	gdc	tz39.0	Luise	37	12	12	0.0000000
13	gdc	tz39.0	Miller	37	13	6	0.1891892
14	gdc	tz39.0	Millerin	37	7	9	-0.0540541
15	gdc	tz39.0	Präsident Von Walter	37	10	2	0.2162162
16	gdc	tz39.0	Sophie	37	4	4	0.0000000
17	gdc	tz39.0	Wurm	37	6	13	-0.1891892

Visualizing Character Presence I

```
play <- loadDrama("gdc:tz39.0")
pres <- presence(play) %>%
  filterCharacters(play) %>%
  characterNames(play)

# plot points
plot(x=pres$active/pres$scenes,
     y=pres$passive/pres$scenes,
     xlim=c(0,1), xlab="Active Presence",
     ylim=c(0,1), ylab="Passive Presence")
# add labels
text(x=pres$actives/pres$scenes,
     y=pres$passives/pres$scenes,
     labels=substr(pres$character,0,20),
     pos=3,
     cex=0.6)
# add lines
lines(x=seq(0,0.5,0.1),seq(0,0.5,0.1), lty=3)
lines(x=1:0,y=0:1, lty=2)
```

Visualizing Character Presence II



Hands-On-Session

- Now it is your turn
- Run `loadMeta(loadAllInstalledIds())` to see which plays are there
- Pick a play
- Do some network analysis!
- This works best and is the most fun in small groups

(do not be afraid, you can not break anything)

Appendix

Subsection 1

Appendix

Word fields

Word fields: Semantically related words

- Represented as a vector of strings in R
- E.g., love, heart is a word field related to love

Work steps

1. Define a word field: base R, `loadFields()`
2. Apply it to text(s): `dictionaryStatistics()`

Word Fields

Define a word field

Definition of a word field manually on the fly

```
fields <- list(
  # words related to family
  Family=c("marriage", "parents", "ancestors", ...),
  # words related to love
  Love=c("love", "heart", "kiss", ...))
```

Creates a named list of lists

Word Fields

Define a word field: Function ‘loadFields()‘

- Function to load word fields from URLs or files
- Load pre-defined (German) word lists

```
fields <- loadFields(fieldnames=c("Liebe", "Familie"))
```

Returns a named list of lists

Word Fields

Other sources

- Defining word fields manually is not trivial (historic language(s), bias, ...)
- Existing dictionaries can be used as sources
- Enriching fields with distributionally similar words

Word Fields

Application: ‘dictionaryStatistics()‘

```
play <- loadDrama("shakedracor:Rom")
ds <- dictionaryStatistics(play, fields)
```

Returns a table with columns

- `corpus`, `drama`: See above
- `character`: The character id
- one column for each field

Word Fields

Application: ‘dictionaryStatistics()‘

```
play <- loadDrama("shakedracor:Rom")
ds <- dictionaryStatistics(play, fields)
```

Returns a table with columns

- corpus, drama: See above
- character: The character id
- one column for each field

```
##        corpus drama         character Family Love
## 1 shakedracor   Rom   Apothecary_Rom      0    0
## 2 shakedracor   Rom      Benvolio_Rom      0    9
## 3 shakedracor   Rom CITIZENS.0.1_Rom      0    0
```

Word Fields

Normalization

- Different characters have different portions of speech in the play
- Word fields may be differently large

Word Fields

Normalization

- Different characters have different portions of speech in the play
- Word fields may be differently large
- Normalization is important
  - No one fits all mechanic
  - It depends on the research question

Word Fields

Normalization

- Different characters have different portions of speech in the play
- Word fields may be differently large
- Normalization is important
  - No one fits all mechanic
  - It depends on the research question

Two parameters (both can be set to TRUE/FALSE)

- `normalizeByCharacter`
- `normalizeByField`

Word Fields

Normalization

- Different characters have different portions of speech in the play
- Word fields may be differently large
- Normalization is important
  - No one fits all mechanic
  - It depends on the research question

Two parameters (both can be set to TRUE/FALSE)

- `normalizeByCharacter`
- `normalizeByField`

Normalized numbers tend to be very small, but that does not hinder their meaningfulness

Word Fields I

Plotting

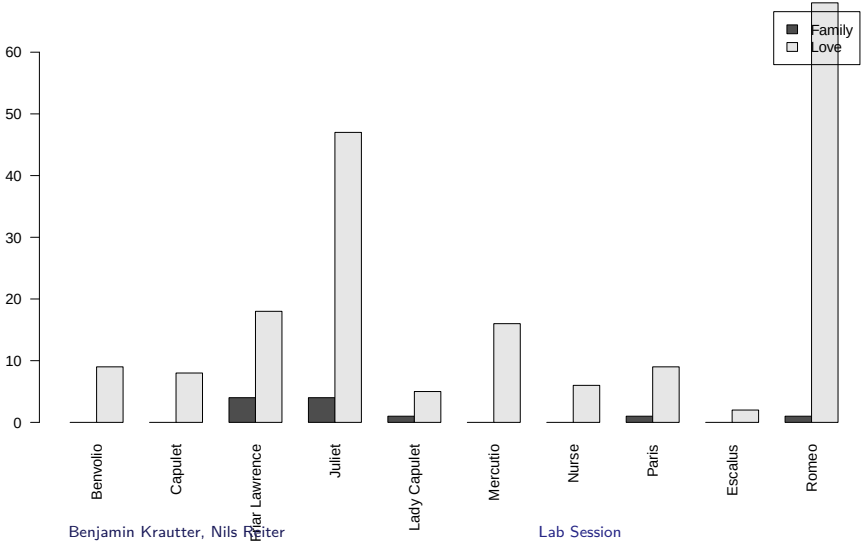
```
ds <- dictionaryStatistics(play, fields) %>%
  filterCharacters(play) %>%
  characterNames(play)

dsm <- as.matrix(ds)

par(mar=c(10,2,1,1))
barplot(t(dsm),
        beside=TRUE,
        names.arg = ds$character,
        legend.text = colnames(dsm),
        las=2)
```

Word Fields II

Plotting



Function utteranceStatistics

```
us <- utteranceStatistics(play)
```

Returns a table with one row for each utterance

- corpus: The collection id
- drama: The play id
- character: the character id
- utteranceBegin: Character position of the first character
- utteranceLength: Portion of this utterance with the total play

(The function View() can be used to get browsable table in RStudio.)

Function `utteranceStatistics` I

Plotting

Function utteranceStatistics II

Plotting

```
play <- loadDrama("shakedracor:Rom")

# get utterance statistics
us <- utteranceStatistics(play) %>%
  # remove uninteresting characters
  filterCharacters(play) %>%
  # replace ids by names
  characterNames(play)

# plot boundaries
par(mar=c(2,7,1,1))
# plot the utterances
stripchart(utteranceLength ~ character,
        data = us,
        las=1,
        pch=20,
        method="jitter")
```