Introduction to R for the Humanities

session 1

Susan Reichelt

ADILT, University of Konstanz

Course organisation

Schedule

09:00 - 10:30	course info, introduction to R, first steps
10:30 - 11:00	coffee break
11:00 - 12:30	data input + practice
12:30 - 13:30	lunch break
13:30 - 15:00	data description + practice
[15:00 - 17:00	*practice time & trouble shooting]*

Course info

sessions

- Today's session provides a general introduction to RStudio, packages and simple data structuring

- session 2: data wrangling
- session 3: data visualisation
- session 4: data documentation, presentation, project

credits

The course is part of the advanced data & information literacy track (block 2) and can be completed with 3 or 6 credits

- **3 credits**: work on 3 tasks throughout the 4 sessions. The data you will be working on is provided in the course workspace on GitHub. The tasks will be given out and can be completed each session.
- **6 credits**: write a markdown report that shows data steps with reference to a specific research question. This should include: basic data management, manipulation, interpretation, communication, contextualisation

what you can expect

- general introduction to R with a focus on diverse data types
- basic data wrangling & visualization using tidyverse
- shifting research process into R (incl. write-ups)

what this course is not

- class on statistics or advanced programming
- class on how quantitative analyses work (starting point is data review)

seminar workflow

The sessions emphasise independent work and thus include many opportunities to practice. You can find data & scripts online starting each session.

If you feel like you need less guidance to complete the tasks, you are welcome to work at your preferred pace - please make sure to allow other students time to ask questions and take their own time. If you are uncertain about the content, feel free to work in teams.

questions?

Data

what kinds of data are we working with in the humanities? 🙂 ✍️ 👂 👀 🌍 🎨 🕰️ 📚 📎

can you think of ways to categorise data?

can you think of ways to categorise data?

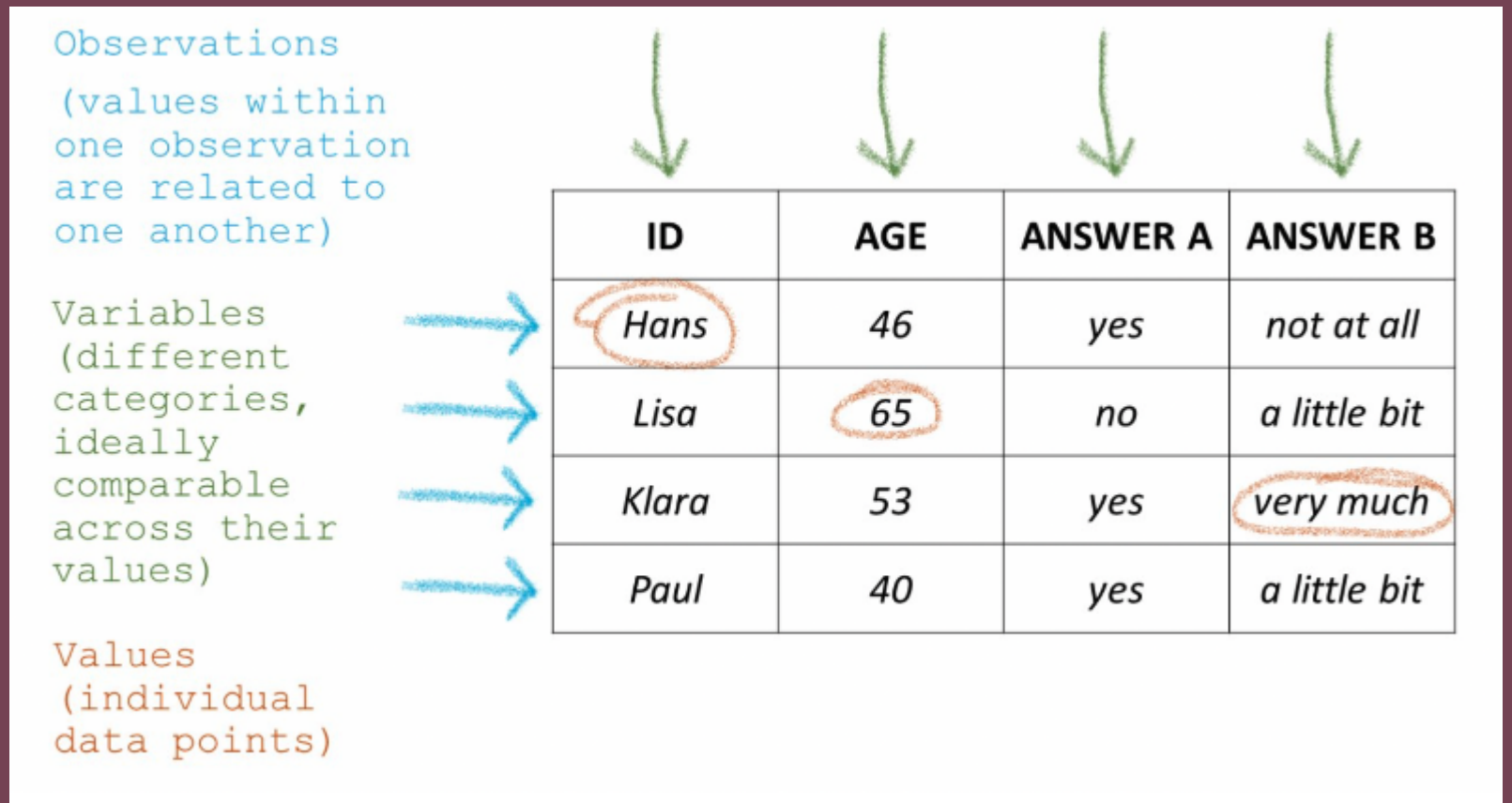
Categorical data		Numerical data	
Nominal	**Ordinal**	**Discrete**	**Continuous**
Data that refers to unordered categories, oftentimes qualitative and character-based (i.e., text). Examples: names, gender, nationality	Data that refers to ordered categories, meaning that the categories are to be interpreted based on an arbitrarily assigned order. (data does not carry any measureable value) Examples: grades, survey responses	Data that is countable and can't be broken down further. Examples: count of students, number of occurrences	Data that is measureable - this is data that you would apply most statistical analyses to. Examples: temperatures over time, distributions, weight

how is data organised?

how is data organised?

observations = rows

variables = columns



Task 1

Describe the data types. Describe what you see in the data:

- observations
- variables
- values

Participant	Year of Birth	Q1	Q2	Q3	Q4
A	1985	hamburger	never	y	5
B	2000	chocolate	sometimes	n	6
C	1998	pizza	never	y	4

Data for the seminar

data included in this seminar

- **politics**: WBL Index
- **general survey data**: European social survey*
- **cultural observation**: Nobel prize awards
- **language**: hedging over 20 years
- **literature**: drama texts, Harry Potter
- **media miscellaneous**: TV dialogue, Great British Bake Off, EuroVision, Spotify
- **history**: political systems
- **ethnography**: mapping

data access

You can find data files on Github. Not all data will be relevant today, so files will be added over time.

Links, etc. to follow in a little bit.

R

you and R

what are your expectations?

what do you want to focus on?

those who've worked with R before, what do you know?

those who've not worked with R before, why now (& what for)?

what is R? what is RStudio?

R: programming language & environment

R Studio/ Posit: IDE (integrated development environment)

Packages can be added to allow for data access, management, and statistics. These can be loaded from CRAN (comprehensive R archive and network) or via development tools. Packages need to be installed **and** loaded.

We'll work primarily with the package collection called tidyverse 🧹

scary disclaimer

R comes with a pretty steep learning curve. However, it's also super flexible and versatile! So, in many ways, working with R has a lot to do with managing expectations.

There is not one right way, not one right script. There're a few rules and many different styles.

Comments are your friends: they save time and nerves. Same goes with proof-reading: make sure to count your brackets!

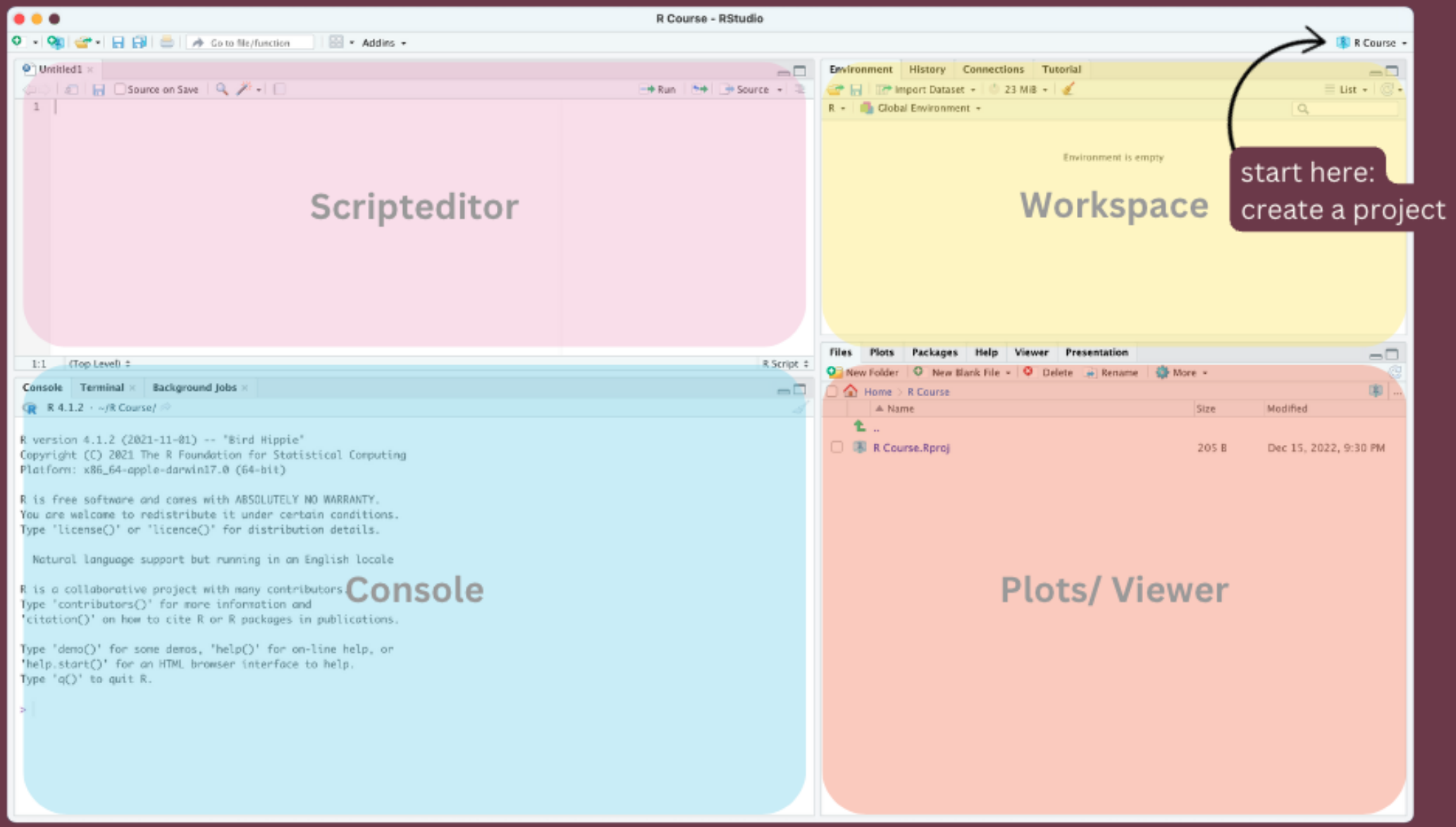
technical check-in

open RStudio.

For the next part of the course, we'll work with the following steps:

1. follow along with the slides first!
2. then, copy slide steps on your own laptop

the layout & start



general best practice

while all commands are executed in the console, it's a good idea to write your code in the script editor.

You can execute your code from the script via the 'run' button or using the shortcut cmd+enter.

#	for commenting. use it!
<-	for creating objects ("this belongs there")
%>%	for function stepsfür Funktionsfolgen ("take and"), from the magrittr package (tidyverse)

Task 2

1. Create a project for the course.
2. Create a script for this session.
3. Adjust your layout if you want.
4. Calculate by writing in & executing from your script:
    i. 2028 divided by 13
    ii. 326.4 divided by 5
    iii. Squareroot (sqrt(x)) from 36 cubed
    iv. Mean of 20, 45, and 1

break 

Task 2 review

2028 divided by 13

▶ Code

```
[1] 156
```

squareroot from 36 cubed

▶ Code

```
[1] 216
```

326.5 divided by 5

▶ Code

```
[1] 65.3
```

mean of 20, 40, and 1

▶ Code

```
[1] 20.33333
```

data input

by data type

- **Numbers (numerical, integer)**
  - this includes NaN (not a number)
  - workspace will identify these as num or int
- **Characters (character, string)**
  - these are placed within quotation marks and can also include numbers ("42" is a character, 42 is a number)
  - workspace will identify these as chr
- **Binary (boolean)**
  - TRUE or FALSE (also T and F)

data as objects in R

- We place data values in objects to keep working with them
- You can find all of your objects in your workspace environment list
- When you define an object, you give the object a name
  - do not start the name with a number
  - do not use the name of a package
  - consider that you need to retype the name a few times, be gentle with yourself

for your script

create an object called a that consists of the value 3.

- Code

create an object called b that conists of the value "house"

- Code

what do you notice in your environment?

If you want to check what's part of your object, you can check in the environment panel or by calling the object:

▶ Code

```
[1] 3
```

The object will be called in the console, so keep an eye on that panel.

Try the same for object b!

You can work with the object instead of the contained value. If you want to make a change to the object, you need to assign the object to its name again - or choose a new name.

```
#work with object
a + 4
```

```
[1] 7
```

```
#does not change the object
a
```

```
[1] 3
```

```
#but it does when you re-assign the name
a <- a + 4
a
```

```
[1] 7
```

data types revisited

You can adjust the data type for some objects!

- as.character(x)
- as.numeric(x)
- as.logical(x)

```
1 a <- as.character(a)
2 a
```

```
[1] "7"
```

Task 3

Observe your environment panel as you work through these steps

- Create object “test” containing the character “1”
- Add 3 to “test” - does it work?
- Change “test” to a numeric object
- Add 3 to “test” (& re-assign object)
- Subtract 4 from “test” (& re-assign object)
- Change “test” to logical object

Task 3 review

- Code

```
[1] FALSE
```

vectors!

Vectors are multiple ordered values of the same type. We need vectors, because data does usually contain more than a single value.

You can create a vector using c( )

▶ Code

```
[1] "house" "cat"   "4"
```

what happens here?

- Code

```
[1] 5 6 7 8 9
```

- Code

```
[1] 4.00 4.25 4.50 4.75 5.00 5.25 5.50 5.75 6.00
```

- Code

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

dataframes!

Data framesare like vertically aligned vectors,

one observation per row, one column per information

Word	POS	WordLen	LogFreq	FuncWord
Die	ART	3	18.7	T
Biene	NN	5	8.97	F
summt	VVFIN	5	7.67	F
laut	ADV	4	13.9	F

building your own dataframe

data frames are what you will work with mostly when going through any analysis of data. You can built a dataframe from vectors using data.frame()

```
1  f <- c(150, 500, 300)
2  g <- c("butter", "flour", "milk")
3
4  recipe <- data.frame(f,g)
```

what do you notice in your environment panel?

data import

working with slightly bigger dataframes

in order for us to start working with bigger dataframes, we can import data into our project space. This can be done either by referring to a file that's already on your computer, or by referring to a file that sits online.

To make it a bit easier, we'll do 2 things:

1. switch to the session 1 script
2. install a package to easily read files into the environment

download the script from my email and copy it into the project space.

open the script by double clicking on the file in your ‘Files’ panel.

For the rest of the session, you can follow along in the script :)

script1.R - let's start!

How this works:

Keep following along on my slides, then execute the corresponding code line from your script. Add your own comments throughout so you remember what you did. I've included tasks throughout that let you test your knowledge.

The 3-credit tasks can be found at the bottom of the script. You don't have to work on them right away, but if you feel like I am moving too slow, go ahead :)

Install package tidyverse

there will be more information about tidyverse tomorrow, right now we need it to import our datasets. Did the install and loading work? The package should appear with a check mark in your packages panel.

All done?
In your script, add the code for installing and loading the praise package. Once it's loaded, type in the following:

```
praise()
```

Importing data from the web

To import our first practice dataframe, we will use the readr package that's part of tidyverse. This package allows us to read files from anywhere on the computer (you only need to know the path) or from online. In the script, you can simply execute the code to avoid typos.

```
1 WBL <- read_csv("https://raw.githubusercontent.com/SusanReichts/ADILT_Rintr
```

Did that work? Now do the same with the other code lines to import the remaining data. Have a look through the data.

lunch 

welcome back. what now?

before we can do anything with the data, we first need to get to know the data a little bit better. For this, we have a couple of functions. We'll use the WBL dataframe as an example, but it's probably a good idea if you have a good look around the other files, too!

What happens here?

```
1 head(WBL)
```

```
# A tibble: 6 × 55
  Economy     EconomyCode ISOCode Region   IncomeGroup ReportYear INDEX
MOBILITY
  <chr>       <chr>       <chr>   <chr>    <chr>             <dbl> <dbl>
<dbl>
1 Afghanistan AFG         AFG     South A… Low income         1971   263
25
2 Afghanistan AFG         AFG     South A… Low income         1972   263
25
3 Afghanistan AFG         AFG     South A… Low income         1973   263
25
4 Afghanistan AFG         AFG     South A… Low income         1974   263
25
5 Afghanistan AFG         AFG     South A… Low income         1975   263
25
```

What happens here?

```
1 str(WBL)
```

```
spc_tbl_ [9,880 × 55] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ Economy
: chr [1:9880] "Afghanistan" "Afghanistan" "Afghanistan" "Afghanistan" ...
 $ EconomyCode
: chr [1:9880] "AFG" "AFG" "AFG" "AFG" ...
 $ ISOCode
: chr [1:9880] "AFG" "AFG" "AFG" "AFG" ...
 $ Region
: chr [1:9880] "South Asia" "South Asia" "South Asia" "South Asia" ...
 $ IncomeGroup
: chr [1:9880] "Low income" "Low income" "Low income" "Low income" ...
 $ ReportYear
: num [1:9880] 1971 1972 1973 1974 1975 ...
 $ INDEX
: num [1:9880] 263 263 263 263 263 263 263 263 263 263 ...
```

Task 4 What do these functions do?

- dim(WBL)
- nrow(WBL)
- ncol(WBL)
- length(WBL)
- tail(WBL)
- names(WBL)
- summary(WBL)
- colnames(WBL)

Indexing data

Indexing describes the process of looking at parts of an object.

```
1 e <- seq(1,20)
2 e[3:5]
```

```
[1] 3 4 5
```

```
1 WBL[35, 4]
```

```
# A tibble: 1 × 1
  Region
  <chr>
1 South Asia
```

```
1 WBL$Region
```

```
 [1] "South Asia"                "South Asia"
 [3] "South Asia"                "South Asia"
 [5] "South Asia"                "South Asia"
 [7] "South Asia"                "South Asia"
 [9] "South Asia"                "South Asia"
```

```
[11] "South Asia"                "South Asia"
[13] "South Asia"                "South Asia"
[15] "South Asia"                "South Asia"
[17] "South Asia"                "South Asia"
[19] "South Asia"                "South Asia"
[21] "South Asia"                "South Asia"
[23] "South Asia"                "South Asia"
[25] "South Asia"                "South Asia"
[27] "South Asia"                "South Asia"
```

You can also use indexing to order vectors within a dataframe.

```
recipe[order(f),]
```

```
    f      g
1 150 butter
3 300   milk
2 500  flour
```

Do you know how we might reverse the order?

Task 5: Indexing data

> **user** with values “bert”, “ernie”, “count”, “oscar”
> **tweets** with values 649, 689, 632, 489
> **followers** with values 364, 659, 126, 986
> **follows** with values 1, 356,476,456
> **face_in_profile** with values T,F,T,T

combine these to a dataframe **SesameTweet**

> Extract the first row
> Extract the 2nd and 3rd column
> Extract the value 476

Bonus: order the dataframe by frequency of followers

practice practice practice

Task 6

Build your own dataframe from vectors - you can freely choose (or make up) your data, but include at least 4 vectors with at least 4 values each. Add it all to a new script.

Think of four quiz questions that include everything we've learned today, from indexing to ordering to maths, etc. Upload your script (without the code for the solutions) to the link below, add your questions and challenge others in the room!

https://shorturl.at/uENQV

Task 7

What do these functions do?

- *min() / max()*
- *nchar()*
- *length()*
- *unique()*

Task 8

Can you guess what the results would be?

```r
sqrt(sum(c(5,3,1)))
```

```
[1] 3
```

```r
max(nchar(c("HGW", "MA", "GSWG")))
```

```
[1] 4
```

```r
 as.character(min(c(9,8,-1)) + max(1:5))
```

```
[1] "4"
```

```r
c(9,3,42)[sqrt(mean(c(7,9,11)))]
```

```
[1] 42
```

Task 9

Task 9 contains the questions for the 3 credits.
To submit your finished tasks, either today or by the end of session 4 at the latest, copy your code, incl. comments, into a new script and send it via email.

Tomorrow:

1. tidyverse!
2. data cleaning
3. data summaries
4. simple data calculations
5. data frame combination, filtering, etc.