Prototyping and User Study

After successful pilot studies with low fidelity paper prototypes, high fidelity prototypes were developed, which can be executed both on desktop PCs and on Smartphones. This document explains the used prototyping tools as well as the different design approaches.

High fidelity prototyping

Shoppinglists with Proto.io

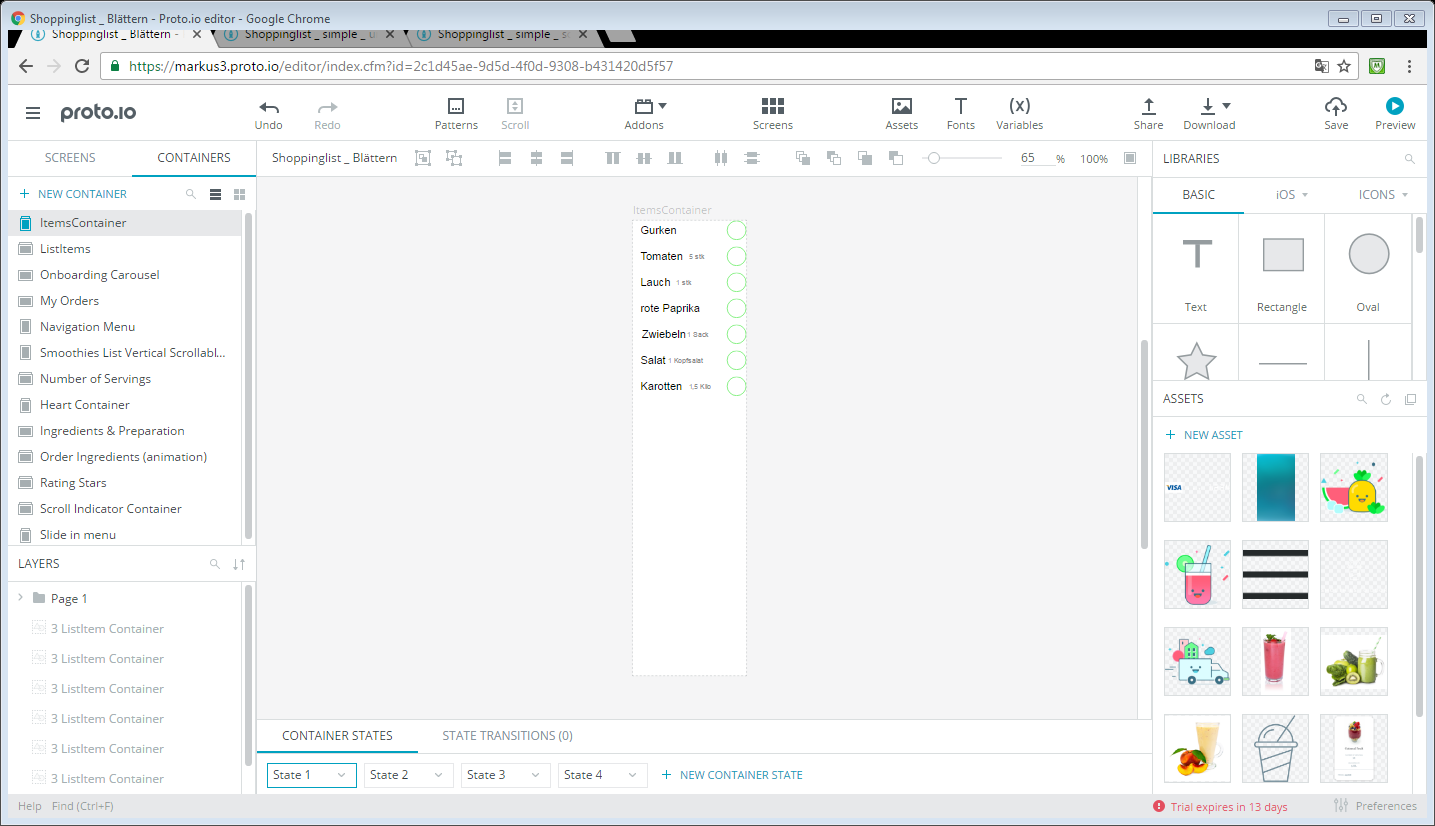
Philipp Riedmann, Thomas Weber, Markus Schnappinger

# Purpose

After the initial paper prototyping with three different approaches (i.e. a sorted list, an unsorted list, and a sorted and visually separated list) our analysis showed that the users liked the sorted lists best. Given the sorting is in alignment with the product placing in the supermarket, the users regarded the shopping list as more efficient than an unsorted list. Following this assumption, we created prototypes with higher fidelity, that enable the user to feel the real smartphone experience while participating in our user study. Hence, we hope to make more detailed observations and gather insights how to design an efficient shopping list application.

# Tools

Widely used for prototyping are the tools Balsamiq (<https://balsamiq.com/>) and Prottapp (<https://prottapp.com/>). Even though they are well documented and a great number of tutorials is available, our tool evaluation showed a lock regarding the possibility to animate items. We do consider this functionality to be essential for the completion of list items. Using paper prototypes, it was not possible for the users to see that they marked a list item as completed. For executable prototypes, we wanted them to really see an effect and therefore needed the animation possibility. Hence, we choose the prototyping app Proto.io (<http://www.proto.io/>). It looks and works very similar to Balsamiq and Prottapp, and offers a quick PDF export and allows simple migration to smartphones.



Unfortunately, Proto.io offers a trial version limited to just 15 days and there is no student version available. There is the possibility to keep one’s projects for a longer period of time on a so called free plan with limited functionality, but information regarding this limitation is quite rare. To ensure that our prototypes are available to the academic advisor and everyone interested, we used the pdf export of proto.io and captured the usage with a video, too.

## Access to the prototypes:

In addition to a video and the pdfs in Google Drive, there is the possibility to experience the developed prototypes and use them for user studies. Each prototype is available via its link given in the following table:

Simple list design with unsorted items: <https://pr.to/PZX2ZR/>

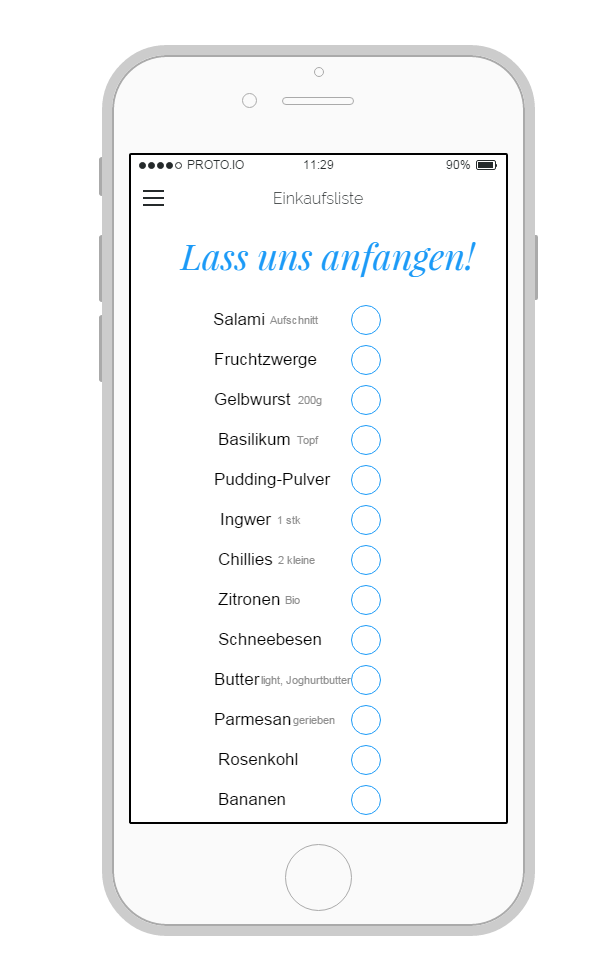
Simple list design with sorted items: <https://pr.to/F180H2/>

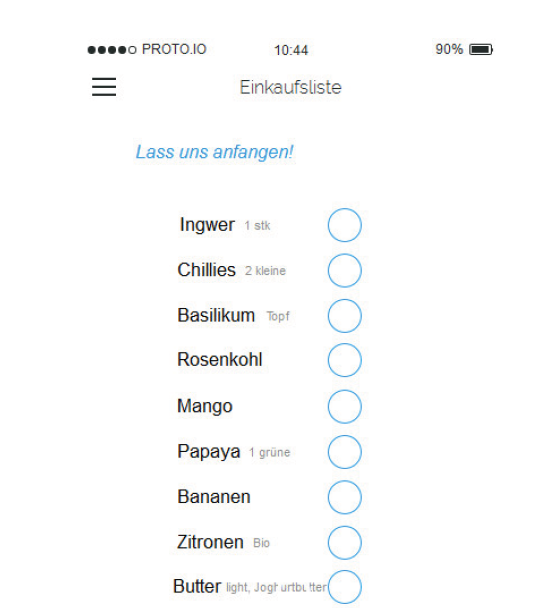
Visually separated list design with grouped items: <https://pr.to/YMTAG5/>

Carousel design with sublists for each category: <https://pr.to/PH8USU/>

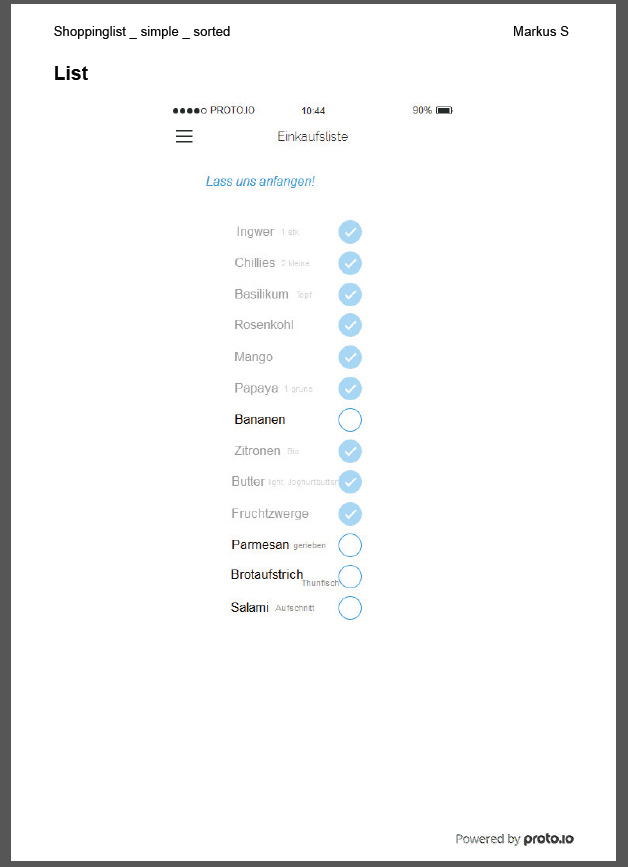
## PDF Export

The exported pdf files are available in Google Drive. They basically show a screens name, the screen and the author. Unfortunately, the layout is not identical with the preview and the live version of the prototypes, especially text size and alignment differ. On the left hand side you can see a smartphone screen as designed and shown in the Webapp, on the right hand side the corresponding screen as appearing in the pdf page.





Due to this unsatisfying representation we recommend to use the executable prototypes instead of the pdf exports. They are available via the links provided in the previous chapter. A page in the pdf report embeds a screen of the prototype along with additional information:



# The Prototypes

Our design approaches were mainly taken from the low fidelity prototypes, as we made good experiences with them. Those were there following:

## Unsorted List

All items supposed to be fetched in the supermarket are listed in an undefined order. This is the design we saw at most shopping lists used by shoppers during our field research in November. This prototype is available via <https://pr.to/PZX2ZR/>

## Sorted List

Items are sorted by categories. As a category we regard items that can usually be found at the same spot in a supermarket, e.g. beverages are one category. This prototype is available via <https://pr.to/F180H2/>

## Sorted and grouped List

Items are sorted by categories and are visually separated from items of different categories. This prototype is available via <https://pr.to/YMTAG5/>

## Sublists for each category

Although our results with the three above list types in the pilot study was satisfying, we added one new type for the executable prototypes. Items are sorted and grouped by lists, but only a sub-list of the actual category is shown to the user while other categories are hidden. This prototype is available via <https://pr.to/PH8USU/>

# Evaluation of the prototypes

To evaluate the designed prototypes we used Nielsen’s 10 usability heuristics following <https://www.nngroup.com/articles/ten-usability-heuristics/> . For each topic the evaluators filled in this form:

1. **Which Prototype?**
   * 1. **Problem:** <text>
     2. **Occurrence:** <text>
     3. **Frequency:** <text>
     4. **Effect:** <text>
     5. **Grading: [1..4]**
     6. **Possible Solution:** <text>

If an issue was identified by more than one evaluator, the grading numbers are shown separated by commas and the texts are merged.

## Prototype Adaptions

Solutions that are feasible for Proto.io Prototypes were applied to the prototypes ‘unsorted list’ and ‘sorted and grouped list’. A list of the concrete changes and screen captures can be found in the folder ‘second iteration’ along with a demonstration video.

## Evaluation Results

1. **Visibility of system status**
   1. **Unsorted List**
      1. **Problem:** Order and additional information seem arbitrary
      2. **Occurrence:** For most list items
      3. **Frequency:** often
      4. **Effect:** Confusing to the user
      5. **Grading: 1**
      6. **Possible Solution:** Make sorting explicit, make additional information consistent
      7. **Note:** Not resolved, as the arbitrary additional information is designed to show that list entries can be augmented with nearly every kind of information
   2. **Sorted List**
      1. **Problem:** Order is implicit only
      2. **Occurrence:** on list screen
      3. **Frequency:** Always
      4. **Effect:** Benefit of sorting only when reading the list in detail, thus greater mental workload
      5. **Grading: 1,2**
      6. **Possible Solution:** Make sorting and categories explicit
2. **Match between system and the real world**
   1. **Unsorted and sorted Simple List**
      1. **Problem:** categories are not explained to the user
      2. **Occurrence:**  on the goodbye screen the user gets information about the number of categories
      3. **Frequency:**  at the end of each shopping
      4. **Effect:**  the user is confused
      5. **Grading: 1**
      6. **Possible Solution:** delete the term from the goodbye screen
   2. **Sorted simple list, sorted grouped list** 
      1. **Problem:** User is not informed about the sorting criteria
      2. **Occurrence:** on list screen
      3. **Frequency:** often
      4. **Effect:** User can be confused if the sorting does not seem intuitive to him
      5. **Grading: 1,2**
      6. **Possible Solution:** Add information about sorting criteria
   3. **Grouped simple list**
      1. **Problem:** User is not informed about the meaning of different colors
      2. **Occurrence:** on list screen
      3. **Frequency:** often
      4. **Effect:** user can be confused if he does not recognize the different categories
      5. **Grading: 1**
      6. **Possible Solution:** Add heading for different categories
   4. **All Prototypes**
      1. **Problem:** Items are greyed out upon checking
      2. **Occurrence:** When checking an item
      3. **Frequency:** Always
      4. **Effect:** Real world shopping lists do not behave this way. Coupled with low contrast between greyed out and original state, makes detection of checked items challenging
      5. **Grading: 3**
      6. **Possible Solution:** Strikeout of checked items
   5. **All categorized variants**
      1. **Problem:** Inserted items are not appended at the end of the total list
      2. **Occurrence:** When adding an item
      3. **Frequency:** Often
      4. **Effect:** Loss of orientation
      5. **Grading: 4**
      6. **Possible Solution:** Animate the new item or the viewport
      7. **Note:** Not resolved due limitations in the Prototyping tool. At most there is the possibility to have exactly one list item at a predefined position with content that was set by the user.
3. **User control and freedom**
   1. **All prototypes**
      1. **Problem:** Undo is not supported for checking items.
      2. **Occurrence:**  On list screen
      3. **Frequency:** To check items is the main function.
      4. **Effect:** Users cannot un-check list entries they checked by mistake
      5. **Grading: 4**
      6. **Possible Solution:** Add undo button next to checked items
   2. **All prototypes**
      1. **Problem:** There is no return possibility from the goodbye screen to the list screen.
      2. **Occurrence:**  On the goodbye screen
      3. **Frequency:** seldom
      4. **Effect:** Users cannot un-check list entries they checked by mistake
      5. **Grading: 2**
      6. **Possible Solution:** Add return button to goodbye screen
4. **Consistency and standards**
   1. **All categorized variants**
      1. **Problem:** Shopping Cart Item might not be the optimal choice
      2. **Occurrence:** On the exit/finish button
      3. **Frequency:** Always
      4. **Effect:** Raise expectation of online shopping functionality
      5. **Grading: 1**
      6. **Possible Solution:** Use different icon
5. **Error prevention**
   1. **All prototypes**
      1. **Problem:** List items can be checked by mistake
      2. **Occurrence:**  Checking list items on list screen
      3. **Frequency:**  seldom
      4. **Effect:**  The user may not buy the mistakenly checked item
      5. **Grading: 3**
      6. **Possible Solution:** 
         1. Add confirmation dialog
         2. Increase padding between items
         3. Add undo function, so the mistake is reversible
6. **Recognition rather than recall**
   1. **All prototypes**
      1. **Problem:** The user is not instructed that items can be checked and how to do so
      2. **Occurrence:**  on list screen
      3. **Frequency:** often
      4. **Effect:**  user can easily forget an item
      5. **Grading: 1**
      6. **Possible Solution:** Add instructions to the welcome screen
   2. **All prototypes except grouped list**
      1. **Problem:** There is no hint that items might be hidden at the bottom and that the list can be scrolled vertically
      2. **Occurrence:**  on list screen
      3. **Frequency:** often
      4. **Effect:**  user can easily forget an item
      5. **Grading: 4**
      6. **Possible Solution:** Add hint that there are more items below the bottom
7. **Flexibility and efficiency of use**
   1. **All prototypes except unsorted simple list**
      1. **Problem:** User cannot change the sorting
      2. **Occurrence:** on list screen
      3. **Frequency:** often
      4. **Effect:** user can be annoyed if the sorting is wrong. Inadequacy for the users’ needs
      5. **Grading: 3, 3**
      6. **Possible Solution:** 
         1. Add reordering functionality e.g. via drag and drop
         2. Add menu to select from different sorting
      7. **Note:** The sorting in the prototypes is created with Wizard of Oz functions and is meant to be in perfect alignment with the sorting in the supermarket.
   2. **All prototypes**
      1. **Problem:** No expert-user interactions available
      2. **Occurrence:** N/A
      3. **Frequency:** Always
      4. **Effect:** No performance improvement
      5. **Grading: 1**
      6. **Possible Solution:** Checking of multiple items by swiping over their checkboxes
      7. **Note:** Not resolved due to limitations of the Prototyping tool.
8. **Aesthetic and minimalist design**
   1. **All prototypes**
      1. **Problem:** Welcome screen is not minimalistic
      2. **Occurrence:** Welcome screen
      3. **Frequency:** every time the screen is shown
      4. **Effect:**  user is disturbed by unnecessary information
      5. **Grading: 2**
      6. **Possible Solution:** remove unnecessary information
   2. **All prototypes**
      1. **Problem:** Goodbye screen is not minimalistic
      2. **Occurrence:**  Goodbye screen
      3. **Frequency:** every time the screen is shown
      4. **Effect:**  user is disturbed by unnecessary information
      5. **Grading: 2**
      6. **Possible Solution:** remove unnecessary information
9. **Help users recognize, diagnose, and recover from errors**
10. **Help and documentation**
    1. **All prototypes**
       1. **Problem:**  There is no help offer
       2. **Occurrence:** every screen
       3. **Frequency:** every time
       4. **Effect:** A confused user can get no help and a new user gets no instructions
       5. **Grading: 4 , 4**
       6. **Possible Solution:**  Add help button with instructions to each page

# User study with High Fidelity Prototypes

We then incorporated the results of the heuristic evaluation into the original Hi Fi prototypes in another iteration. The changed attributes are explained in detail in the document `High Fidelity Prototyping II`. These prototypes are the basis for a user study to assess their usability and general support for grocery shopping.

## Evaluation Hypotheses

For this user study we vary the way the shopping list is presented to the user in order to evaluate the developed prototypes. There were considerations to vary the availability of features like the undo function as well. Given the exponential increase of tests needed to evaluate not only presentation forms but also features, we will only focus on the different list presentations.

With varying the prototype used we assume that:

* The unsorted list requires the highest number of interaction with the device
* The unsorted list requires the longest distance walked
* The unsorted list requires the longest time
* The unsorted list produces the highest error rate
* Grouped lists require the lowest number of interactions
* Swipe-able sublists require the shortest distance walked
* Swipe-able sublists require the shortest time
* Swipe-able sublists produce the lowest error rate

Considering the environment, we assume that:

* Number of interactions correlates with interaction time
* Long distance walked correlates with long time
* Errors impair the fun

## Evaluation Variables

Independent variables for the user study are the prototype used. The granularity of categories / the number of categories as well as the total list length are not considered as independent variables as they are not varied.

Dependent variables are

* Fun and Preference (measured with a Questionnaire)
* Usability and Aesthetics (AttrakDiff questionnaire et al.)
* Interaction frequency (counted by examiner or device, e.g. eye tracker)
* Error rate (counted by examiner)
* Total time (measured by examiner)
* Total distance walked (measured by a step counter)

## Study setting

The study takes place in a simulated supermarket already used for paper prototyping. Throughout the room there were scattered labeled stations/categories as found in a supermarket. Each station had four to eight items usually found under that category. For this study, it is also possible to evaluate the prototypes in a real supermarket.

Due to our limited resources and therefore limited test persons, the study is meant to be within subject, meaning every tester tries every prototype in different order. This will be latin squared in order to avoid unwanted effects occurring from the test order.

To evaluate the results t-tests are used. As t-tests are allowed only for studies with two levels for the independent variable, our study is split into three sub-studies. The first study uses prototypes ‘random list’ and ‘grouped and sorted list’. The second study uses the prototypes ‘random list’ and ‘swipe-able sublists’ while the third study compares ‘grouped and sorted list’ and ‘swipe-able sublists’.

## Study process

The general process will consist of six steps. After welcoming the tester and explaining the study purpose, they are asked to give their permission that their actions can be recorded by the examiners. After that, the tools are calibrated. Then the actual experiment starts and the testers are told which task they have to complete. Their performance is recorded. For our within-groups study, this will be repeated until each tester has used both prototypes and answered the questionnaires.

## External factors

As external factors we consider the order of the categories in the list and in the supermarket. The users’ usual shopping behavior can influence the study, too. Items he is used to buy might be more easily remembered than exotic products.

## Questionaires

In the following, the questionnaires used to measure the dependent variables fun and ease of use can be found. In addition, forms about general tech affinity and shopping behavior are included as well as a survey on demographics. Participants are asked to evaluate each of the provided sentences with grades from 1 (`I totally agree`) to 5 `I totally disagree`.

Usability is measured with the AttrakDiff[[1]](#footnote-1) framework, where the participants are asked to evaluate the prototype with the following form:

With the help of the word pairs please enter what you consider the most appropriate description for the prototype used?

Adjective Aadjective B

Word pairs used are for example:

Human technical

isolating connective

pleasant unpleasant

inventive coonventional

simple complicated

professional unprofessional

ugly attractive

practical impractical

likeable disagreeable

cumbersome straightforward

stylish tacky

predictable unpredictable

cheap premium

alienating integrating

brings me closer to people separates me from people

unpresentable presentable

rejecting inviting

unimaginative creative

good bad

Fun is measured with this questionnaire[[2]](#footnote-2):

* Do you work with the program without someone telling you to?
* Would you like to work with the program when other household members can decide for themselves what to do?
* Do you think it is boring to work with the program?
* When you started working with the program, did you want to continue working with it?
* Do you think your friends would like the program?
* Do you think the program is childish?
* Is the program too difficult to play with?
* When you have worked with the program once, does it remain fun?
* Do you enjoy yourself when you are working with the program?
* Does the program contain many surprises?
* Would you like to work with the program more often?
* Do you perform well on the exercises in the program?
* Would you like to have the program at home?
* Do you make many mistakes while you are working with the program?

At the end of these two questionnaires there are open questions allowing a semi-structured interview:

Would you like to have an undo function?

Which prototype would you prefer and why?

For demographics variables we collect information about:

* Age
* Gender
* Marital Status
* Household size
* Household income
* Do you live in the countryside or in the city?
* What is your professional situation (student, full time employee, part time employee, retired, not working)?

To get to know the participants and get background knowledge to evaluate their results properly, we collect information about their shopping behavior as well as their tech affinity. This information allows us to evaluate the need for a shopping list app in general and is less related to the specific prototypes. This questionnaire is meant to be answered after the participant has finished all three runs.

Tech affinity is determined with the following questionnaire:[[3]](#footnote-3)

* I stay informed about electronical devices
* I love to own new electronical devices
* I am excited when a new electronical device enters the market
* I like to go to electronic shops
* I like to play around with a new electronical device
* I know the most features of my electronical devices
* I have or would have comprehension problems while reading magazines about electronical devices
* I usually do not have problems while learning how to handle a new electronical device
* Electronical devices help to retrieve information
* electronical devices enable a high living standard
* Electronical devices make the user independent
* Electronical devices help to handle every day live
* Electronical devices reduce personal contact between human beings
* Electronical devices make things complicated
* Electronical devices lead to illnesses
* Electronical devices enforce mental impoverishment

Grocery Shopping behavior is evaluated with the following sentences: [[4]](#footnote-4)

* How often do you do grocery shopping?
  + Many times a week, once a week, once every two weeks, one a month, other
* Which transport do you use the most often to do grocery shopping:
  + Car, Bus, by foot, Bike, other
* How long do you spend doing your grocery shopping on average per week, transportation included?
  + Less than an hour, two hours, three hours, more than three hours
* How do you choose were you will do your grocery shopping? (many answers pos.)
  + Proximity, Price, Products choice, Quality, other
* How do you choose your products? (many answers possible)
  + Brand, Price, Quality, Habit, other

## Study Evaluation

As mentioned above, our study is split into three studies, which evaluate two prototypes each. This allows the use of paired t-tests. A typical paired t-Test shows whether there is a significant difference between two prototypes in one aspect. For example, whether there is a difference in the time used between the grouped prototype and the unsorted prototype. To exercise this evaluation, we create sample datasets for all three studies and evaluate them. We focus on a small subset of the designed questionnaires and choose the ATT section of the AttrakDiff. This section measures general attractiveness of the evaluated program and contains 7 wordpairs:

Unpleasant <-> Pleasant ugly <-> attractive disagreeable <-> likeable

good <-> bad rejecting <-> inviting good <-> bad

repelling <-> appealing discouraging <-> motivating

The score of each attribute is calculated with the following form for each word pair:

Positive attributenegative attribute

+3 +2 +1 0 -1 -2- 3

To create example datasets and evaluate them, the R framework is used. The implementation is attached here:

numParticipants = 10

# Data generation helpers

NormalN = function(mean=0, sd=1) sapply(

as.integer(rnorm(numParticipants, mean=mean, sd=sd)),

function(x) {if (x < 0) 0 else x }

)

NormalRange = function(min, max, mean=0, sd=1) sapply(

as.integer(rnorm(numParticipants, mean=mean, sd=sd)),

function(x) {if (x < min) min else {if (x>max) max else x }}

)

Normal3 = function(mean=0, sd=1) NormalRange(-3, 3, mean, sd)

ttest = function(l, val, inverse=FALSE) {

res = t.test(l$a[val][[1]],l$b[val][[1]], paired=TRUE)

m1 = mean(l$a[val][[1]])

m2 = mean(l$b[val][[1]])

data.frame(attr=val, res$statistic, res$p.value, mean1=m1, mean2=m2, rating=rating(res$p.value, m1, m2, inverse))

}

rating = function(p, mean1, mean2, inverse){

ifelse(p > 0.05, "NO significant difference", ifelse(ifelse(inverse, mean1 < mean2, mean1 > mean2),"1) is significatly better than 2)","2) is significatly better than 1)"))

}

ttests = function(a,b){

l = list(a=a,b=b)

rbind(

ttest(l, "errors", TRUE),

ttest(l, "time", TRUE),

ttest(l, "distance", TRUE),

ttest(l, "interactions", TRUE),

ttest(l, "usability", FALSE),

ttest(l, "fun", FALSE)

)

}

ttestsAD = function(a,b){

l = list(a=a,b=b)

rbind(

ttest(l, "pleasant"),

ttest(l, "attractive"),

ttest(l, "likeable"),

ttest(l, "inviting"),

ttest(l, "good"),

ttest(l, "appealing"),

ttest(l, "motivating")

)

}

#-------------------------------------------

# A: random list vs. grouped and sorted list

#-------------------------------------------

# random list

#------------

a1 = data.frame(

errors = NormalN(2, 2), # number of errors

time = NormalN(600, 100)+100, # time in seconds

distance = NormalN(100, 40)+30, # distance in m

interactions = NormalN(5, 2), # number of interactions,

usability = Normal3(-1, 2), # or attrakdiff

fun = Normal3(-2, 2)

)

# attrakdiff (each -3 to +3)

attrakDiff1 = data.frame(

pleasant = Normal3(-2, 2),

attractive = Normal3(-2, 2),

likeable = Normal3(-2, 2),

inviting = Normal3(-2, 2),

good = Normal3(-2, 2),

appealing = Normal3(-2, 2),

motivating = Normal3(-2, 2)

)

# grouped and sorted list

#------------------------

a2 = data.frame(

errors = NormalN(0, 2), # number of errors

time = NormalN(200, 80)+100, # time in seconds

distance = NormalN(50, 30)+30, # distance in m

interactions = NormalN(2, 2), # number of interactions,

usability = Normal3(2, 2), # or attrakdiff

fun = Normal3(2, 2)

)

# attrakdiff (each -3 to +3)

attrakDiff2 = data.frame(

pleasant = Normal3(2, 2),

attractive = Normal3(2, 2),

likeable = Normal3(2, 2),

inviting = Normal3(2, 2),

good = Normal3(2, 2),

appealing = Normal3(2, 2),

motivating = Normal3(2, 2)

)

studyA = function(){

print("Study A:")

print(" 1) random list vs.")

print(" 2) grouped and sorted list")

print("")

ttests(a1,a2)

}

attrakDiff = function(){

print("Study A - AttrakDiff:")

print(" 1) random list vs.")

print(" 2) grouped and sorted list")

print("")

ttestsAD(attrakDiff1,attrakDiff2)

}

#-------------------------------------------

# B: random list vs. swipe-able sublists

#-------------------------------------------

# random list

#------------

b1 = data.frame(

errors = NormalN(2, 2), # number of errors

time = NormalN(600, 100)+100, # time in seconds

distance = NormalN(100, 40)+30, # distance in m

interactions = NormalN(5, 2), # number of interactions,

usability = Normal3(-1, 2), # or attrakdiff

fun = Normal3(-2, 2)

)

# swipe-able sublists

#------------------------

b2 = data.frame(

errors = NormalN(1, 2), # number of errors

time = NormalN(250, 80)+100, # time in seconds

distance = NormalN(80, 30)+30, # distance in m

interactions = NormalN(5, 2), # number of interactions,

usability = Normal3(1, 2), # or attrakdiff

fun = Normal3(3, 2)

)

studyB = function(){

print("Study B:")

print(" 1) random list vs.")

print(" 2) swipe-able sublists")

print("")

ttests(b1,b2)

}

#----------------------------------------------------

# C: grouped and sorted list vs. swipe-able sublists

#----------------------------------------------------

# grouped and sorted list

#------------

c1 = data.frame(

errors = NormalN(0, 2), # number of errors

time = NormalN(200, 80)+100, # time in seconds

distance = NormalN(50, 30)+30, # distance in m

interactions = NormalN(2, 2), # number of interactions,

usability = Normal3(2, 2), # or attrakdiff

fun = Normal3(2, 2)

)

# swipe-able sublists

#------------------------

c2 = data.frame(

errors = NormalN(1, 2), # number of errors

time = NormalN(250, 80)+100, # time in seconds

distance = NormalN(80, 30)+30, # distance in m

interactions = NormalN(5, 2), # number of interactions,

usability = Normal3(1, 2), # or attrakdiff

fun = Normal3(3, 2)

)

studyC = function(){

print("Study C:")

print(" 1) grouped and sorted list vs.")

print(" 2) swipe-able sublists")

print("")

ttests(b1,b2)

}

1. http://www.attrakdiff.de [↑](#footnote-ref-1)
2. http://dare.ubvu.vu.nl/handle/1871/9782 [↑](#footnote-ref-2)
3. https://www.zmms.tu-berlin.de/fileadmin/f25/dokumente/8BWMMS/13.3-Karrer.pdf [↑](#footnote-ref-3)
4. http://www.marketest.co.uk/market-research-questionnaire/8/grocery-shopping-home-delivery [↑](#footnote-ref-4)