

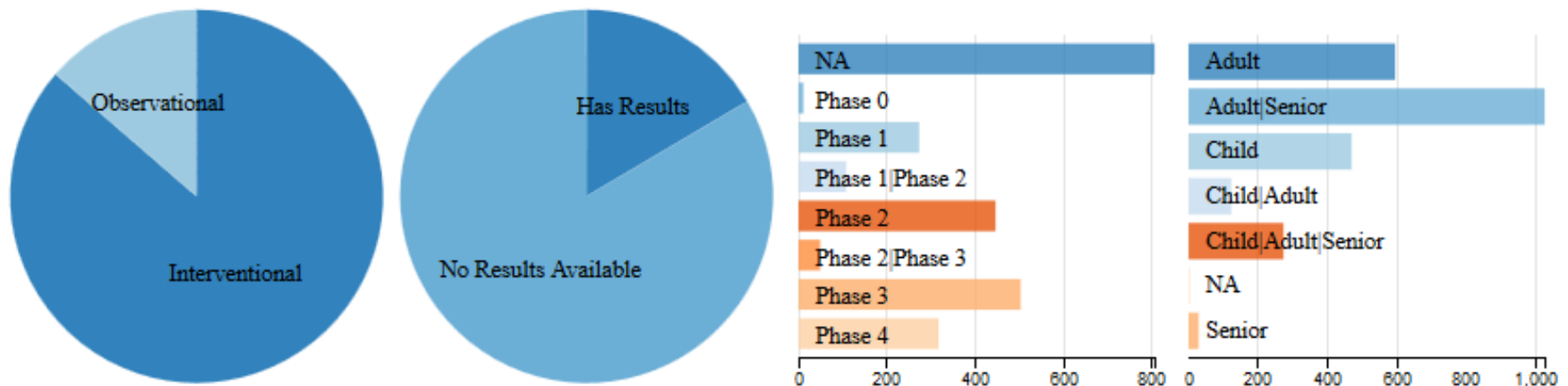
Create interactive web graphics out of your SAS or R datasets

HMS Analytical Software GmbH - Dr. P. Warnat

PhUSE Conference 2015, Vienna

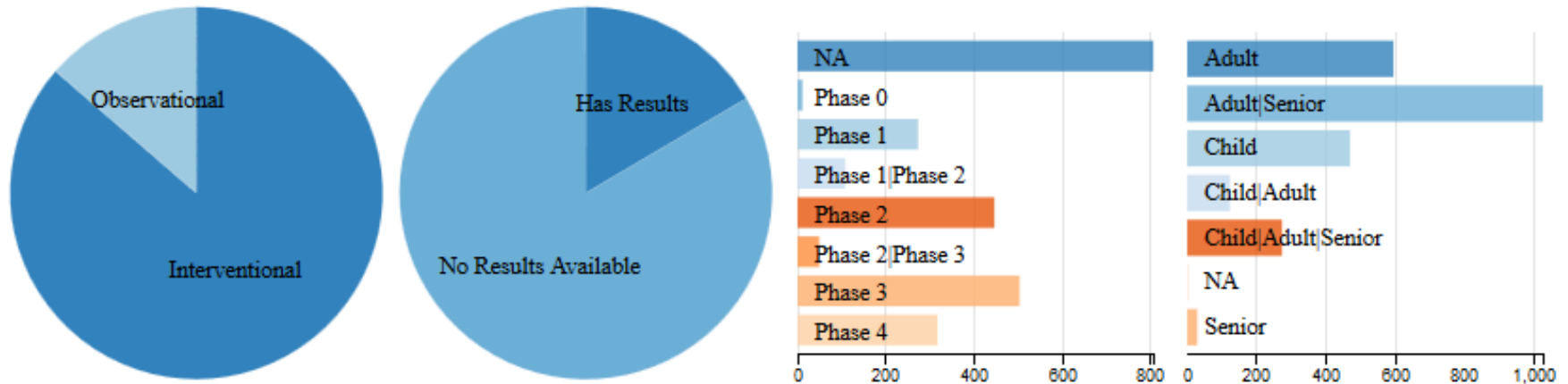
Demo for interactive web graphics

Descriptive data for trials found by search term “Influenza” on ClinicalTrials.gov at 13/August/2015



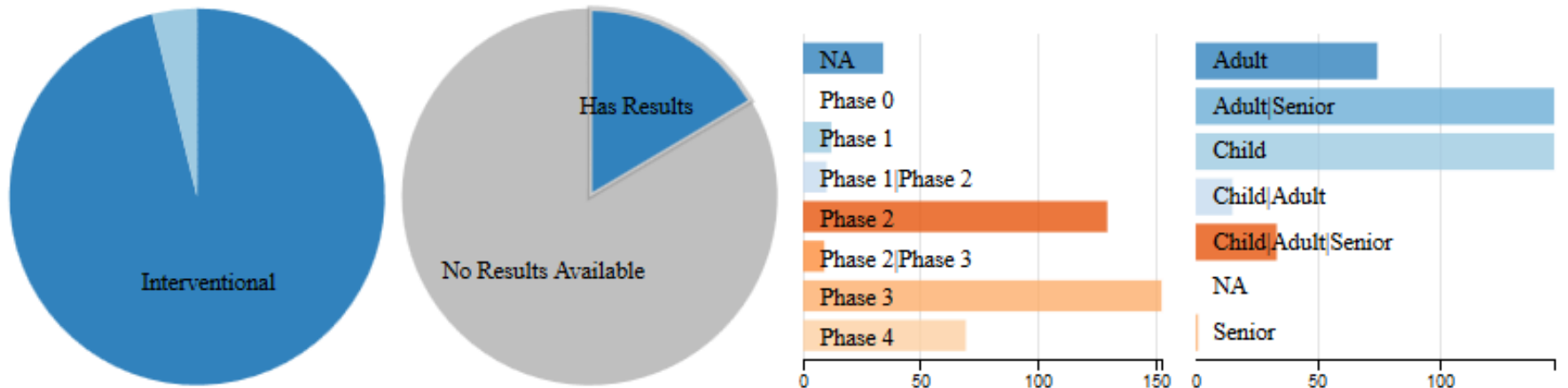
2,513 selected out of 2,513 records | [Reset All](#)

Panel before interaction:



2,513 selected out of 2,513 records | [Reset All](#)

Panel after interaction:

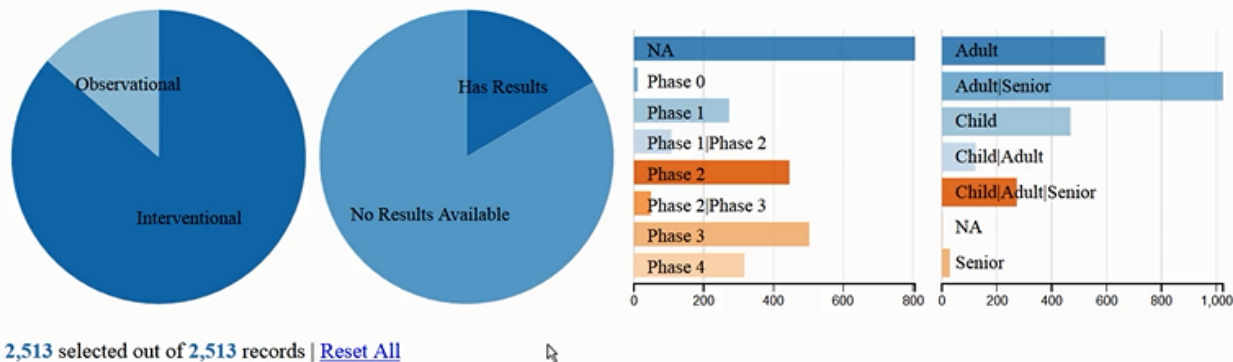


415 selected out of 2,513 records | [Reset All](#)

Demo video

Demo for interactive web graphics - data from ClinicalTrials.gov

Descriptive data for studies found by search term "Influenza" at 13/August/2015



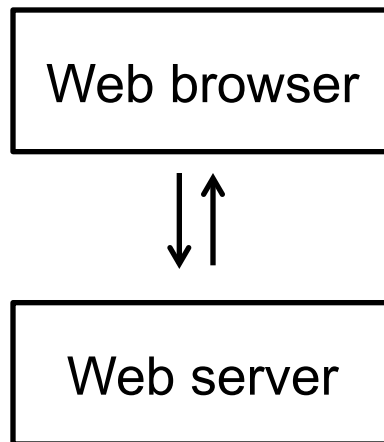
dc.js - Dimensional Charting Javascript Library

- **dc.js is a javascript charting library allowing highly efficient exploration on large multi-dimensional datasets**
- **based on libraries d3.js and crossfilter.js**
- **<http://dc-js.github.io/dc.js/>**



Used Components and Files

- **Simplest scenario: data as flat file on web server**



components

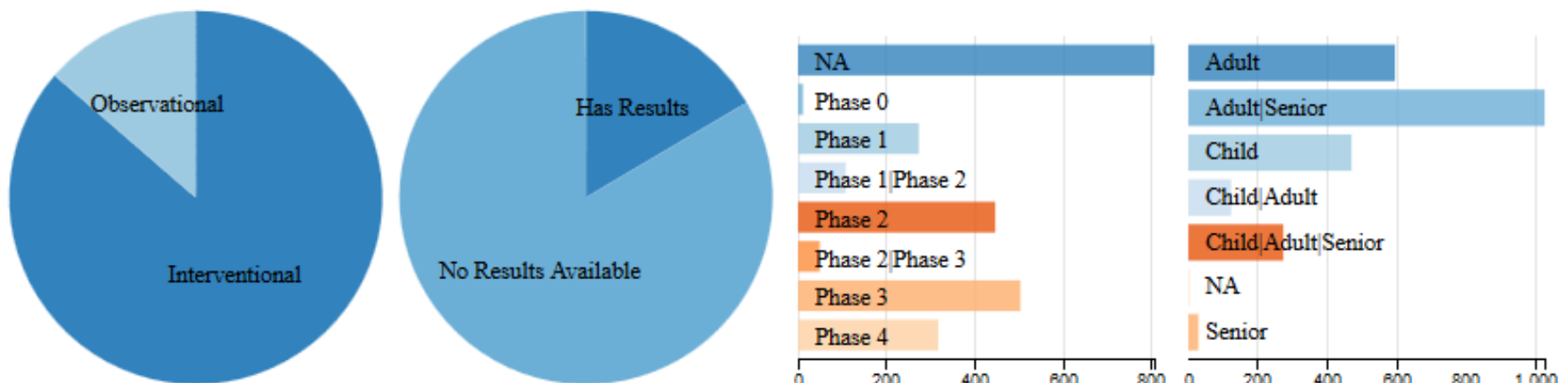
```
index.html
- css
    dc.css
    style.css
- data
    study_fields.tsv
- js
    crossfilter.min.js
    d3.min.js
    dc.min.js
```

file tree on web server

Summary of index.html

- HTML Head with title definition and links (imports) of CSS files
- HTML Body:
 - Headlines definitions
 - several Div-Blocks for definition of the different panel elements
 - links (imports) of js libraries
 - custom Java Script code to read in data and define the diagrams

Descriptive data for trials found by search term “Influenza” on ClinicalTrials.gov at 13/August/2015



2,513 selected out of 2,513 records | [Reset All](#)

Summary JavaScript code

- **function createCharts:**

used to define the interactive diagrams using dc.js, this function is designed to be used as a callback of an d3 data import function

- **function replaceMissingWithMarkerNA:**

simple replacement of empty strings with string NA, used during data import

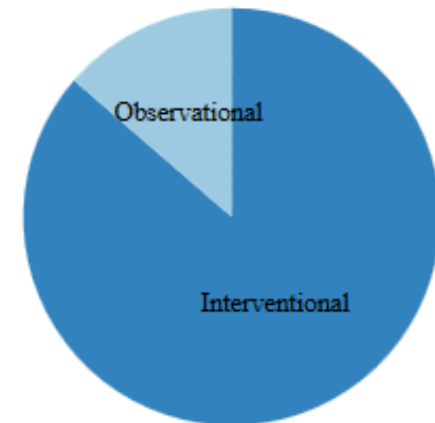
- **a call to the function d3.tsv:**

a function that reads tabular separated files, allows definition of preprocessing (here a call to replaceMissingWithMarkerNA) and which calls function createCharts when data is read and available

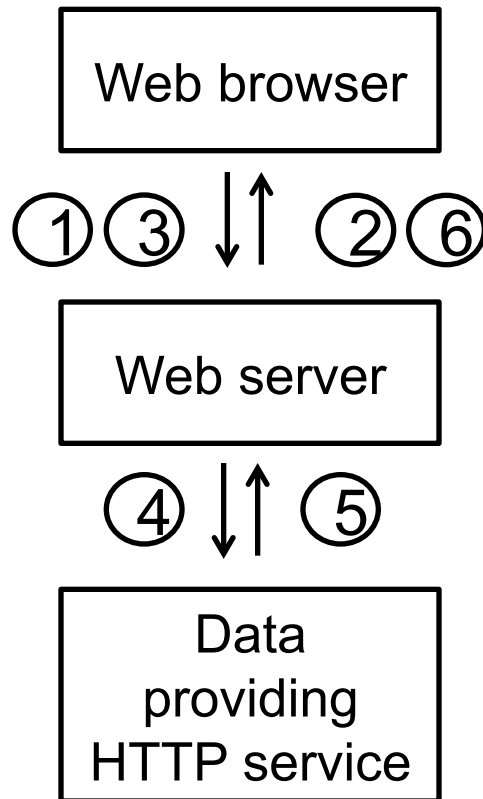
JavaScript function: createCharts

steps necessary to define one pie chart

```
// the variable data contains the tabular input data read by function d3.tsv as a list of JSON objects
// the crossfilter function takes a list of JSON objects, and creates a crossfilter object
var crf          = crossfilter(data);
// we define the column 'types' of the data as a dimension, which can be used to group or filter data
var typesDimension = crf.dimension(function(d) {return d.types;});
// the group function constructs a new grouping for the given dimension, according to a specified groupValue
// function. The groupValue function is optional; if not specified, as it is the case here, the number of records
// per group will be counted.
var typesGroup     = typesDimension.group();
// define a pie chart; the referenced HTML DIV element defines where on the page the diagram will be located
// width, height and radius define the size of the diagram and dimension and group define the shown information
var typesPieChart = dc.pieChart('#chart-pie-types');
    typesPieChart
        .width(230)
        .height(230)
        .radius(110)
        .dimension(typesDimension)
        .group(typesGroup)
    ;
// finally, a function call to render the diagram on the page
dc.renderAll();
```

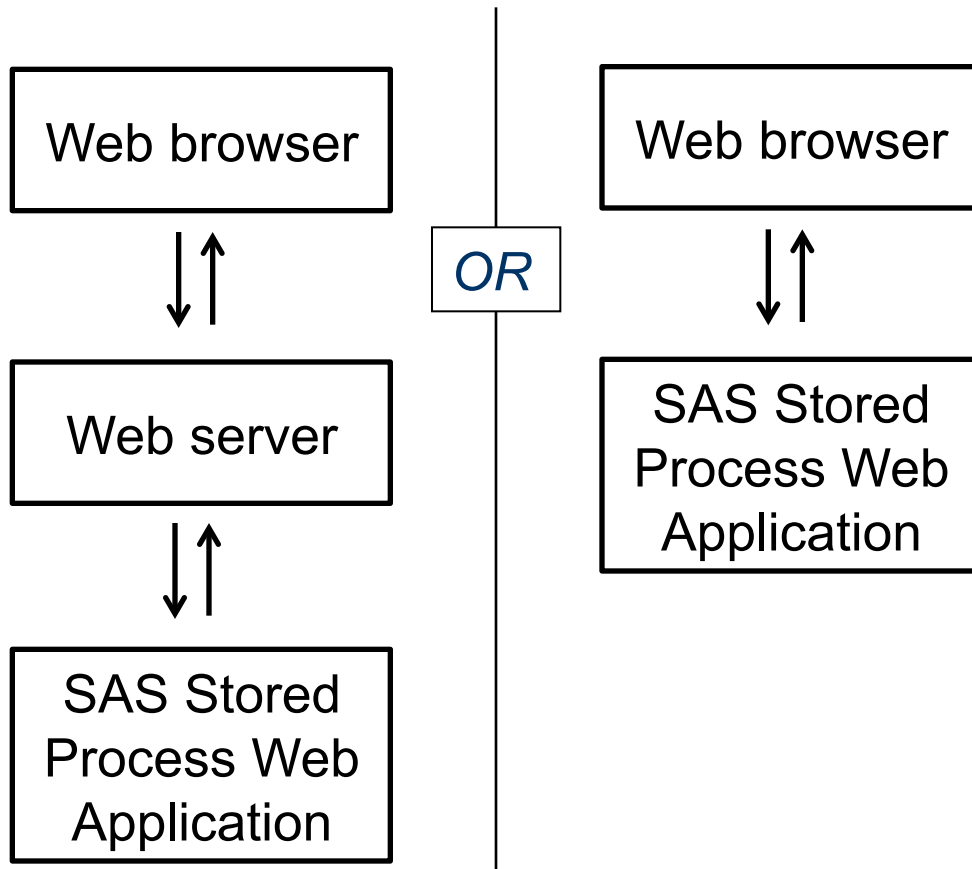


Adding a DATA PROVIDING BACKEND



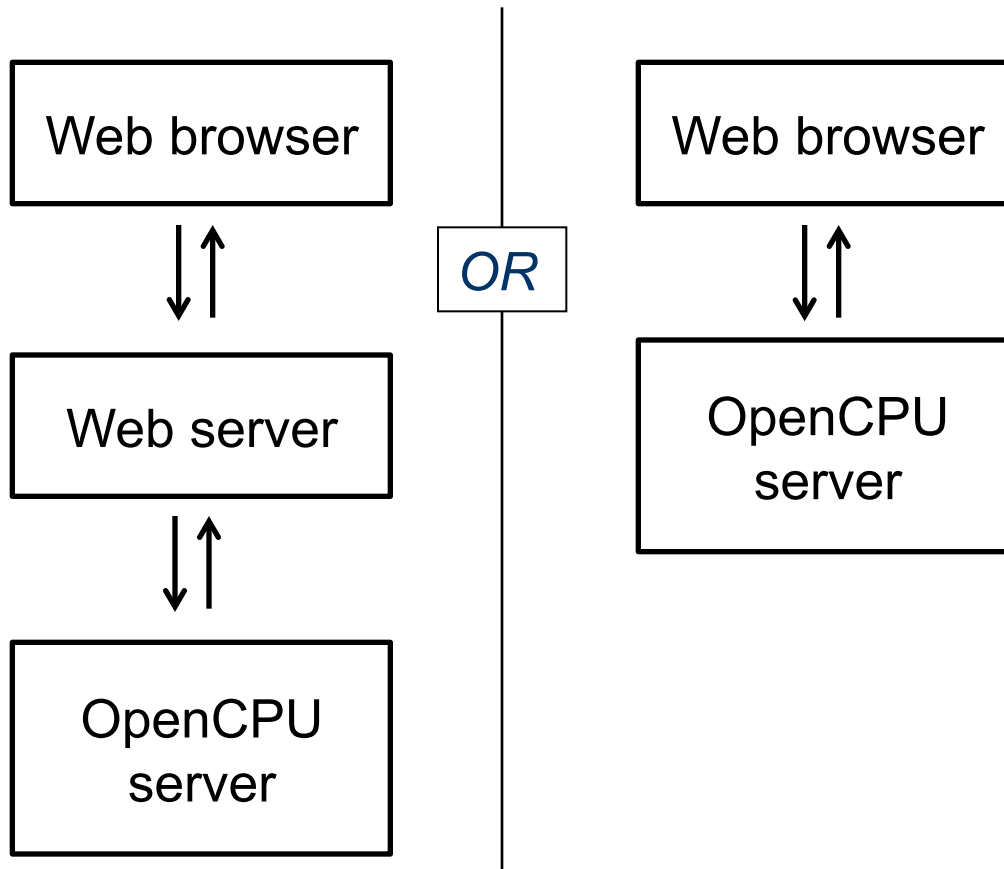
1. The web browser requests the index.html page.
2. The webserver provides the front end Code (HTML, CSS and JavaScript) to the web browser.
3. The delivered Java Script code executed at the web browser fetches data (e.g. by a call to the d3.csv function) using an URL on the web server.
4. The web server acts as a reverse proxy to the OpenCPU server.
5. The OpenCPU server is providing data out of an R installation over HTTP.
6. The reverse proxy feature of the web server passes the data to the web browser.

SAS STORED PROCESS AS DATA PROVIDING BACKEND



- Using the SAS Stored Process Web Application, STPs can be invoked using HTTP GET and STPs can return data e.g. in CSV format that can be processed by JavaScript in the browser.
- As an alternative configuration, it is possible to provide the front end code itself as the result of an STP, without intermediate web server.

HTTP SERVICE INTERFACE TO R AS DATA PROVIDING BACKEND



- **OpenCPU is a system that provides a HTTP API to an R installation, providing ways to call R functions or R scripts and/or to retrieve data over HTTP**
- **As an alternative configuration, it is possible to provide the front end code itself with OpenCPU, without intermediate web server.**

Conclusion

- **For some tasks, open source software solutions can be sufficient to present data in an interactive way.**
- **In this presentation one way was shown to accomplish this with minimal effort.**
- **The presented example can be ...**
 - ... enhanced in several ways
 - ... combined with different data sources
 - ... integrated in existing frontend code.

Thank you for your attention

Dr. Patrick René Warnat

HMS Analytical Software GmbH
Rohrbacher Str. 26
69115 Heidelberg
Germany

www.analytical-software.de/en/