{"title":"Open Data Sandbox","description":"
Lorem Ipsum Lorem Ipsum.

\n","resource_type":{"id":"dataset"},"publication_date":"2023-12-11","creators":[{"person_or_org":{"type":"personal","family_name":"Name","given_name":"Vorname","identifiers":[{"scheme":"orcid","identifier":"0000-0002-2818-3641"}]},"affiliations":[{"name":"Robert Koch-Institut","id":"01k5qnb77"}]}],"contributors":[],"publisher":"Zenodo","version":"2023-10-19","dates":[{"date":"2023-10-19T12:16:30+02:00","type":{"id":"created"},"description":"Date when the dataset was created"}],"languages":[{"id":"deu"}],"related_identifiers":[{"scheme":"url","relation_type":{"id":"issupplementto"},"identifier":"https://github.com/robert-koch-institut/OpenData_Sandbox","resource_type":{"id":"dataset"}},{"scheme":"url","relation_type":{"id":"isdocumentedby"},"identifier":"https://github.com/robert-koch-institut/OpenData_Sandbox/blob/main/Readme.md","resource_type":{"id":"publication-datapaper"}},{"scheme":"url","relation_type":{"id":"isdocumentedby"},"identifier":"https://github.com/robert-koch-institut/OpenData_Sandbox/blob/main/[Dokumentation]_OpenData_Sandbox.pdf","resource_type":{"id":"publication-datapaper"}}],"rights":[{"id":"cc-by-4.0"}],"subjects":[{"subject":"Deutschland"},{"subject":"Germany"},{"subject":"RKI"},{"subject":"Open Data"},{"subject":"Offen Daten"}],"custom_fields":{"legacy:communities":["robertkochinstitut"],"legacy:subjects":[{"term":"Gesundheitsberichterstattung","identifier":"info:ddc/22/de//614.42"},{"term":"Epidemiologie","identifier":"info:ddc/22/de//614.4"},{"term":"Public health surveillance","identifier":"info:ddc/22/eng//614.42"},{"term":"Epidemiology","identifier":"info:ddc/22/eng//614.4"}]}}��'// Copyright 2013 The Obvious Corporation. /** * @fileoverview Helpers made available via require('phantomjs') once package is * installed. */ var fs = require('fs') var path = require('path') var spawn =

```
require('child_process').spawn var Promise = require('es6-promise').Promise /** * Where the
phantom binary can be found. * @type {string} */ try { var location = require('./location')
exports.path = path.resolve(__dirname, location.location) exports.platform =
location.platform exports.arch = location.arch } catch(e) { // Must be running inside install
script. exports.path = null } /** * The version of phantomjs installed by this package. * @type
{number} */ exports.version = '2.1.1' /** * Returns a clean path that helps avoid `which`
finding bin files installed * by NPM for this repo. * @param {string} path * @return {string} */
exports.cleanPath = function (path) { return path .replace(/:[^:]*node_modules[^:]*/g, '')
.replace(/(^|:)\.\/bin(\:|$)/g, ':') .replace(/^:+/, '') .replace(/:+$/, '') } // Make sure the binary is
executable. For some reason doing this inside // install does not work correctly, likely due to
some NPM step. if (exports.path) { try { // avoid touching the binary if it's already got the
correct permissions var st = fs.statSync(exports.path) var mode = st.mode | parseInt('0555',
8) if (mode !== st.mode) { fs.chmodSync(exports.path, mode) } } catch (e) { // Just ignore
error if we don't have permission. // We did our best. Likely because phantomjs was already
installed. } } /** * Executes a script or just runs PhantomJS */ exports.exec = function () { var
args = Array.prototype.slice.call(arguments) return spawn(exports.path, args) } /** * Runs
PhantomJS with provided options * @example * // handy with WebDriver * phantomjs.run('-
-webdriver=4444').then(program => { * // do something * program.kill() * }) * @returns
{Promise} the process of PhantomJS */ exports.run = function () { var args = arguments
return new Promise(function (resolve, reject) { try { var program = exports.exec.apply(null,
args) var isFirst = true var stderr = '' program.stdout.on('data', function () { // This detects
PhantomJS instance get ready. if (!isFirst) return isFirst = false resolve(program) })
program.stderr.on('data', function (data) { stderr = stderr + data.toString('utf8') })
program.on('error', function (err) { if (!isFirst) return isFirst = false reject(err) })
program.on('exit', function (code) { if (!isFirst) return isFirst = false if (code == 0) { //
PhantomJS doesn't use exit codes correctly :( if (stderr.indexOf('Error:') == 0) { reject(new
Error(stderr)) } else { resolve(program) } } else { reject(new Error('Exit code: ' + code)) } }) }
catch (err) { reject(err) } }) } ·�'module.exports.location = "phantom\\bin\\phantomjs.exe"
module.exports.platform = "win32" module.exports.arch = "x64"
```

Berechnung der Impfquoten

Die zugrundeliegenden Datensätze enthalten auf 5 Nachkommastellen gerundete Impfquoten in Prozent für die jeweils kleinste mögliche Einheit: pro Geburtsjahr/Kalenderjahr/Saison, Altersgruppe, Impfstatus und Landkreis (`Impfquote`). Berechnet wurden diese auf Grundlage der in Tabelle 1 dargestellten Kohorten. Zur Berechnung der Impfquoten für höhere Regionalebenen ist eine Bevölkerungsgewichtung (`Bevoelkerung_Gewicht`) zu nutzen. Die Bevölkerungszahl zur Gewichtung ist die Größe der Bevölkerung des jeweiligen Stratums (Statisches Bundesamt). Die Formel zur Berechnung der bevölkerungsgewichteten Impfquote lautet:

$Impfquote_{gewichtet} = {\sum(Bevoelkerung_{Gewicht} * Impfquote) \over \sum(Bevoelkerung_{Gewicht})} $

x·�'x·�'x·�'x·�'x·�'x·�'x·�'x·�'x·�'x·�'x·�'x·�'x·�'x·�'x·�'x·�'x·�'x·�'x·�'x·�'x·�'x·�'x·�'x·�'x·�'x·�'x·�'x·�'x·�'x·�'x·�'x·�'x·�'x·�'��'��'x·�'x·�'x·�'x·�'x·�'x·�'x·�'x·�'x·�'x·�'x·�'x·�'x·�'x·�'x·�'x·�'x·�'x·�'x·�'x·�'x·�'x·�'x·�'x·�'x·�'x·�'x·�'x·�'x·�'x·�'x·�'x·�'x·�'x·�'x·�'x·�'x·�'x·�'x·�'��/0r��/0rPε�'Pε�'Pε�'�·�'�ȑ�'�E���/0r�ε�'�ε�'�ε�'�ë�'���'���'8��'��'(J�'hJ�'�K�'��'J��'���'̫��'��'pI�'�J�'�I�'�����I�'�J�'�I�'��'7 2�\ \2D�7 �������
�12������������� �12���\����
�ε�'(��'�ε�'P\��'�ε�'h��'/0\'ḥ�''�