Hospital Capacity Planning Using Discrete Event Simulation: Introduction

true true true true

2020-11-06

Abstract

Resource planning for hospitals under special consideration of the COVID-19 pandemic.

Introduction

- Resource and capacity planning for hospitals.
- Paper: A novel modelling technique to predict resource-requirements in critical care: a case study (Lawton, McCooe) see (Lawt19a).

Packages

```
## install.packages("devtools")
## devtools::install_github("r-lib/devtools")
```

```
url <- "http://owos.gm.fh-koeln.de:8055/bartz/spot.git"
devtools::install_git(url = url)
```

```
url <- "http://owos.gm.fh-koeln.de:8055/bartz/babsim.hospital.git"
devtools::install_git(url = url)
```

```
rm(list = ls())
suppressPackageStartupMessages({
library("SPOT")
library("babsim.hospital")
library("simmer")
library("simmer.plot")
library("plotly")
})
```

- Package version of `SPOT` must be larger than `2.0.64`:

```
packageVersion("SPOT")
```

Data used by babsim.hospital

- We combine data from two different sources:
    1. `simData`: simulation data, i.e., input data for the simulation. Here, we will use data from the Robert-Koch Institute in Germany.
    2. `fieldData`: real data, i.e., data from the DIVI-Intensivregister. The field data will be used for validating the simulation output.

- Statistically speaking, the `babsim.hospital` simulator models resources usage in hospitals, e.g., number of ICU beds (y), as a function of the number of infected individuals (x).
- In addition to the number of infections, information about age and gender will be used as simulation input.
- We will take a closer look at these data in the following sections.

Simulation Data: RKI Data

Get Data From the RKI Server

- `babsim.hospital` provides a simple function to update the (daily) RKI Data.

```
updateRkidataFile("https://www.arcgis.com/sharing/rest/content/items/f10774f1c63e40168479a1feb6c7ca74/da
```

- The downloaded data will be available as `rkidata`.

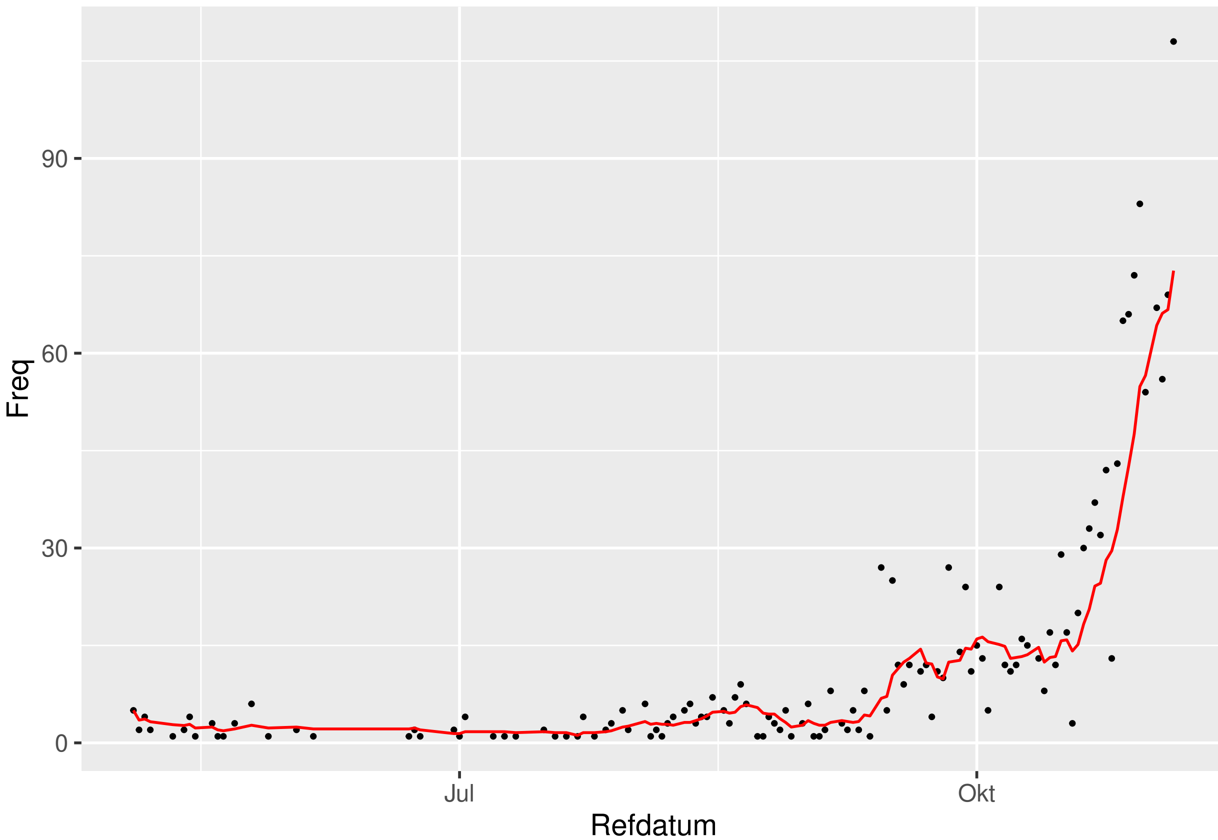
The Original RKI Data

- Data from the Robert-Koch Institute is included in the `babsim.hospital` package as `rkidata`:

```
rkidata <- babsim.hospital::rkidata
str(rkidata)
%> 'data.frame':    394879 obs. of  18 variables:
%>  $ FID                 : int  1876385 1876386 1876387 1876388 1876389 1876390 1876391 1876392 187639
%>  $ IdBundesland        : int  1 1 1 1 1 1 1 1 1 1 ...
%>  $ Bundesland          : chr  "Schleswig-Holstein" "Schleswig-Holstein" "Schleswig-Holstein" "Schles
%>  $ Landkreis           : chr  "SK Flensburg" "SK Flensburg" "SK Flensburg" "SK Flensburg" ...
%>  $ Altersgruppe        : chr  "A00-A04" "A00-A04" "A00-A04" "A00-A04" ...
%>  $ Geschlecht          : chr  "M" "M" "M" "W" ...
%>  $ AnzahlFall          : int  1 1 1 1 1 1 1 1 1 1 ...
%>  $ AnzahlTodesfall     : int  0 0 0 0 0 0 0 0 0 0 ...
%>  $ Refdatum            : chr  "2020/09/30 00:00:00" "2020/10/29 00:00:00" "2020/11/03 00:00:00" "202
%>  $ IdLandkreis         : int  1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 ...
%>  $ Datenstand          : chr  "06.11.2020, 00:00 Uhr" "06.11.2020, 00:00 Uhr" "06.11.2020, 00:00 Uhr
%>  $ NeuerFall           : int  0 0 0 0 0 0 0 0 0 0 ...
%>  $ NeuerTodesfall      : int  -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 ...
%>  $ Meldedatum          : chr  "2020/09/30 00:00:00" "2020/10/29 00:00:00" "2020/11/03 00:00:00" "202
%>  $ NeuGenesen          : int  0 -9 -9 0 0 0 0 0 -9 0 ...
%>  $ AnzahlGenesen       : int  1 0 0 1 1 1 1 1 0 1 ...
%>  $ IstErkrankungsbeginn: int  0 0 0 0 0 1 1 0 0 0 ...
%>  $ Altersgruppe2       : chr  "Nicht übermittelt" "Nicht übermittelt" "Nicht übermittelt" "Nicht übe
```

- Copyright Note (quoted from https://npgeo-corona-npgeo-de.hub.arcgis.com/datasets/dd4580c810204019a7b8eb3e0b329dd6_0): Die Daten sind die „Fallzahlen in Deutschland“ des Robert Koch-Institut (RKI) und stehen unter der Open Data Datenlizenz Deutschland Version 2.0 zur Verfügung.
  Quellenvermerk: Robert Koch-Institut (RKI), dl-de/by-2-0
  Haftungsausschluss: „Die Inhalte, die über die Internetseiten des Robert Koch-Instituts zur Verfügung gestellt werden, dienen ausschließlich der allgemeinen Information der Öffentlichkeit, vorrangig der Fachöffentlichkeit“.
- RKI Data can be visualized as follows (`region = 0` is Germany, `region = 5` North Rhine-Westphalia, `region 5374` Oberbergischer Kreis, etc.):

```
ggVisualizeRki(data=babsim.hospital::rkidata, region = 5374)
```



Preprocessed RKI Data

- Not all the information from the original `rkidata` data set is mandatory for the `babsim.hospital` simulations. We will only use a subset.
- `babsbim.simulator` provides the function `getRkiData()` that extracts the subset of the `rkidata` used in our simulation, optimization, and analysis:

```
rki <- getRkiData(rki = babsim.hospital::rkidata)
str(rki)
%> 'data.frame':    619089 obs. of  7 variables:
%>  $ Altersgruppe: chr  "A35-A59" "A15-A34" "A35-A59" "A15-A34" ...
%>  $ Geschlecht  : chr  "W" "M" "M" "W" ...
%>  $ Day         : Date, format: "2020-01-03" "2020-01-28" ...
%>  $ IdBundesland: int  8 9 9 9 9 9 9 9 9 9 ...
%>  $ IdLandkreis : int  8215 9181 9188 9162 9179 9179 9188 9188 9189 9189 ...
%>  $ time        : num  0 25 25 26 26 28 28 28 28 28 ...
%>  $ Age         : num  47 25 47 25 25 47 47 47 2 47 ...
```

- As illustrated by the output from above, we use the following data:
  1. `Altersgruppe`: age group (intervals, categories), represented as character string
  2. `Geschlecht`: gender
  3. `Day`: day of the infection
  4. `IdBundesland`: federal state
  5. `IdLandkreis`: county
  6. `time`: number of days (`0` = start data). It will be used as `arrivalTimes` for the `simmer` simulations.
  7. `Age`: integer representation of `Altersgruppe`

Field Data (Real ICU Beds)

Get Data From the DIVI Server

- Similar to the `rkidata`, which is available online and can be downloaded from the RKI Server, the field data is also available online.
- It can be downloaded from the DIVI Server as follows, where `YYY-MM-DD` should be replaced by the current date, e.g, `2020-10-26`:

```
updateIcudataFile(
  "https://www.divi.de/joomlatools-files/docman-files/divi-intensivregister-tagesreports-csv/DIVI-Inten
```

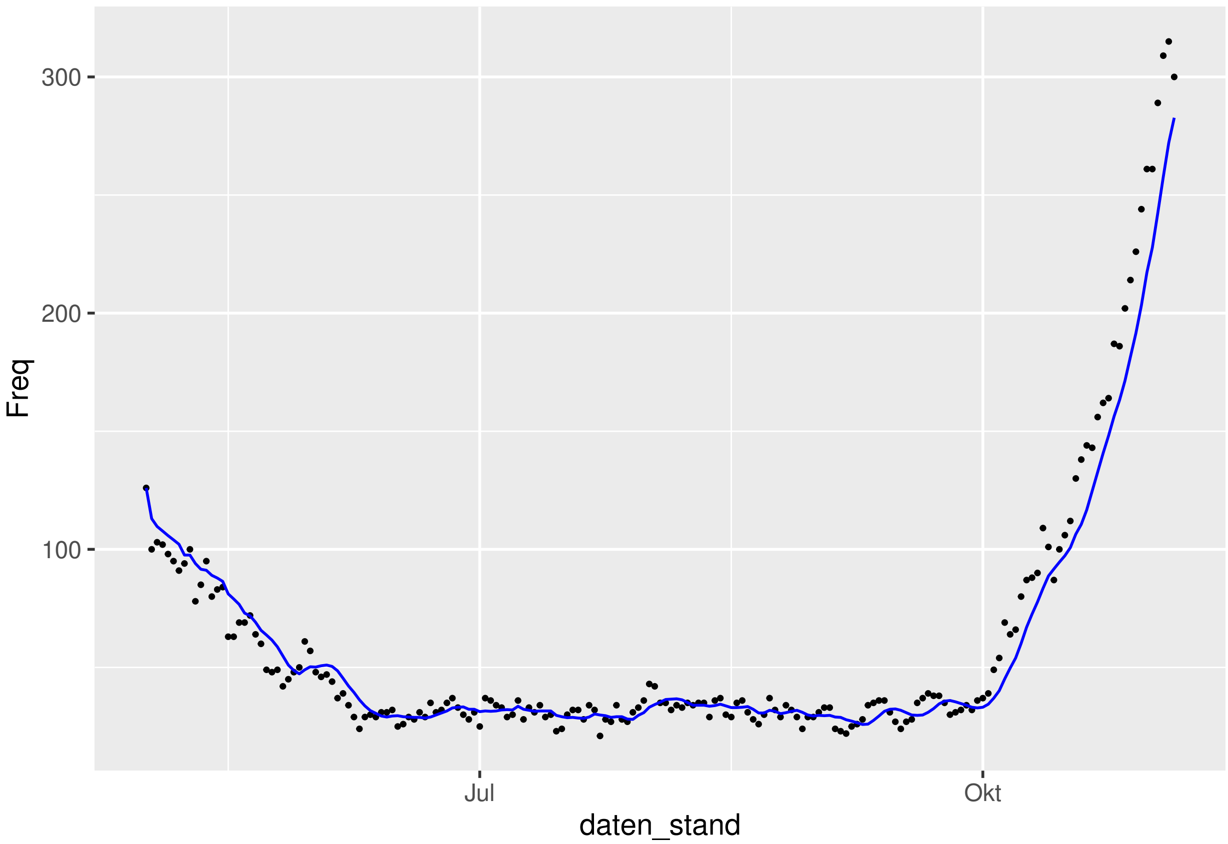
- The downloaded data will be available in `babsim.hospital` as `icudata`.
- Note:
  - The `rkidata` download saves one large `rda` file that contains the whole data set (approx 1 MB).
  - In contrast to the download of the `rkidata`, the download of the `icudata` data saves only data from one day.
- Warning: If `updateIcudataFile()` is executed several times for the same data, there will be duplicates in the resulting file.
- So, please check in advance, if really new data are on the DIVI Server.
- Note: The DIVI data are not open data. The following statement can be found on the DIVI Webpage: " Eine weitere wissenschaftliche Nutzung der Daten ist nur mit Zustimmung der DIVI gestattet."
- Data from the DIVI register is included in the `babsim.hospital` package as `icudata`:

```
icudata <- babsim.hospital::icudata
str(icudata)
%> 'data.frame':    74751 obs. of  9 variables:
%>  $ bundesland                 : int  1 1 1 1 1 1 1 1 1 1 ...
%>  $ gemeindeschluessel         : int  1001 1002 1003 1004 1051 1053 1054 1055 1056 1057 ...
%>  $ anzahl_meldebereiche       : int  2 3 2 1 1 2 1 3 2 1 ...
%>  $ faelle_covid_aktuell       : int  0 3 5 1 3 1 0 0 5 1 ...
%>  $ faelle_covid_aktuell_beatmet: int  0 2 5 1 1 1 0 0 4 0 ...
%>  $ anzahl_standorte           : int  2 3 2 1 1 2 1 3 2 1 ...
%>  $ betten_frei                : int  44 113 115 19 54 7 7 18 10 7 ...
%>  $ betten_belegt              : int  38 110 108 19 26 17 3 34 27 5 ...
%>  $ daten_stand                : Date, format: "2020-05-01" "2020-05-01" ...
```

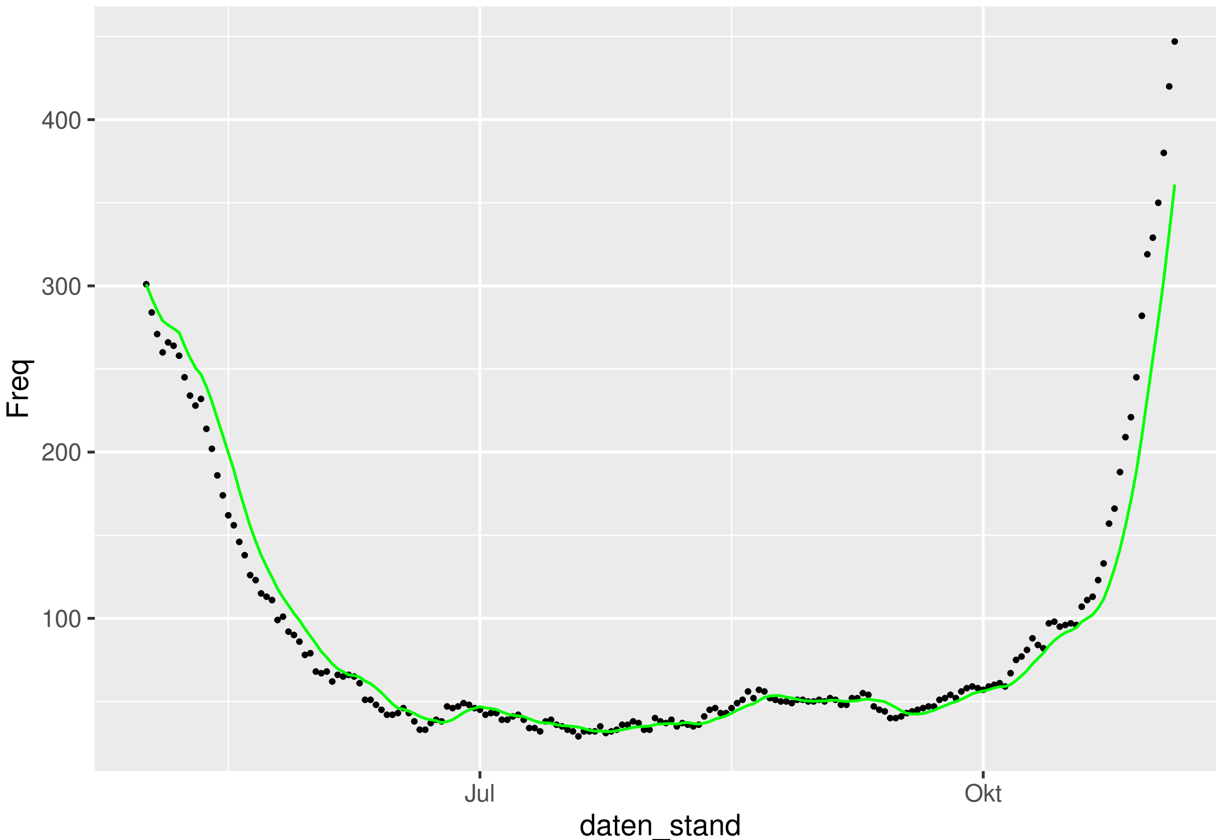
- DIVI Data can be visualized as follows (region = 0 ist Germany, 5 North Rhine-Westphalia, 5374 Oberbergischer Kreis, etc.)

```
p <- ggVisualizeIcu(region = 5)
```

```
print(p[[1]])
```




```
print(p[[2]])
```



Preprocessing DIVI/ICU Data

- Note: ICU bed without ventilation can be calculated as `faelle_covid_aktuell - faelle_covid_aktuell_beatmet`
- The function `getIcuBeds` converts the 9 dim DIVI ICU data `icudata` (bundesland,gemeindeschluessel,. . . , daten_stand) into a data.frame with
  1. bed
  2. intensiveBedVentilation, and
  3. Day

```
fieldData <- getIcuBeds(babsim.hospital::icudata)
str(fieldData)
%> 'data.frame':    189 obs. of  3 variables:
%>  $ intensiveBed           : int  640 597 538 553 591 573 593 527 529 508 ...
%>  $ intensiveBedVentilation: int  1549 1508 1441 1396 1346 1311 1230 1185 1121 1073 ...
%>  $ Day                    : Date, format: "2020-05-01" "2020-05-02" ...
```

- The field data based on DIVI uses two bed categories:
  1. `intensiveBed`: ICU bed without ventilation
  2. `intensiveBedVentilation`: ICU bed with ventilation

Performing Simulations

- To perform a simulation, the setting must be configured (seed, number of repeats, sequential or parallel evaluation, variable names, dates, etc.)
- `region = 5315` represents Cologne.

```
region = 5315
seed = 123
simrepeats = 2
parallel = FALSE
percCores = 0.8
resourceNames =  c("intensiveBed", "intensiveBedVentilation")
resourceEval = c("intensiveBed", "intensiveBedVentilation")
```

- We can specify the field data based on `icudata` (DIVI) for the simulation as follows:

```
FieldStartDate = "2020-09-01"
# Felddaten (Realdaten, ICU):
icudata <- getRegionIcu(data = icudata,
  region = region)
fieldData <- getIcuBeds(icudata)
fieldData <- fieldData[which(fieldData$Day >= as.Date(FieldStartDate)), ]
rownames(fieldData) <- NULL
icu = TRUE
icuWeights = 1
```

- Next, simulation data (RKI data) can be selected. The simulation data in our example, depend on the field data:

```
SimStartDate = "2020-08-01"
rkidata <- getRegionRki(data = rkidata,
  region = region)
simData <- getRkiData(rkidata)
simData <- simData[which(simData$Day >= as.Date(SimStartDate)), ]
## Auch mit fieldData cutten damit es immer das gleiche Datum ist
simData <-
simData[as.Date(simData$Day) <= max(as.Date(fieldData$Day)),]
## time muss bei 1 starten
simData$time <- simData$time - min(simData$time)
rownames(simData) <- NULL
```

- Finally, we combine all data in one data frame `data`:

```
data <- list(simData = simData,
 fieldData = fieldData)
```

- Configuration information is stored in the `conf` list, i.e., `conf` refers to the simulation configuration, e.g., sequential or parallel evaluation, number of cores, resource names, log level, etc.

```
conf <- getConfFromData(simData = data$simData,
fieldData = data$fieldData)
  conf$parallel = parallel
  conf$simRepeats = simrepeats
  conf$ICU = icu
  conf$ResourceNames = resourceNames
  conf$ResourceEval = resourceEval
  conf$percCores = percCores
  conf$logLevel = 1
  conf$w2 = icuWeights
  set.seed(conf$seed)
```

Simulation Model Parameters

- The core of the `babsim.hospital` simulations is based on the `simmer` package.
- It uses simulation parameters, e.g., arrival times, durations, and transition probabilities.
- These are currently 42 parameters (shown below) that are stored in the list `para`.

```
para <- babsimHospitalPara()
str(para)
%> List of 42
%>  $ FactorPatientsInfectedToHospital         : num 0.169
%>  $ AmntDaysInfectedToHospital               : num 8.4
%>  $ FactorPatientsHospitalToNormal           : num 1e-06
%>  $ AmntDaysHospitalToNormal                 : num 1e-06
%>  $ FactorPatientsHospitalToIntensive        : num 0.04
%>  $ AmntDaysHospitalToIntensiv               : num 1e-06
%>  $ FactorPatientsHospitalToVentilation      : num 0.036
%>  $ AmntDaysHospitalToVentilation            : num 1e-06
%>  $ FactorPatientsNormalToHealthy            : num 1e-06
%>  $ AmntDaysNormalToHealthy                  : num 11.6
%>  $ FactorPatientsNormalToIntensive          : num 0.0506
%>  $ AmntDaysNormalToIntensive                : num 1.25
%>  $ FactorPatientsNormalToVentilation        : num 0.101
%>  $ AmntDaysNormalToVentilation              : num 3.63
%>  $ FactorPatientsNormalToDeath              : num 0.139
%>  $ AmntDaysNormalToDeath                    : num 11.4
%>  $ FactorPatientsIntensiveToAftercare       : num 0.25
%>  $ AmntDaysIntensiveToAftercare             : num 7
%>  $ FactorPatientsIntensiveToVentilation     : num 0.25
%>  $ AmntDaysIntensiveToVentilation           : num 2
%>  $ FactorPatientsIntensiveToDeath           : num 0.25
%>  $ AmntDaysIntensiveToDeath                 : num 2
%>  $ FactorPatientsIntensiveToHealthy         : num 0.25
%>  $ AmntDaysIntensiveToHealthy               : num 13
%>  $ FactorPatientsVentilationToAftercare     : num 0.08
%>  $ AmntDaysVentilationToAftercare           : num 9
%>  $ FactorPatientsVentilationToIntensiveAfter: num 0.42
%>  $ AmntDaysVentilationToIntensiveAfter      : num 23
%>  $ FactorPatientsVentilationToDeath         : num 0.5
%>  $ AmntDaysVentilationToDeath               : num 16
%>  $ FactorPatientsAftercareToHealthy         : num 1
%>  $ AmntDaysAftercareToHealthy               : num 21
%>  $ FactorPatientsIntensiveAfterToAftercare  : num 0.5
%>  $ AmntDaysIntensiveAfterToAftercare        : num 7
%>  $ FactorPatientsIntensiveAfterToHealthy    : num 0.5
%>  $ AmntDaysIntensiveAfterToHealthy          : num 18
%>  $ FactorPatientsIntensiveAfterToDeath      : num 1e-05
%>  $ AmntDaysIntensiveAfterToDeath            : num 1
%>  $ GammaShapeParameter                      : num 1
%>  $ RiskFactorA                              : num 0.0205
%>  $ RiskFactorB                              : num 0.01
%>  $ RiskMale                                 : num 2
```

Run simulation

- The `babsim.hospital` simulator requires the specification of
  1. `arrivalTimes`
  2. configuration list `conf`
  3. parameter list `para` for the simulation.
- Arrival times were not discussed yet.
- `babsim.hospital` provides the function `getRkiRisk()` that generates arrivals with associated risks.
- `Risk` is based on age (`Altersgruppe`) and gender (`Geschlect`):

```
rkiWithRisk <- getRkiRisk(data$simData, para)
head(rkiWithRisk)
%>   Altersgruppe Geschlecht        Day IdBundesland IdLandkreis time Age
%> 1      A15-A34          M 2020-08-01            5        5315    0  25
%> 2      A15-A34          M 2020-08-01            5        5315    0  25
%> 3      A15-A34          M 2020-08-01            5        5315    0  25
%> 4      A15-A34          M 2020-08-01            5        5315    0  25
%> 5      A15-A34          M 2020-08-01            5        5315    0  25
%> 6      A15-A34          W 2020-08-01            5        5315    0  25
%>         Risk
%> 1 0.05261803
%> 2 0.05261803
%> 3 0.05261803
%> 4 0.05261803
%> 5 0.05261803
%> 6 0.02630901
```

- To perform simulations, only two parameters are required:
  1. `time`: arrival time
  2. `Risk`: risk (based on age and gender)
- A data.frame with these two parameters is passed to the main simulation function `babsimHospital`.
- Output from the simulation is stored in the variable `envs`.

```
%> [1] "babsimHospital:sequential"
```

Visualize Output

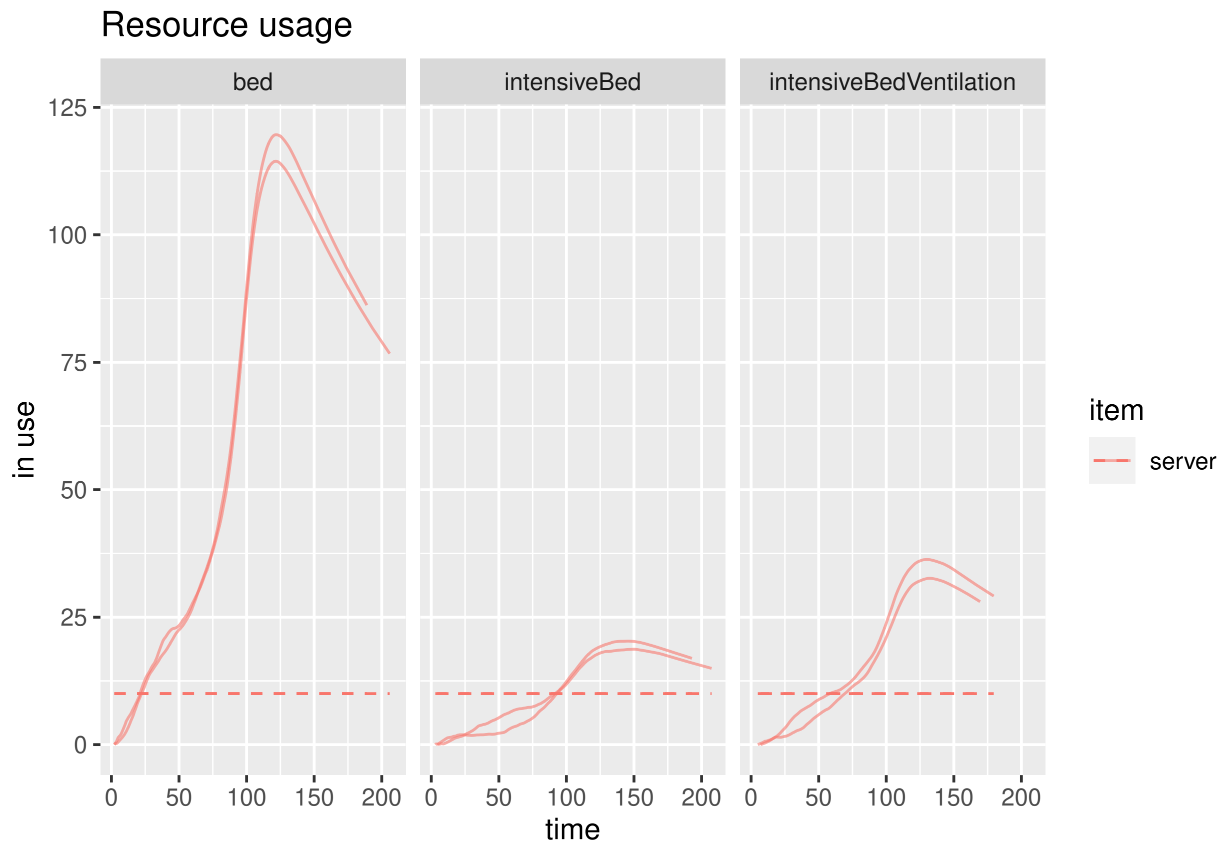
Simmer Plots

- First, we illustrate how to generate plots using the `simmer.plot` package.
- In the following graph, the individual lines are all separate replications. The smoothing performed is a cumulative average.
- Besides `intensiveBed` and `intensiveBedVentilation`, `babsim.hospital` also provides information about the number of non-ICU beds. The non-ICU beds are labeled as `bed`.
- Summarizing, `babsim.hospital` generates output for three bed categories:

1. `bed`
2. `intensiveBed`
3. `intensiveBedVentilation`

- To plot resource usage for three resources side-by-side, we can proceed as follows:

```
resources <- get_mon_resources(envs)
resources$capacity <- resources$capacity/1e5
plot(resources, metric = "usage", c("bed", "intensiveBed", "intensiveBedVentilation"), items = "server")
```

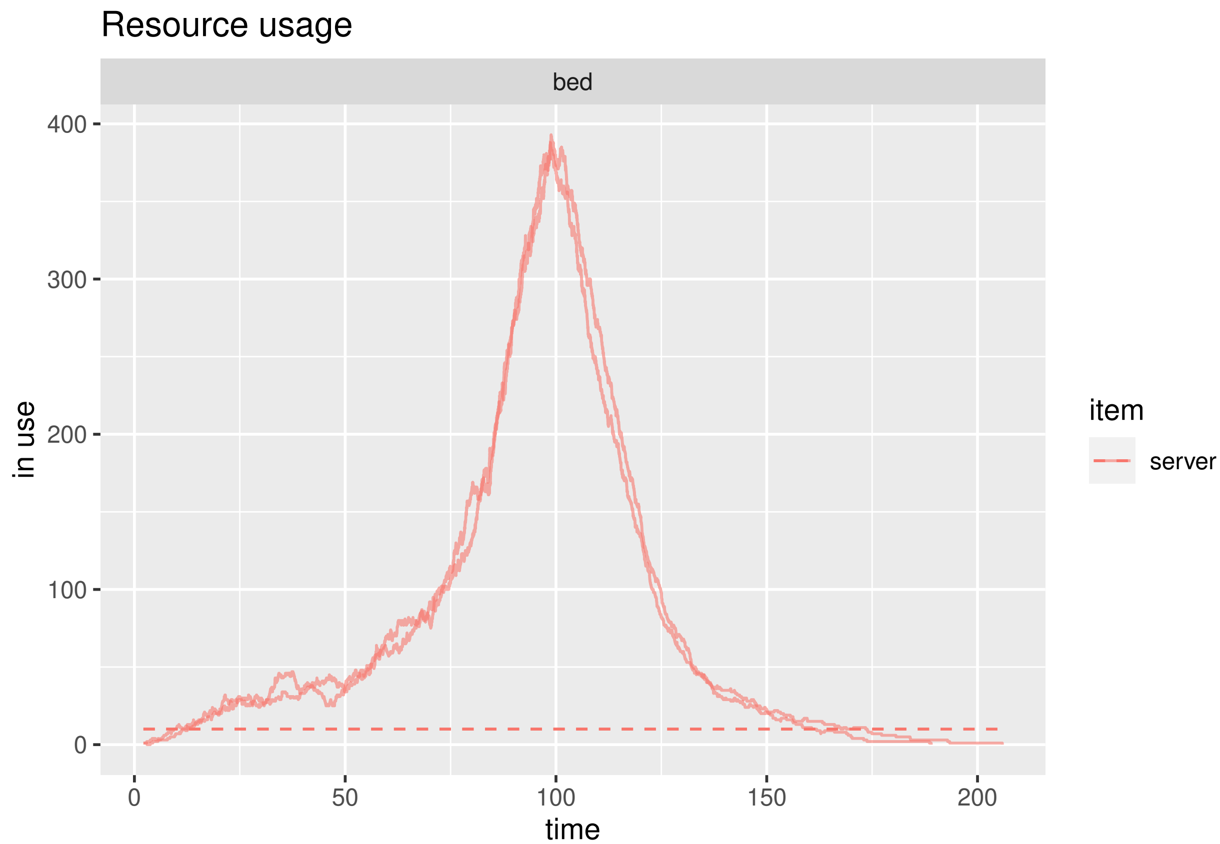


*

Each resource can be plotted separately.

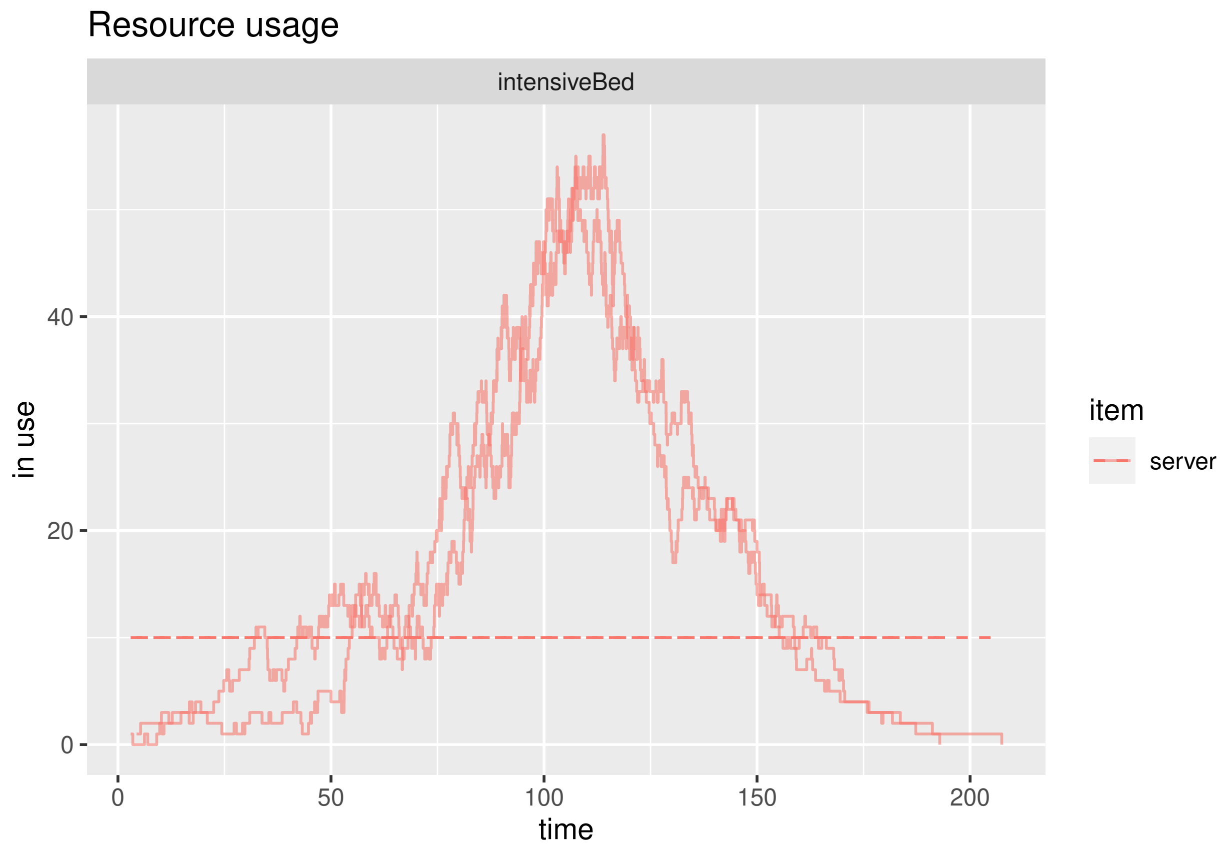
1. The following command generates a plot of non icu beds:

```
plot(resources, metric = "usage", "bed", items = "server", steps = TRUE)
```



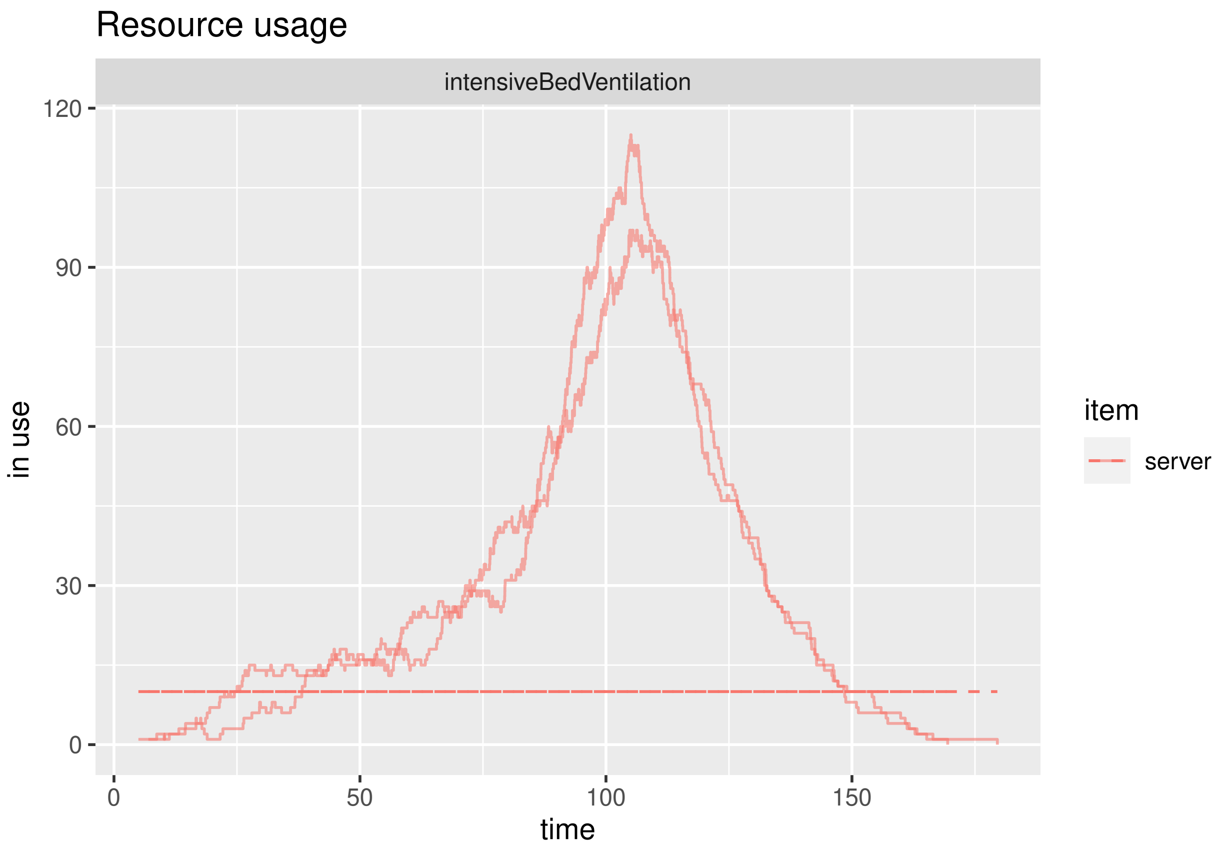
2. The following command generates a plot of icu beds without ventilation:

```
plot(resources, metric = "usage", "intensiveBed", items = "server", steps = TRUE)
```



3. The following command generates a plot of icu beds with ventilation:

```
plot(resources, metric = "usage", "intensiveBedVentilation", items = "server", steps = TRUE)
```



Evaluate Simulation Results

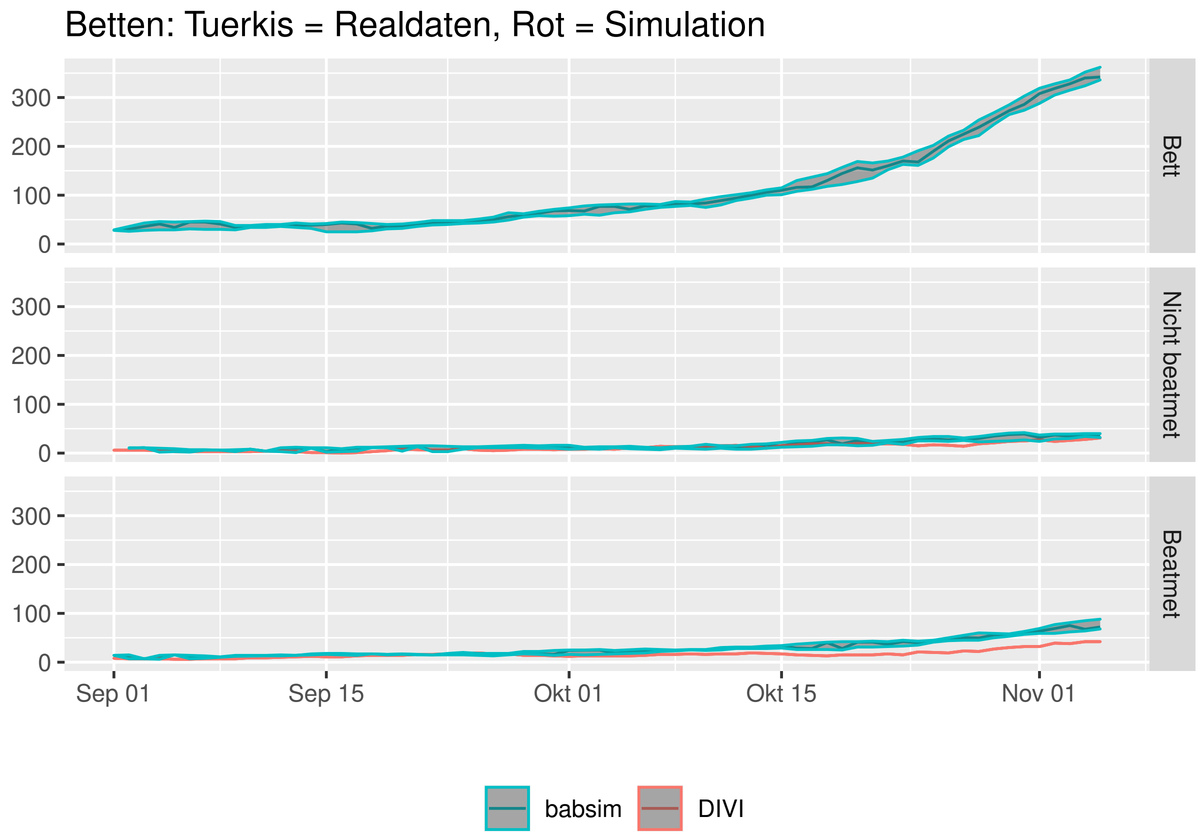
- `babsim.hospital` provides functions for evaluating the quality of the simulation results.
- Simulation results depend on the transition probabilities and durations, i.e., a vector of more than 30 variables.
- These vectors represent *parameter settings*.
- `babsim.hospital` provides a *default* parameter set, that is based on knowledge from domain experts (doctors, members of COVID-19 crises teams, mathematicians, and many more).
- We can calculate the error (RMSE) of the default parameter setting, which was used in this simulation, as follows:

```
fieldEvents <- getRealBeds(data = data$fieldData,
                           resource = conf$ResourceNames)
res <- getDailyMaxResults(envs = envs,  fieldEvents = fieldEvents, conf=conf)
resDefault <- getError(res)
```

The error is 1274.7342104.

- Here, we illustrate how `babsim` plots can be generated.

```
p <- plotDailyMaxResults(res)
plot(p)
```



- Using `ggplot` and `plotly`can be used to generate interactive plots.

```
ggplotly(p)
%> PhantomJS not found. You can install it with webshot::install_phantomjs(). If it is installed, please
```

Optimization

- As discussed above, `babsim.hospital` provides a default parameter set, which can be used for simulations.
- The function `babsimHospitalPara()` provides a convenient way to access the default parameter set:

```
para <- babsimHospitalPara()
```

- `babsim` provides an interface to optimize the parameter values of the simulation model.
- The following code is just a quick demo.

```
### Version: 4.10.26.6
library(babsim.hospital)
library("SPOT")
library("simmer")

runopt(
  expName = "koelnpara20201105PM-",
  rkiwerte = babsim.hospital::rkidata,
  icuwerte = babsim.hospital::icudata,
  region = 5315,
  TrainFieldStartDate = "2020-10-01",
  TrainSimStartDate = "2020-09-01",
  TestFieldStartDate = "2020-10-21",
```

```
  TestSimStartDate = "2020-09-21",
  Overlap = 7,
  seed = 101156,
  repeats = 100,
  funEvals = 250,
  funEvalsFactor = 50,
  size = 100,
  simrepeats = 10,
  subset = 100,
  parallel = TRUE,
  percCores = 0.9,
  icu = TRUE,
  icuWeights = c(1,10)
)
```

- Results from the `runopt()` runs are stored as `*.rda` files.
- `babsim.hospital` provides results from the following regions (towns and counties in Germany):
  - `obkpara`: Oberbergischer Kreis
  - `koelnpara`: City of Cologne
  - `nrwpara`: North-Rhine Westphalia
  - `deutschland`: Germany

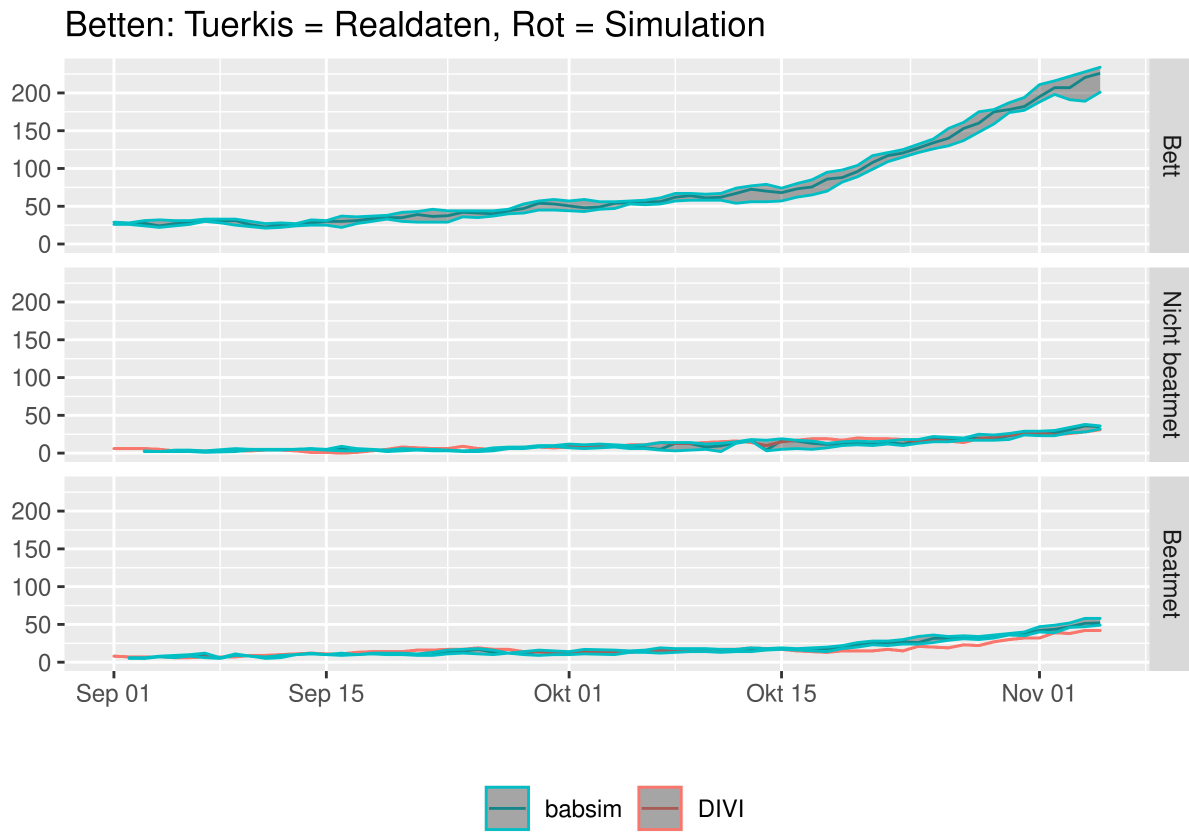
Use Optimized Parameters

- Results (parameter settings) of the short `runopt()` optimization from above can be used as follows:

```
para <- getBestParameter(babsim.hospital::koelnpara202010262)
res <- modelResultHospital(para=para,
                           conf=conf,
                           data = data)
%> [1] "babsimHospital:sequential"
resOpt <- getError(res)
```

- Optimization improves the error from 1274.7342104 to 402.2009295.
- This improvement can also be visualized.

```
p <- plotDailyMaxResults(res)
print(p)
```



- `ggplot` and `plotly`can be used to generate interactive plots.

```
ggplotly(p)
```

Smooth Parameters

- Smooth a parameter set using another parameter set
- Calculate the average of two parameter sets to smooth out any local anomalies.
- Mostly useful to smooth out a local (say OBK) parameter set using a global one (say NRW).
- Technically this function calculates `(1-weight) * para + weight * other` ensuring that the names etc. of `para` are preserved.
- Parameters:
  - para Parameter set to smooth
  - other Other parameters to average in
  - weight Weight of other parameters
- return Weighted parameter set

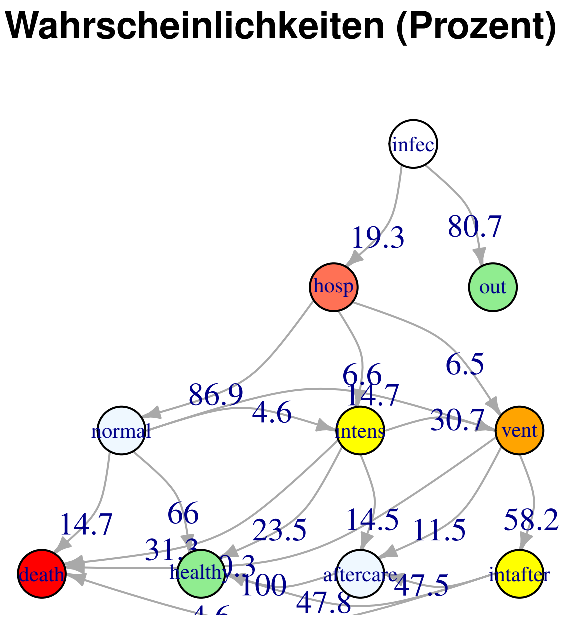
```
para <- smoothParameter(getBestParameter(babsim.hospital::obkpara),
                getBestParameter(babsim.hospital::nrwpara) )
```

Visualize and Analyse Parameter Settings

- `babsim.hospital` includes tools to analyse parameter settings.
- You might recall that parameter settings consist of
  1. transition probabilities, e.g., the probability that an infected individual has to go to the hospital.
  2. durations, e.g., the time span until an infected individual goes to the hospital (in days).
- The following plot illustrates the transition probabilities.
- States are as follows:

1. `infec`: infected
2. `out`: transfer out, no hospital required
3. `hosp`: hospital
4. `normal`: normal station, no ICU
5. `intens`: ICU (without ventilation)
6. `vent`: ICU ventilated
7. `intafter`: intensive aftercare (from ICU with ventilation, on ICU)
8. `aftercare`: aftercare (from ICU, on normal station)
9. `death`: patient dies
10. `healthy`: recovered

```
visualizeGraph(para=para, option = "P")
```



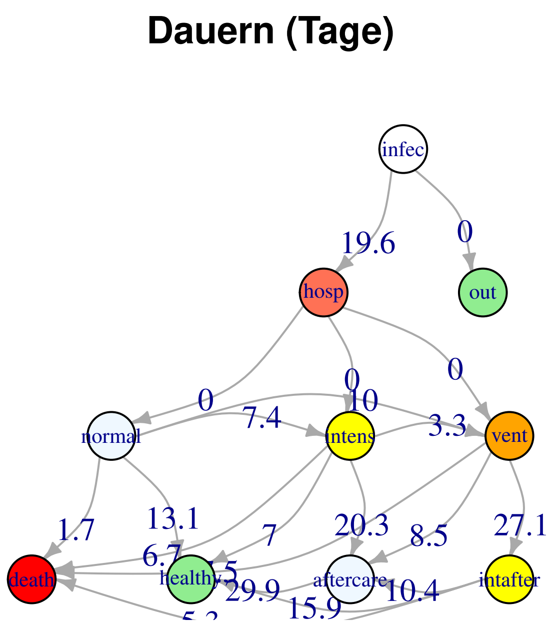
- The transition matrix, that stores the probabilities, is shown below:

```
getMatrixP(para = para )
%>        [,1]      [,2]      [,3]     [,4]       [,5]      [,6]      [,7]
%>   [1,]    0 0.8066371 0.1933629 0.000000 0.00000000 0.0000000 0.0000000
%>   [2,]    0 1.0000000 0.0000000 0.000000 0.00000000 0.0000000 0.0000000
%>   [3,]    0 0.0000000 0.0000000 0.869208 0.06586391 0.0649281 0.0000000
%>   [4,]    0 0.0000000 0.0000000 0.000000 0.04637738 0.1468970 0.0000000
%>   [5,]    0 0.0000000 0.0000000 0.000000 0.00000000 0.3067853 0.0000000
%>   [6,]    0 0.0000000 0.0000000 0.000000 0.00000000 0.0000000 0.5817101
%>   [7,]    0 0.0000000 0.0000000 0.000000 0.00000000 0.0000000 0.0000000
%>   [8,]    0 0.0000000 0.0000000 0.000000 0.00000000 0.0000000 0.0000000
%>   [9,]    0 0.0000000 0.0000000 0.000000 0.00000000 0.0000000 0.0000000
%>  [10,]    0 0.0000000 0.0000000 0.000000 0.00000000 0.0000000 0.0000000
%>             [,8]       [,9]     [,10]
%>   [1,] 0.0000000 0.00000000 0.0000000
%>   [2,] 0.0000000 0.00000000 0.0000000
%>   [3,] 0.0000000 0.00000000 0.0000000
%>   [4,] 0.0000000 0.14674310 0.6599825
%>   [5,] 0.1452354 0.31308642 0.2348929
%>   [6,] 0.1148214 0.30346846 0.0000000
%>   [7,] 0.4753872 0.04634198 0.4782708
```

```
%>  [8,] 0.0000000 0.00000000 1.0000000
%>  [9,] 0.0000000 1.00000000 0.0000000
%> [10,] 0.0000000 0.00000000 1.0000000
```

- Similar to the probabilites, durations can be visualized:

```r
visualizeGraph(para = para, option = "D")
```



- The corresponding matrix is shown below:

```r
getMatrixD(para = para)
%>       [,1] [,2]     [,3] [,4]     [,5]      [,6]     [,7]      [,8]     [,9]
%>  [1,]    0    0 19.63422    0 0.000000  0.000000  0.00000  0.000000 0.000000
%>  [2,]    0    0  0.00000    0 0.000000  0.000000  0.00000  0.000000 0.000000
%>  [3,]    0    0  0.00000    0 0.000000  0.000000  0.00000  0.000000 0.000000
%>  [4,]    0    0  0.00000    0 7.374458 10.041055  0.00000  0.000000 1.736055
%>  [5,]    0    0  0.00000    0 0.000000  3.284602  0.00000 20.303914 6.715912
%>  [6,]    0    0  0.00000    0 0.000000  0.000000 27.14301  8.508941 7.542270
%>  [7,]    0    0  0.00000    0 0.000000  0.000000  0.00000 10.437134 5.273630
%>  [8,]    0    0  0.00000    0 0.000000  0.000000  0.00000  0.000000 0.000000
%>  [9,]    0    0  0.00000    0 0.000000  0.000000  0.00000  0.000000 0.000000
%> [10,]    0    0  0.00000    0 0.000000  0.000000  0.00000  0.000000 0.000000
%>           [,10]
%>  [1,]  0.000000
%>  [2,]  0.000000
%>  [3,]  0.000000
%>  [4,] 13.148139
%>  [5,]  7.040906
%>  [6,]  0.000000
%>  [7,] 15.936337
%>  [8,] 29.910717
%>  [9,]  0.000000
%> [10,]  0.000000
```

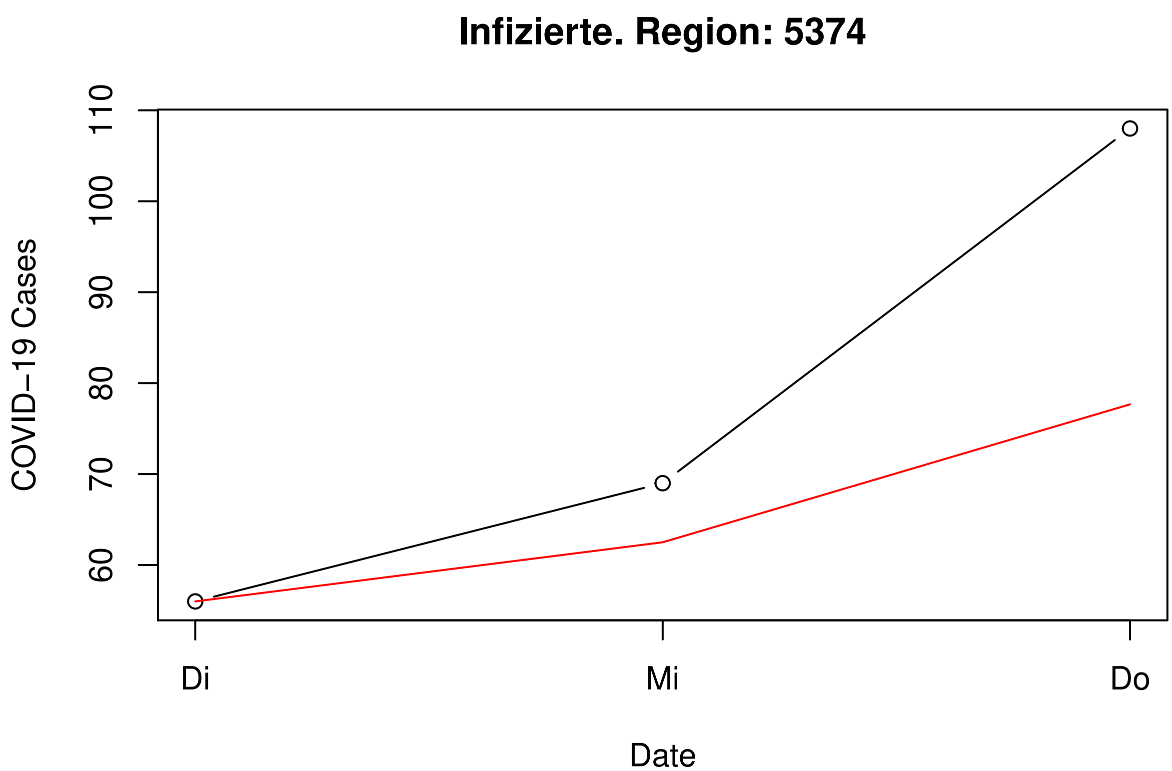
Extend RKI Data

- `babsim.hospital`can be used to simulate scenarios, i.e., possible developments of the pandemic.
- To simulate these scenarios, arrival events must be generated.
- The function `extendRki()` adds new arrival events.
- To generate new arrivals, three parameters must be specified:
    1. `data`: an already existing data set, i.e., the history
    2. `EndDate`: last day of the simulated data (in the future)
    3. `R0`: base reproduction values (R0) at the first day of the scenario and at the last day of the scenario. A linear interpolation between these two values will be used, e.g., if `R0 = c(1,2)` and ten eleven days are specified, the following R0 values will be used: (1.0, 1.1, 1.2, 1.3, . . . , 1.9,2.0).

```
data <- getRkiData(babsim.hospital::rkidata)
n <-  as.integer( max(data$Day)-min(data$Day) )
StartDay <- min(data$Day) + round(n*0.995)
data <- data[which(data$Day >=  StartDay), ]
EndDate <- max(data$Day) + 2
dataExt <- extendRki(data = data,
                     EndDate = EndDate,
                     R0 = c(0.1, 0.2))
```

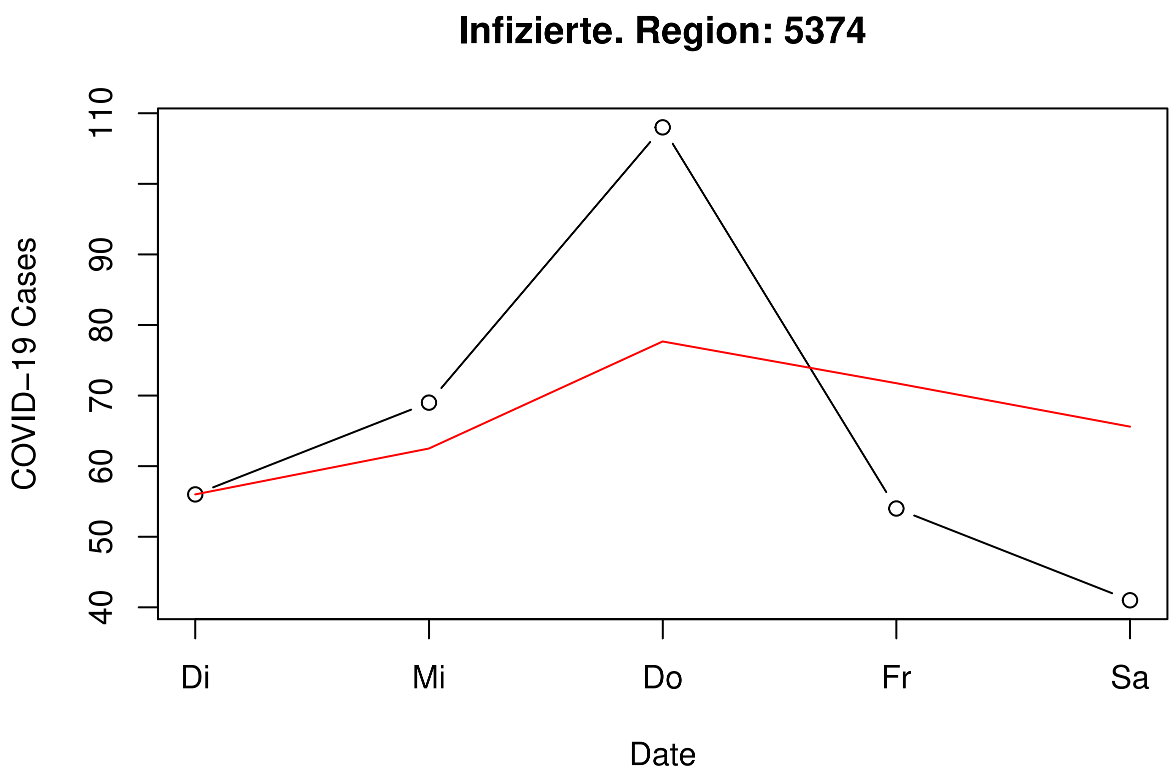
- To illustrate the `extendRki()` data extension procedure, a short example is shown below:

```
visualizeRkiEvents(data = data, region=5374)
```



- The following plot shows the result of the data extension:

```
visualizeRkiEvents(data = dataExt, region = 5374)
```



Sensitivity Analysis

- `babsim.hopital` uses the `R` package SPOT (sequential parameter optimization toolbox) to improve parameter settings.
- `SPOT` implements a set of tools for model-based optimization and tuning of algorithms (surrogate models, optimizers, DOE).
- `SPOT` can be used for sensitivity analysis, which is in important under many aspects, especially:
  - understanding the most important factors (parameters) that influence model behavior. For example, it is of great importance for simulation practitioners and doctors to discover relevant durations and probabilities.
  - detecting interactions between parameters, e.g., do durations influence each other?
- Before visualizations are presented, we show the underlying parameter setting.

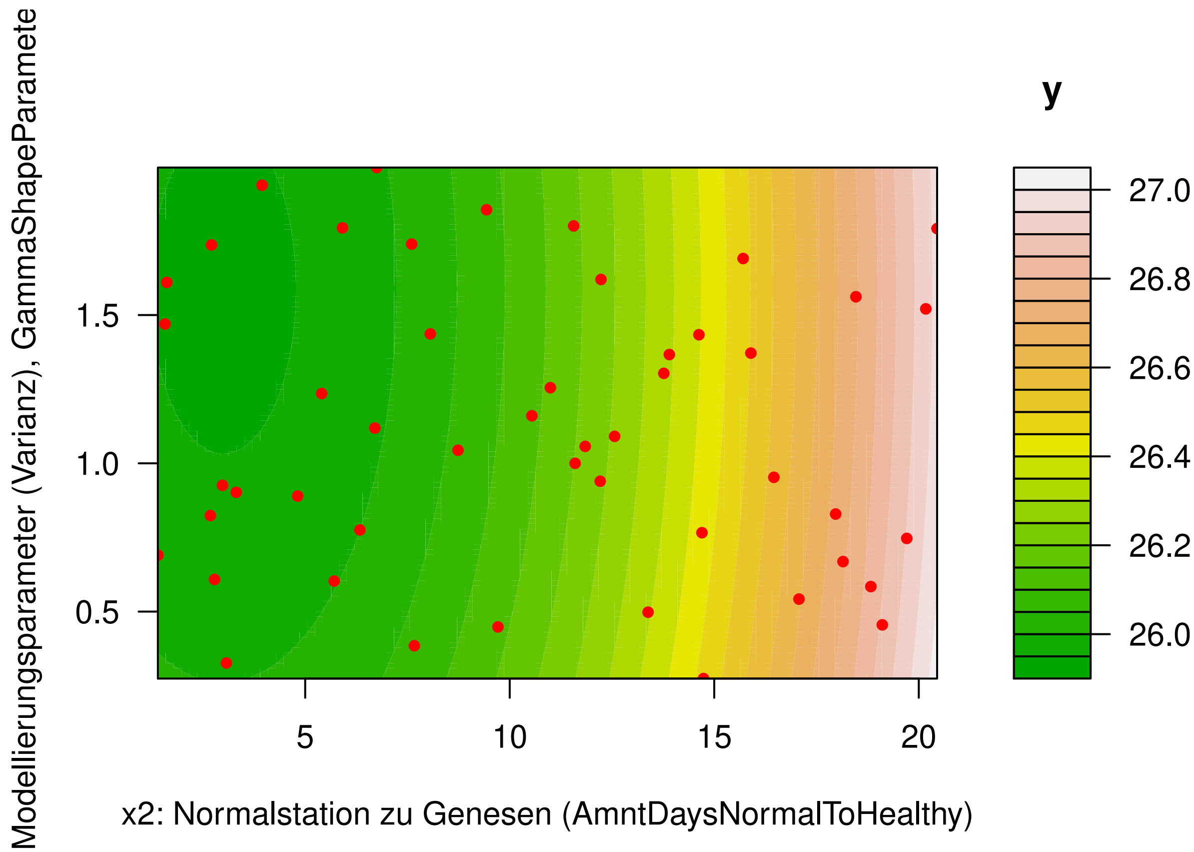
```
res <- res202010262[[2]][[1]]
xBest <- res$xbest
print(xBest)
%>          [,1]     [,2]     [,3]     [,4]     [,5]    [,6]     [,7]     [,8]
%> [1,] 4.082035 8.736332 2.832726 1.111197 20.78704 5.11996 3.531499 10.52963
%>          [,9]    [,10]    [,11]    [,12]    [,13]    [,14]    [,15]    [,16]
%> [1,] 15.33385 17.05922 20.57488 27.67876 6.530299 9.084683 5.720612 1.044091
%>          [,17]      [,18]      [,19]      [,20]      [,21]     [,22]     [,23]
%> [1,] 0.1652369 0.07853186 0.03356371 0.04108942 0.08273478 0.1357528 0.2421358
%>          [,24]    [,25]    [,26]     [,27]     [,28]      [,29]    [,30]
%> [1,] 0.2105247 0.324397 0.506581 0.2699054 0.5109871 0.06013544 20.07326
%>          [,31]      [,32]    [,33]
%> [1,] 0.7280921 0.03720725 1.199678
```

- The corresponding parameter names are:

```r
t(getParameterNameList(1:33))
%>      x1                             x2
%> [1,] "AmntDaysInfectedToHospital" "AmntDaysNormalToHealthy"
%>      x3                           x4
%> [1,] "AmntDaysNormalToIntensive" "AmntDaysNormalToVentilation"
%>      x5                       x6
%> [1,] "AmntDaysNormalToDeath" "AmntDaysIntensiveToAftercare"
%>      x7                               x8
%> [1,] "AmntDaysIntensiveToVentilation" "AmntDaysIntensiveToDeath"
%>      x9                             x10
%> [1,] "AmntDaysIntensiveToHealthy" "AmntDaysVentilationToAftercare"
%>      x11                                    x12
%> [1,] "AmntDaysVentilationToIntensiveAfter" "AmntDaysVentilationToDeath"
%>      x13                                  x14
%> [1,] "AmntDaysIntensiveAfterToAftercare" "AmntDaysIntensiveAfterToHealthy"
%>      x15                              x16
%> [1,] "AmntDaysIntensiveAfterToDeath" "GammaShapeParameter"
%>      x17                                 x18
%> [1,] "FactorPatientsInfectedToHospital" "FactorPatientsHospitalToIntensive"
%>      x19                                    x20
%> [1,] "FactorPatientsHospitalToVentilation" "FactorPatientsNormalToIntensive"
%>      x21                                  x22
%> [1,] "FactorPatientsNormalToVentilation" "FactorPatientsNormalToDeath"
%>      x23                                     x24
%> [1,] "FactorPatientsIntensiveToVentilation" "FactorPatientsIntensiveToDeath"
%>      x25
%> [1,] "FactorPatientsIntensiveToHealthy"
%>      x26
%> [1,] "FactorPatientsVentilationToIntensiveAfter"
%>      x27                                 x28
%> [1,] "FactorPatientsVentilationToDeath" "FactorPatientsIntensiveAfterToHealthy"
%>      x29                                    x30
%> [1,] "FactorPatientsIntensiveAfterToDeath" "AmntDaysAftercareToHealthy"
%>      x31           x32           x33
%> [1,] "RiskFactorA" "RiskFactorB" "RiskMale"
```

- The fitness landscape can be visualized using the function `plotModel`.
- Note, `plotModel` requires two parameter values.
- In the following example, `GammaShapeParameter` (x16) and `AmntDaysNormalToHealthy` (x2) were chosen.
- The plot can be interpreted as follows:
  - The model error is reduced, if patients stay longer on the normal station before they leave the hospital (healthy).
  - The effect of the parameter `GammaShapeParameter` is smaller than the effect of the parameter `AmntDaysNormalToHealthy`.

```r
library(SPOT)
plotModel(res$modelFit, which = c(16,2) ,xlab = c("Modellierungsparameter (Varianz), GammaShapeParamete
```



- A regression-based parameter screening cn be performed to discover relevant (and irrelevant) model parameters:

```
library(SPOT)
fitLm <- buildLM(x=res$x,
                 y=res$y,
                 control = list(useStep=TRUE))
summary(fitLm$fit)
%>
%> Call:
%> lm(formula = y ~ V1 + V2 + V4 + V7 + V11 + V13 + V14 + V17 +
%>     V19 + V21 + V22 + V24 + V27 + V28 + V30 + V32 + V33, data = df)
%>
%> Residuals:
%>     Min      1Q  Median      3Q     Max
%> -46.458 -13.488  -2.839  11.510  42.067
%>
%> Coefficients:
%>              Estimate Std. Error t value Pr(>|t|)
%> (Intercept) -148.5022    48.4632  -3.064 0.004406 **
%> V1             1.5063     0.7995   1.884 0.068685 .
%> V2            -1.6478     0.7641  -2.156 0.038671 *
%> V4            -3.0348     1.0291  -2.949 0.005917 **
%> V7            12.7276     2.3178   5.491 4.75e-06 ***
%> V11            1.9718     0.6003   3.285 0.002478 **
%> V13            3.3057     2.0558   1.608 0.117655
%> V14            2.3737     0.7161   3.315 0.002289 **
```

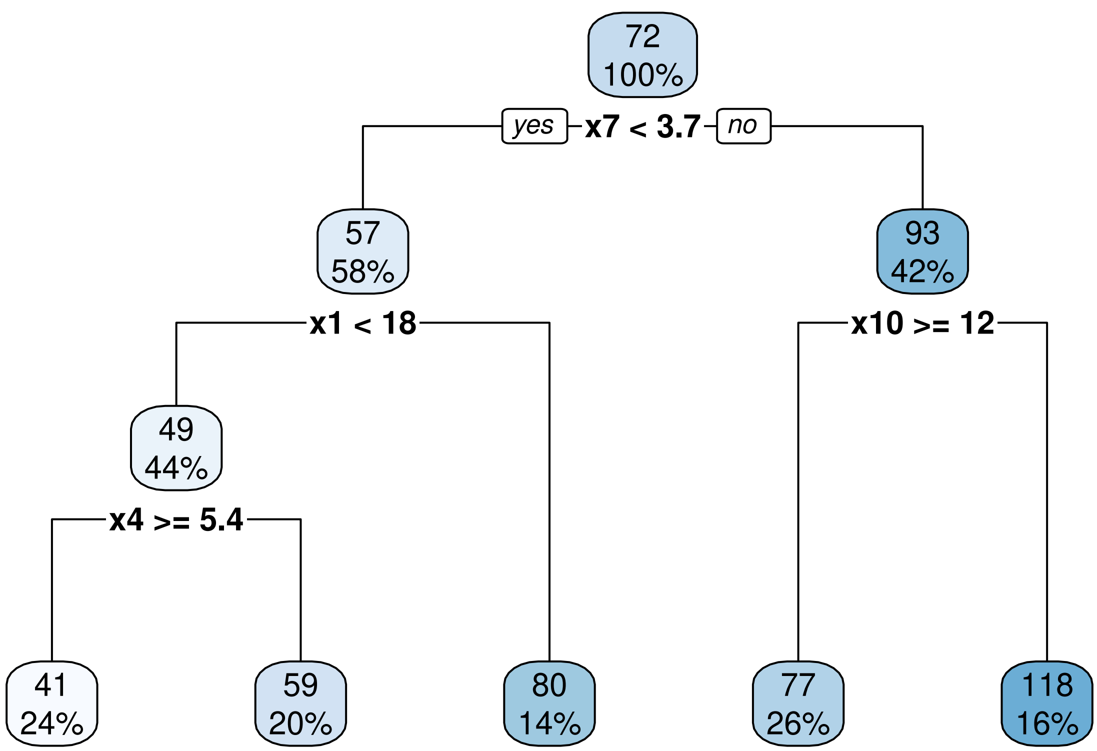
```
%> V17          158.4638    90.8208   1.745 0.090619 .
%> V19          957.6029   262.6256   3.646 0.000935 ***
%> V21          466.3274   195.0063   2.391 0.022837 *
%> V22         -349.1636   119.1696  -2.930 0.006206 **
%> V24         -123.2476    70.1424  -1.757 0.088465 .
%> V27           52.6455    31.2498   1.685 0.101783
%> V28          -40.3978    27.7670  -1.455 0.155442
%> V30            1.1381     0.8606   1.322 0.195389
%> V32          269.1634   222.3064   1.211 0.234848
%> V33           18.1985     6.4617   2.816 0.008251 **
%> ---
%> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
%>
%> Residual standard error: 23.46 on 32 degrees of freedom
%> Multiple R-squared:  0.709,  Adjusted R-squared:  0.5544
%> F-statistic: 4.586 on 17 and 32 DF,  p-value: 0.0001042
```

- Parameter x_7 should be considered important.

```
getParameterName(7)
%> [1] "AmntDaysIntensiveToVentilation"
```

- This finding is supported by a simple regression tree analysis:

```
library("rpart")
library("rpart.plot")
fitTree <- buildTreeModel(x=res$x,
                 y=res$y,
                 control = list(xnames = paste0('x', 1:33)))
rpart.plot(fitTree$fit)
```



Estimate Risiks

- We consider the model $r(x) = exp(bx)$, where x denotes the input variable, e.g., the age, and $b \in R$ is an unknown parameter.
- We use empirical data to estimate b.
- An exponential model with two parameters was chosen to model `risk` as a function of `age` and `gender`: $ r(x) = a exp(bx)$.

```r
age <- c(2,10,25,47,70,90)
    risk <- c(0.01,0.07,0.15,0.65,3,12.64)
    # ab <- getExpCoeff(x=age, y=risk)
    data <- data.frame(x=age, y=risk)
    fit <- nls(y ~ a * exp( b * x),
               data = data,
               start = list(a = 1, b = 0),
               control= nls.control(maxiter = 50, tol = 1e-05, minFactor = 1/1024,
                           printEval = FALSE, warnOnly = FALSE)
    )
    print(coef(fit))
%>          a          b
%> 0.02048948 0.07138200
```

```r
{plot(age,2*risk)
    # female:
    lines(age, 1* predict(fit, list(x = age)))
    # male:
    lines(age, 2* predict(fit, list(x = age) ), col ="red")}
```