Working from self-driving car - parameters for Monte-Carlo Simulation

Georg Hirte

28 01 2022

1. Introduction

We calculate the parameters required in the Monte-Carlo Simulation of the paper "Working from self-driving car'' by Georg Hirte.

- If we have **one observation** far away from zero, we assume this is the mean of a normal distribution with a standard deviation we arbitrarily choose, or a uniform distribution with arbitrarily chosen supports. If the observation is right-skewed and has a left truncation at zero, we use a Gamma distribution. We take 10,000 draws from these distributions.
- If we have **two or more data points** we either use them as limits of a 95% confidence interval of a normal distribution, the mean supports of a truncated distribution, supports and mean of a Gamma distribution, or support of a uniform distribution
- Further restrictions are considered as they come along.

2. Install and load libraries

Clear data

```
rm(list=ls(all=T))
graphics.off()
```

Install packages

```
install.packages("C:/Users/Hirte/Aufsatz/AutonomCarOffice/GAMS_InVehicleWork/gdxrrw_1.0.6.zip"
                 ,repos=NULL, type = "win.binary")
```

```
## Installiere Paket nach 'C:/Users/Hirte/Aufsatz/AutonomCarOffice/R_inVehicleWork/renv/library/R-4.1/x
## (da 'lib' nicht spezifiziert)
```

```
## next: truncated normal distribution and truncated distributions
#install.packages("truncnorm","crch")
# install.packages("UnivRNG")
#source("install.r")
#to read GDX files we need the gdxrrw package

library(reshape2);
library(ggplot2);
library("mcsm");
```

```
## Lade nötiges Paket: MASS
```

```
## Lade nötiges Paket: coda
```

```
library("pastecs");
library(truncnorm);
library(crch);
library(UnivRNG);
library(gdxrrw);
igdx("")
igdx("c:/GAMS/34/");
install.packages('gdxdt')
# next: left truncated gamma
rm(list=ls())
```

Definitions and Functions

```
## Calculate the parameters with cut-offs (interval) and mean used in MC simulation
# we apply a truncated normal distribution

# Definitions
## number of random draws
numberrandom <- 10000
kmmile <- 1.60934
# 23.03.2021
EURDollar <- 0.84
# sqft per sqm
sqmsqft <- 10.7639



# Define plot gamma distribution
#
# ## Calculate the parameters with cut-offs (interval) and mean used in the MC
# if distance between mean and cut-offs is not symmetric we apply a gamma distribution
# if parameters must be non-negative we also apply a gamma distribution
# 1. fit a gamma distribution to parameters (mean and interval)
# 2. make 10000 draws from this distribution

# Fit a gamma distribution knowing that:
# - 20% fall below 15 - only start values here
# - 80% fall below 60 - only start values here
# Inspired by http://www.johndcook.com/blog/2010/01/31/parameters-from-percentiles/
# x <- c(0.2, 0.8)
# y <- c(15, 60)
# # ?dgamma
# # ?qgamma
#
# # plot the gamma curve for a given shape and rate argument ----------------
#
# # draws vertical lines at p
 plotGamma <- function(shape=2, rate=0.5, to=0.99, p=c(0.1, 0.9), cex=1, ...){
   to <- qgamma(p=to, shape=shape, rate=rate)
   curve(dgamma(x, shape, rate), from=0, to=to, n=500, type="l",
         main=sprintf("gamma(x, shape=%1.2f, rate=%1.2f)", shape, rate),
         bty="n", xaxs="i", yaxs="i", col="blue", xlab="", ylab="",
         las=1, lwd=2, cex=cex, cex.axis=cex, cex.main=cex, ...)
```

```
  gx <- qgamma(p=p,  shape=shape, rate=rate)
  gy <- dgamma(x=gx, shape=shape, rate=rate)
  for(i in seq_along(p)) { lines(x=rep(gx[i], 2), y=c(0, gy[i]), col="blue") }
  for(i in seq_along(p)) { text(x=gx[i], 0, p[i], adj=c(1.1, -0.2), cex=cex) }
};

plotGamma2 <- function(shape=2, rate=0.5, to=0.99, p=c(0.1, 0.9), cex=1, ngy=c(1.0), ...){
  to <- qgamma(p=to, shape=shape, rate=rate)
  curve(dgamma(x, shape, rate), from=0, to=to, n=500, type="l",
        main=sprintf("gamma(x, shape=%1.2f, rate=%1.2f)", shape, rate),
        bty="n", xaxs="i", yaxs="i", col="blue", xlab="", ylab="",
        las=1, lwd=2, cex=cex, cex.axis=cex, cex.main=cex, ...)
  gx <- qgamma(p=p,  shape=shape, rate=rate)
  gy <- dgamma(x=gx, shape=shape, rate=rate) * ngy
  for(i in seq_along(p)) { lines(x=rep(gx[i], 2), y=c(0, gy[i]), col="blue") }
  for(i in seq_along(p)) { text(x=gx[i], 0, p[i], adj=c(1.1, -0.2), cex=cex) }
};


# formulate a gamma error function for non-linear least squares -----------
# from http://www.r-bloggers.com/parameters-and-percentiles-the-gamma-distribution/
errorGamma <- function(p=c(10,3), quantiles, exp){
  gx <- c(qgamma(p=quantiles, shape=p[1], rate=p[2]))
  sum((gx-exp)^2)
};
```

3. Travel parameters

3.1 Commuting costs per VKT: gkm

3.1.1 pargkmDE

pargkmDE ≡ avg. private travel cost per km by car () in Germany

Use truncated normal distribution

- ADAC (2020a) compares 73 electric cars ($N = 73$) with hybrid and conventional cars (CV). 15000 VKT per year.
- Results do not consider electricity tax (tax free until 2030. Electricity tax in Germany is 0.0205 EUR/kWh, (StromStG, 2020) → 0.0037 EUR/km given consumption of 18 kWh/100km)
- Assumptions: current prices; lowest 30.6 (CitiGo e IV Ambition von Skoda, lower limit: lower1) and highest 120.8 (Model X Performance von Tesla, upper limit: upper1) are borders of the symmetric 95% confidence intercal CI
- CI is 2×1.96 s.e. wide in case of a normal distributon, $\mu = 1/2 * (0.306 + 1.208)$

$$\rightarrow sd = sqrt(N) * (\text{upperlimit} - \text{lowerlimit})/3.92$$

- 10000 (numberrandom) draws from the truncated normal distribution

```
# lower bound
lower1 <- 0.306
lower1
```

```
## [1] 0.306
```

```r
# upper bound of the truncation
upper1 <- 1.208
upper1
```

```
## [1] 1.208
```

```r
# mean and sd if symmetric
mean1 <- 1/2 * (lower1 + upper1)
mean1
```

```
## [1] 0.757
```

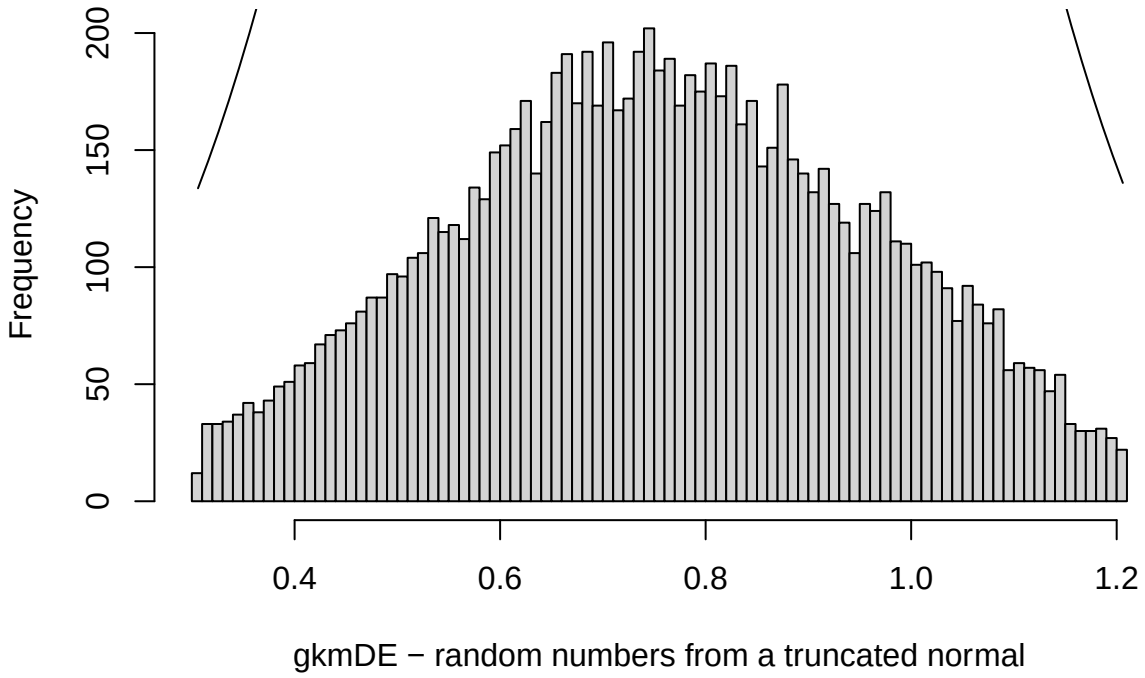
```r
# observations
nn <-
# lower1 is the 2.5% percentile of a normal distribution
#sd1 <- (lower1-mean1)/qnorm(0.025)
#sd1
sd1 <- (upper1-mean1)/qnorm(0.975)
sd1
```

```
## [1] 0.2301063
```

```r
# 2. 10000 random draws from normal. but we have a truncated normal
# do not use rtrunc (there is a hint that this procedure is not fully correct)
# set seed to produce exactly reproducable random numbers
set.seed(1)
#pargkmDE <- rtruncnorm(numberrandom, a = lower1, b=upper1, mean = mean1, sd =sd1)
pargkmDE <- rtruncnorm(numberrandom, a = lower1, b=upper1, mean = mean1, sd =sd1)
meangkmDE <- mean(pargkmDE)
meangkmDE
```

```
## [1] 0.756519
```

```r
# plot distribution
## sequence of 1000 values between lower and upper bounds
xgkm <- seq(from=lower1, to = upper1, by=0.01)


## plot density and histogram
hist(pargkmDE, breaks=80, main="", xlab="gkmDE - random numbers from a truncated normal")
# multiply N with bin width 0.1 of the histogram
ygkm <- dtruncnorm(xgkm, a=lower1, b=upper1, mean=mean1, sd=sd1) * (numberrandom*0.05)
lines(xgkm,ygkm)
```




```
png("imggkmDE.png")
hist(pargkmDE, breaks=80, main="", xlab="gkmDE - random numbers from a truncated normal")
# multiply N with bin width 0.1 of the histogram
ygkm <- dtruncnorm(xgkm, a=lower1, b=upper1, mean=mean1, sd=sd1) * (numberrandom*0.05)
lines(xgkm,ygkm)
dev.off()
```

```
## pdf
##   2
```

3.1.2 pargkmUS

pargkmUS ≡ avg. private travel cost per km by car () in the U.S.

Use truncated normal distribution

- Compostella et al. (2020) provide survey and own calculations. We use their own calculations because they improve on the literature.
- 68 conventional (CV) and autonomous cars (AV) (including ride sharing) ($N = 68$). Small and medium sized cars in 2020 and in 2030.
- Their numbers in VMT. We use them in VKT and EUR ((mile -> 1.60934 km; 1 US-$ = 0.84 EUR at March 19, 2021);
- Assumptions: current prices. data for near-term cars (Fig. 2); lowest 50 US-ct (Private ICEV in 2020 - Small SUV, lower limit) and highest 2.35 US-ct (Ridesource, upper limit) are borders of the symmetric 95% confidence intercal CI

- lower1 is used as 95% quartil, $\mu = 1/2 * (0.50 + 2.35)$

$$\rightarrow sd = \text{lowerlimit})/1.96$$

- 10000 (numberrandom) draws from the truncated normal distribution with support lowerlimit and upperlimit

```
lower1 <- 0.50*EURDollar/kmmile
lower1
```

```
## [1] 0.2609765
```

```
upper1 <- 2.35*EURDollar/kmmile
upper1
```

```
## [1] 1.22659
```

```
mean1 <-1/2 * (lower1 + upper1)
mean1
```

```
## [1] 0.7437832
```

```
#lower1 2.5% percentile of a normal distribution
sd1 <- (lower1-mean1)/qnorm(0.025)
sd1
```

```
## [1] 0.2463344
```

```
# 2. 10000 random draws
## random draws
set.seed(1)
pargkmUS <- rtruncnorm(numberrandom, a = lower1, b=upper1, mean = mean1, sd =sd1)
meandgkmUS <- mean(pargkmUS)
meandgkmUS
```

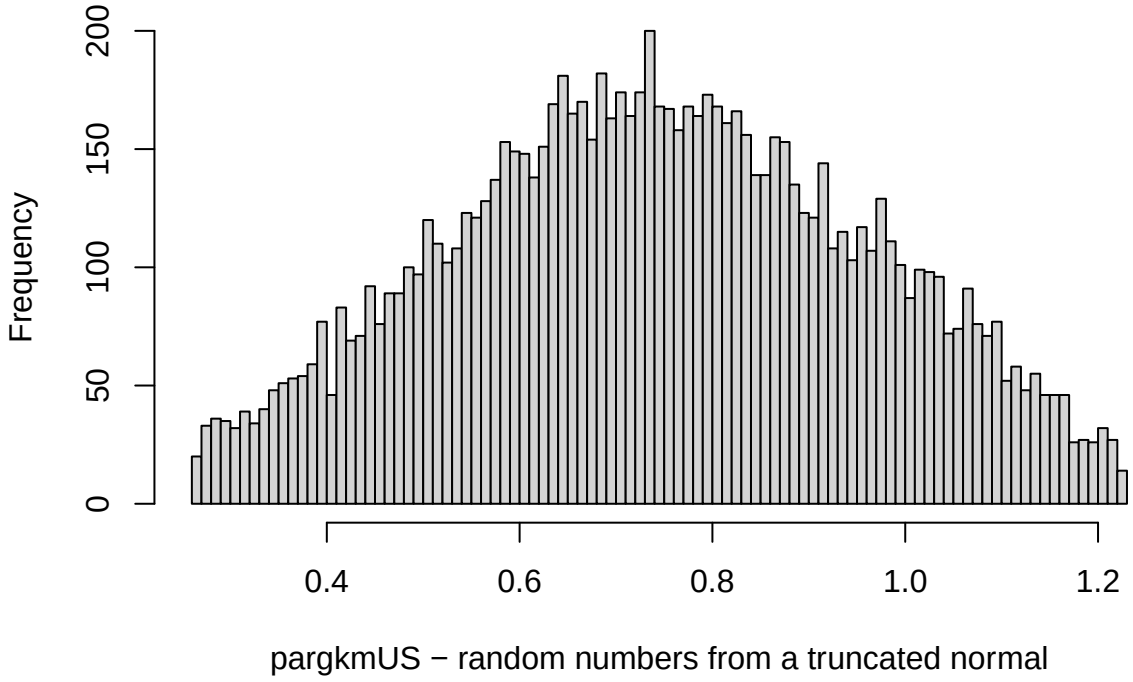
```
## [1] 0.7432682
```

```
# plot distribution
## sequence of 1000 values between lower and upper bounds
xgkm <- seq(from=lower1, to = upper1, by=0.01)
xgkm
```

```
##  [1] 0.2609765 0.2709765 0.2809765 0.2909765 0.3009765 0.3109765 0.3209765
##  [8] 0.3309765 0.3409765 0.3509765 0.3609765 0.3709765 0.3809765 0.3909765
## [15] 0.4009765 0.4109765 0.4209765 0.4309765 0.4409765 0.4509765 0.4609765
## [22] 0.4709765 0.4809765 0.4909765 0.5009765 0.5109765 0.5209765 0.5309765
## [29] 0.5409765 0.5509765 0.5609765 0.5709765 0.5809765 0.5909765 0.6009765
## [36] 0.6109765 0.6209765 0.6309765 0.6409765 0.6509765 0.6609765 0.6709765
## [43] 0.6809765 0.6909765 0.7009765 0.7109765 0.7209765 0.7309765 0.7409765
## [50] 0.7509765 0.7609765 0.7709765 0.7809765 0.7909765 0.8009765 0.8109765
## [57] 0.8209765 0.8309765 0.8409765 0.8509765 0.8609765 0.8709765 0.8809765
## [64] 0.8909765 0.9009765 0.9109765 0.9209765 0.9309765 0.9409765 0.9509765
## [71] 0.9609765 0.9709765 0.9809765 0.9909765 1.0009765 1.0109765 1.0209765
## [78] 1.0309765 1.0409765 1.0509765 1.0609765 1.0709765 1.0809765 1.0909765
## [85] 1.1009765 1.1109765 1.1209765 1.1309765 1.1409765 1.1509765 1.1609765
## [92] 1.1709765 1.1809765 1.1909765 1.2009765 1.2109765 1.2209765
```

```
## vector of values of the heigth of the probability distribution for each value of x
xgkm <- seq(from=lower1, to = upper1, by=0.01)
```

```r
## plot density and histogram
hist(pargkmUS, breaks=80, main="", xlab="pargkmUS - random numbers from a truncated normal")
# multiply N with bin width 0.1 of the histogram
ygkm <- dtruncnorm(xgkm, a=lower1, b=upper1, mean=mean1, sd=sd1)*1000
lines(xgkm,ygkm)
```




```r
png("imggkmUS.png")
hist(pargkmUS, breaks=80, main="", xlab="pargkmUS - random numbers from a truncated normal")
# multiply N with bin width 0.1 of the histogram
ygkm <- dtruncnorm(xgkm, a=lower1, b=upper1, mean=mean1, sd=sd1)*1000
lines(xgkm,ygkm)
dev.off()
```

```
## pdf
##   2
```

3.2 gx (variable monetary costs of VKT during in-vehicle work)

3.2.1 pargsDE

Variable monetary costs of VKT for firms.

Assumptions

- Use only AV that are electric cars
- variable costs = electriciy costs
- energy consumption: 18 kWh/100 km (Deloitte, 2019)

- ADAC (2020b) 5.53 EUR-ct/km (operating costs of 830EUR/anno divided by 15.000 km/anno) (upperlimit) and 3.8 EUR-ct/km (lowerimit) if charged at the firm (the latter at lower commercial electricity costs. Small firms up to 50000 kWh/anno pay 21.19 EUR-ct per kWh). Private households pay on average 27.38 (source?)

```
lower1 <- 0.038
lower1
```

```
## [1] 0.038
```

```
upper1 <- 0.0553
upper1
```

```
## [1] 0.0553
```

```
# 2. 10000 random draws from a uniform distribution
set.seed(12)
pargsDE <- runif(numberrandom, min = lower1, max=upper1)
meangsDE <- mean(pargsDE)
meangsDE
```

```
## [1] 0.04668847
```

3.2.2 pargsUS

Variable monetary costs of VKT for firms in the U.S.

Assumptions

- only fuel costs of ridesource vehicles in 2030-35 (VKT of commercial cars > VKT of private cars)
- Compostella et al. (2020): N=6 (t-distribution; 5 dgf) 0.03 $/VMT (lowerlimit), 0.09 $/VMT (only fuel cost USD/VMT)

```
kmmile <- 1.06934
EURDollar <- 0.84
lower1 <- 0.03*EURDollar/kmmile
lower1
```

```
## [1] 0.02356594
```

```
upper1 <- 0.09*EURDollar/kmmile
upper1
```

```
## [1] 0.07069781
```

```
# We use a uniform distribution of energy/fuel cost per km
# 2. 10000 random draws from uniform distribution with support lower1 and
set.seed(12)
#pargsUS <- rtruncnorm(numberrandom, a = lower1, b=upper1, mean = mean1, sd =sd1)
pargsUS <- runif(numberrandom, min = lower1, max=upper1)
meandgsUS <- mean(pargsUS)
meandgsUS
```

```
## [1] 0.04723668
```

3.3 gd (average fixed monetary costs of travel per hour of invehicle work (EUR/v))

3.3.1 pargdDE

Fixed monetary travel costs per hour of in-vehicle work (calculations: 171.83 EUR/month BMW i3's fixed costs per month from ADAC (2020b) that include insurance, taxes, maintenance costs. Add monthly leasing rate 180 EUR/month. Sum is aggregate fixed costs per month. If the car is used 19 workdays per month á 12 hours per day we get 1.54 EUR/h). Result: 1.54 EUR/h.

We assume a uniform distribution between 1.04 EUR/h and 7 EUR/h due to leasing rates for 30000 km/a including environmental subsidy (Umweltbonus, 5000€). The only information we have are both limits. (Compostella et al., 2020). (Leasing costs are currently between 50 EUR/month and 354 EUR/month.) The car is fully leased by the firm, while Compostella et al. (2020) assumes that the car is rented per km (like a driverless taxi). {To be checked } Following an announcement of VW that leasing will be 7 EUR/h, we use this is upper limit.

```
set.seed(123)
pargdDE <- runif(numberrandom, 1.04, 7)
meangdDE <- mean(pargdDE)
meangdDE
```

```
## [1] 4.005394
```

3.3.2 pargdUS

- Compostella et al. (2020): 80,000 mileage per year, lower limit 0.33 $/VMT, upper limit: 0.37 $/VMT. (Tab A3/A4, used total USD/VMT minus fuel costs for car offered by the employer: ridesource)
- We assume: 53 weeks → 106 free days, plus 4 to consider additional holidays of the firm (Christmas, labor day) = 110. Hence travel days = 365 − 110, usage of AV: 12 hours per day

```
traveldays <- 365-110
hoursday <- 12
mianno <- 80000
costannolow <- mianno*0.33/(traveldays*hoursday)*EURDollar
costannolow
```

```
## [1] 7.247059
```

```
costannoupp <- mianno*0.37/(traveldays*hoursday)*EURDollar
costannoupp
```

```
## [1] 8.12549
```

```
lower1 <- costannolow
lower1
```

```
## [1] 7.247059
```

```
upper1 <- costannoupp
upper1
```

```
## [1] 8.12549
```

```
mean1 <-1/2 * (lower1 + upper1)
mean1
```

```
## [1] 7.686275
```

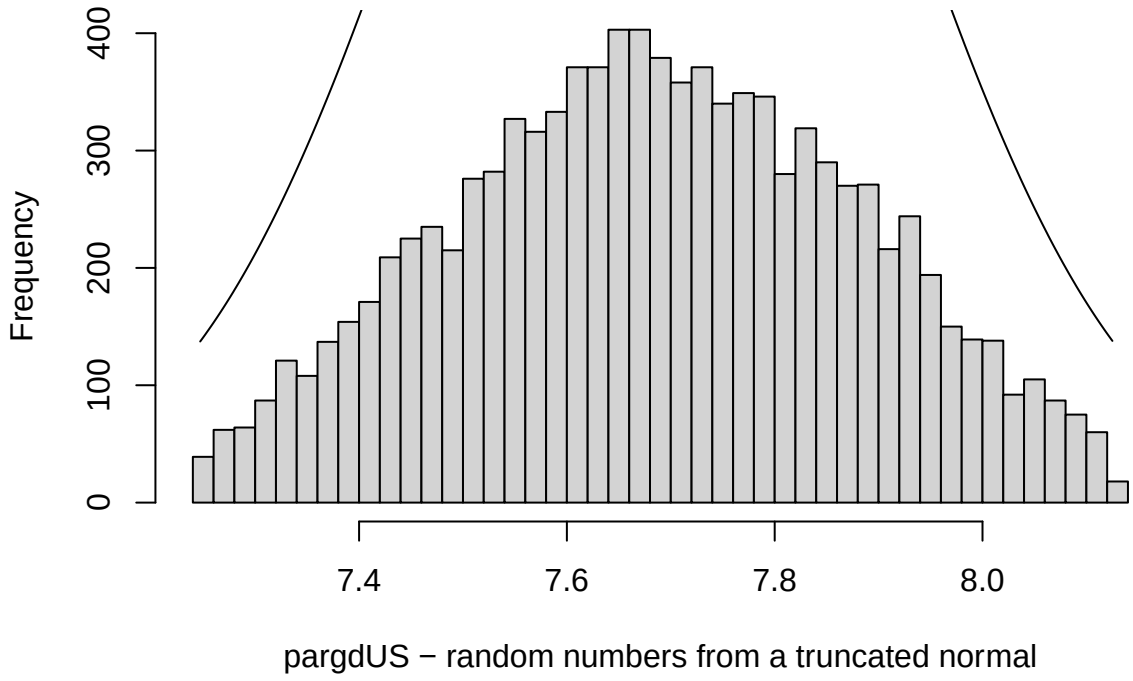
```r
# lower1 defines the 2.5% percentile of a normal distribution
sd1 <- (upper1-mean1)/qnorm(0.975)
sd1
```

```
## [1] 0.2240938
```

```r
# 2. 10000 random draws from truncated normal distribution
set.seed(123)
pargdUS <- rtruncnorm(numberrandom, a = lower1, b=upper1, mean = mean1, sd =sd1)
meangdUS <- mean(pargdUS)
meangdUS
```

```
## [1] 7.687163
```

```r
## sequence of 500 values between lower and upper bounds
xgh <- seq(from=lower1, to=upper1, by=0.001)
hist(pargdUS, breaks=40, main="", xlab="pargdUS - random numbers from a truncated normal")
# multiply N with bin width 0.1 of the histogram
ygh <- dtruncnorm(xgh, a = lower1, b=upper1, mean = mean1, sd =sd1) * (numberrandom*0.05)
lines(xgh,ygh)
```




```r
## histogram and density
png("imggdUS.png")
hist(pargdUS, breaks=40, main="", xlab="pargdUS - random numbers from a truncated normal")
# multiply N with bin width 0.1 of the histogram
ygh <- dtruncnorm(xgh, a = lower1, b=upper1, mean = mean1, sd =sd1) * (numberrandom*0.05)
lines(xgh,ygh)
```

Further

3.4 Two-way Commuting Distance

- distance (two-way commuting distance)

3.4.1 Commuting distance Germany (distanceDE)

First estimate: Dauth and Haller (2018): 21 km (two-way). 9.6% commute between 0-2 km. We approximate the histogram provided by (Dauth and Haller, 2018).

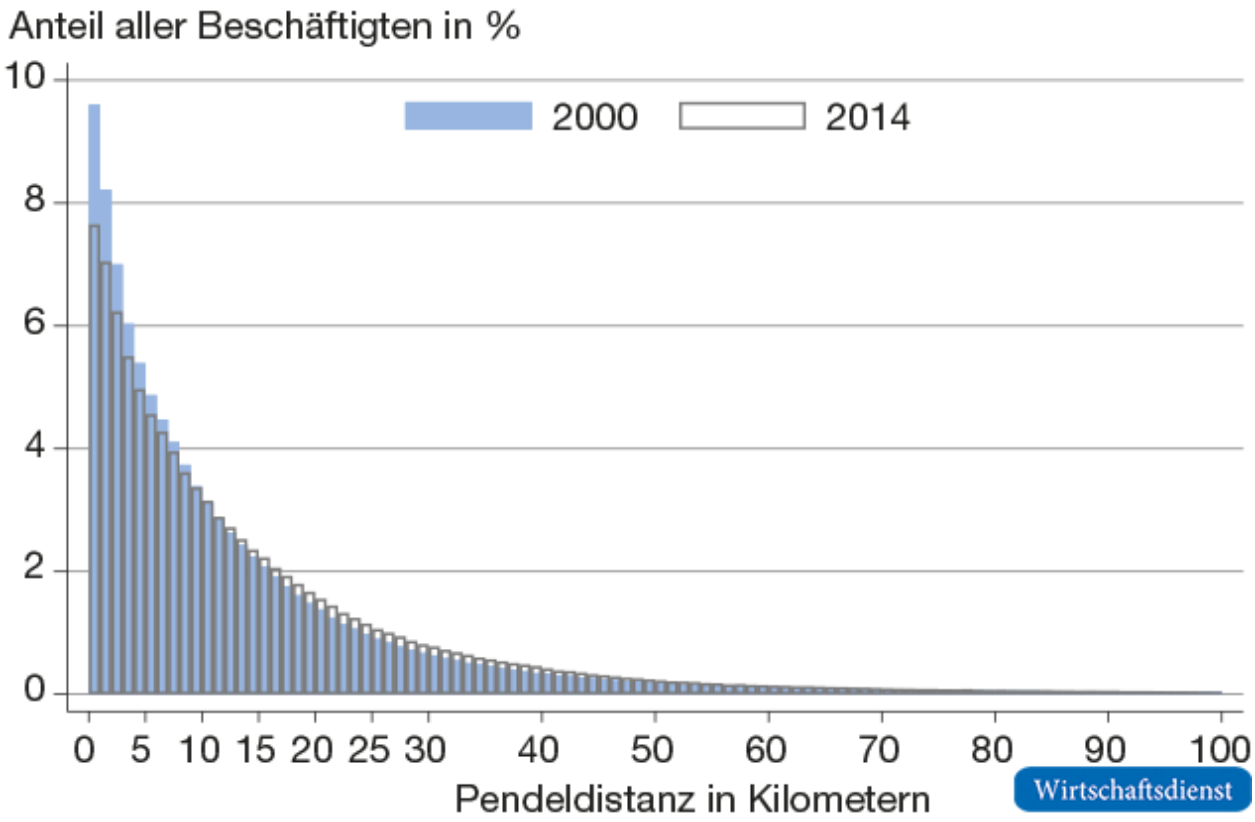
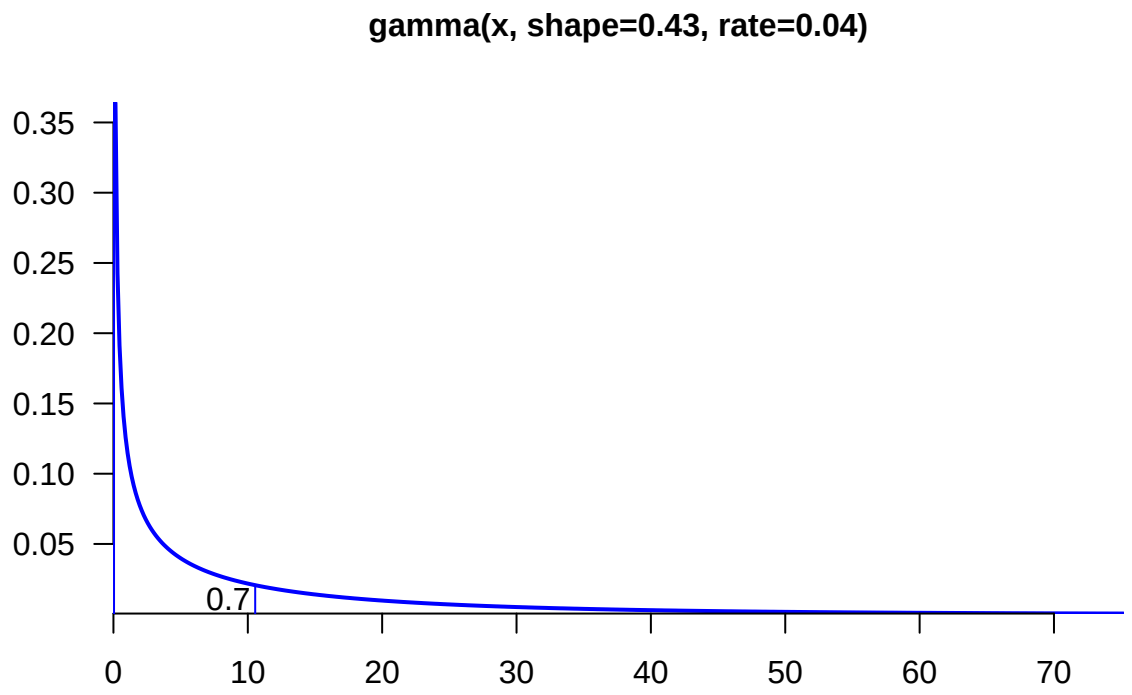


Figure 1: Commuting distance in Germany

```
# calculate gamma distribution for commuting distance (one-way)
distlowerDE <- 2
distmeanDE <- 10.5
distupperDE <- 90
# interval [lower,upper] whith borders at plower% and pupper% quantile and mean
# with non linear maximization
# calculate distribution for plower, mean as 70%-percentil, pupper
p <- c(0.076,0.7,0.995)
# calculate distribution
## p=c(2,0.1) are start values for shape and rate
## draw vertical lines at p
solution <- nlm(f=errorGamma, p=c(2,0.1), quantiles=p,
                exp=c(distlowerDE, distmeanDE, distupperDE),print.level=0);
shape.distDE <- solution$estimate[1]
rate.distDE  <- solution$estimate[2]
```

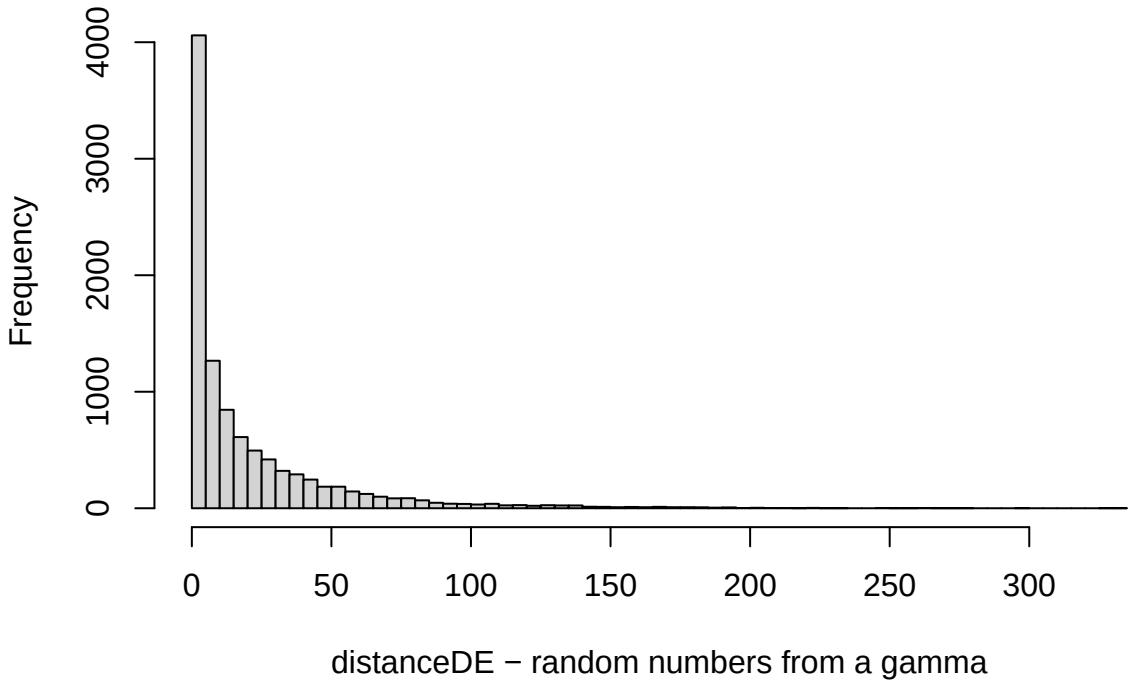
```
#plotGamma(shape.elastMOI, rate.elastMOI[,1], p=p)
plotGamma(shape.distDE, rate.distDE, p=p)
```




```
# use one-way distance Gamma-distribution for random draws of two-way distances
set.seed(1234)
distanceDE <- rgamma(numberrandom,shape.distDE,rate.distDE)
distanceDE <- 2*distanceDE
meanddistDE <- mean(distanceDE)
meanddistDE
```

```
## [1] 21.09938
```

```
# plot histogram
hist(distanceDE, breaks=80, main="", xlab="distanceDE - random numbers from a gamma")
```




```
png("imgdistanceDE.png")
hist(distanceDE, breaks=80, main="", xlab="distanceDE - random numbers from a gamma")
```

3.4.2 Commuting distance US (distanceUS)

Commute distance (miles) 2003	Percent
1-5	29
6-10	22
11-15	17
16-20	10
21-25	7
26-30	5
30-34	3
$\geq$ 35	8
BTS (2003), Quick Facts	
Stretch commute $\geq$ 59 miles	
35-49	5.3
$\geq$ 50	2.7
$\geq$ 200	0.16
BTS (2017)	

Table 1: Commuting distance: 2003

```
# one-way commuting
# 3.3 millions employees from about 121 in the U.S. stretch commute
# (travel more than 50 miles one way) (BTS, 2017). This is one-way distance.
```

```r
sharestretch <- 3.3/121
# From these 2.7% of all commuters share of of travel distances are:
# 22% 50-74 miles, 19% 75-99 miles, 7% 100-124, 6% 125-199, 6% 200+ (BTS, 2017)
share200 <- 0.06*sharestretch
# 6% of those commute more than 200 miles almost each day
# calculate gamma distribution for commuting distance (one-way)
distlowerUS <- 5*kmmile
# mean 25.75 km (16 miles)
distmeanUS <- 15.3*kmmile
distmedianUS <- 10*kmmile
distupperUS <- 200*kmmile
distmeanUS
```

```
## [1] 16.3609
```

```r
distlowerUS
```

```
## [1] 5.3467
```

```r
distupperUS
```

```
## [1] 213.868
```

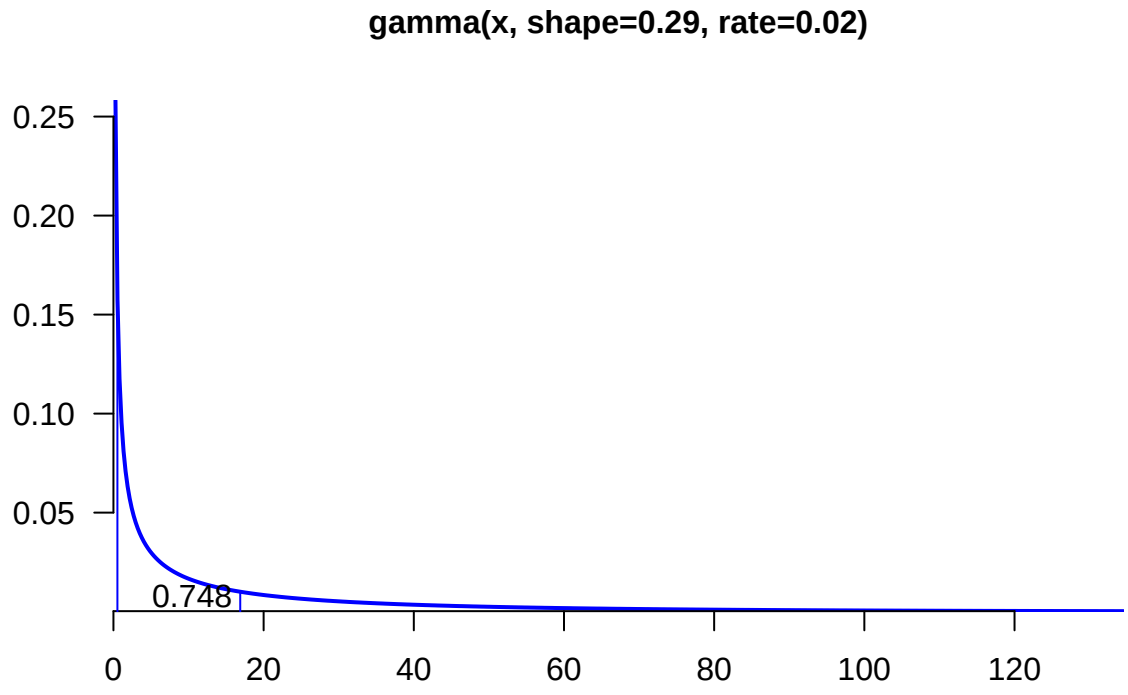
```r
#percent lower and upper (share of lowest and highest distance interval in the data)
plower <- 29/100
plower
```

```
## [1] 0.29
```

```r
pupper <- 1-share200
pupper
```

```
## [1] 0.9983636
```

```r
# interval [lower,upper] whith borders at plower% and pupper% percentile and mean
# with non linear maximization
# calculate distribution for plower, mean as 74,5%-percentil, pupper
p <- c(plower,0.748,pupper)
# calculate distribution for plower, mean as 50%, pupper
#p <- c(0.001,0.5,0.9984)
# calculate distribution
## p=c(2,0.1) are start values for shape and rate
## draw vertical lines at p
solution <- nlm(f=errorGamma, p=c(2,0.1), quantiles=p,
                exp=c(distlowerUS, distmeanUS, distupperUS),print.level=0);
shape.distUS <- solution$estimate[1]
rate.distUS  <- solution$estimate[2]
plotGamma(shape.distUS, rate.distUS, p=p)
```



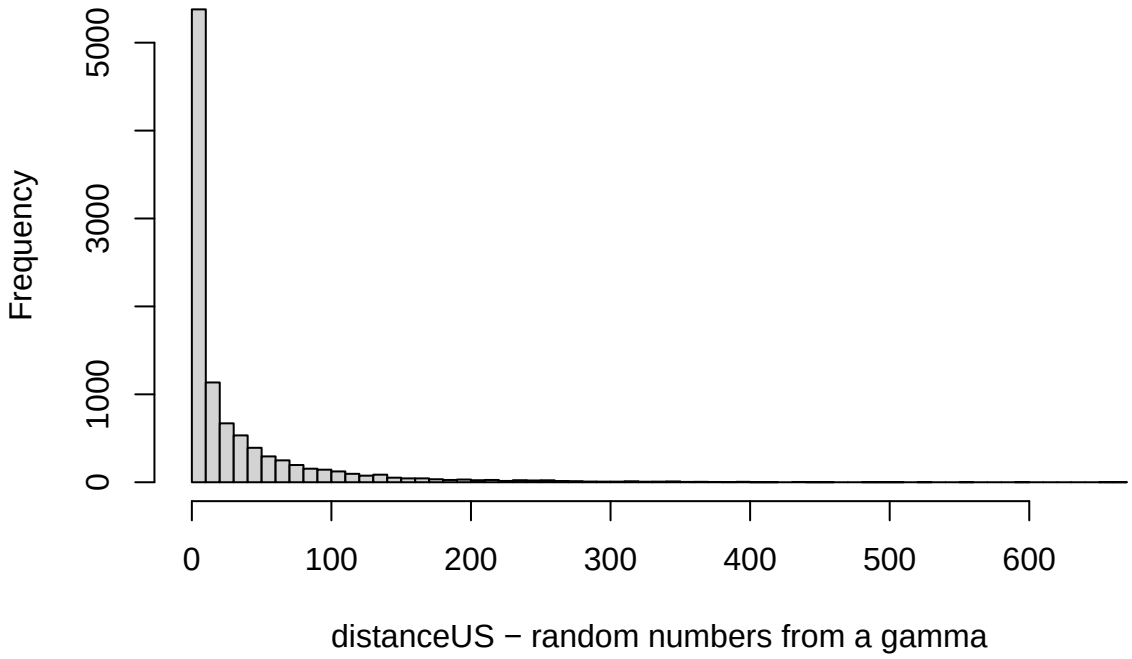

```
# use Gamma-distribution for random draws (two-way, drawn from one-way)
set.seed(1234)
distanceUS2 <- rgamma(numberrandom,shape.distUS,rate.distUS)
# two-way distance
distanceUS <- 2*distanceUS2
meandistUS <- mean(distanceUS)
meandistUS
```

```
## [1] 30.68762
```

```
mediandistUS <- median(distanceUS)
mediandistUS
```

```
## [1] 7.87761
```

```
# mean should be 30.6
## plot histogramn
hist(distanceUS, breaks=80,main="", xlab="distanceUS - random numbers from a gamma")
```




```
png("imgdistanceUS.png")
hist(distanceUS, breaks=80,main="", xlab="distanceUS - random numbers from a gamma")
```

3.5 Two-way Commuting Time

3.5.1 Commuting time Germany (avgTimeDE)

One-Way Commute time min	Percent
$\leq$ 15	19
15-29	31
30-59	20
60-119	5
$\geq$ 120	1
no commut	24

Source: Survey conducted by (Statista, 2021a), N =2099, age: 18-64

Table 2: Daily one-way commuting time DE, 2020

Average commuting time in Germany in 2010 is 44 minutes two way. Gimenéz-Nadal et al. (2020)

```
# two way
avgTimeDE <- 44
# calculate gamma distribution for one-way commuting distance
timelowerDE <- 15
timemeanDE <- 22
timeupperDE <- 120
#percent lower and upper in the table (76% of workforce is commuting)
```

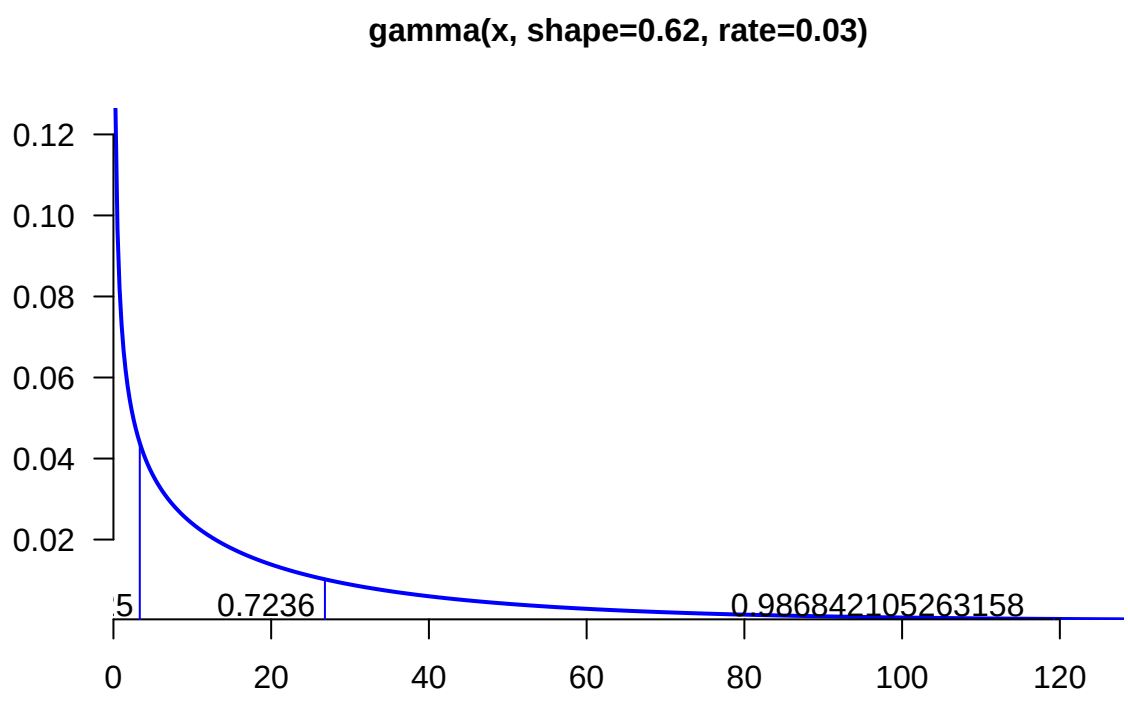
```
plower <- 19/76
plower
```

```
## [1] 0.25
```

```
pupper <- 1-1/76
pupper
```

```
## [1] 0.9868421
```

```
# interval [lower,upper] with borders at plower% and pupper% percentile and mean
# with non linear maximization
# calculate distribtion for plower, mean as 72.2\% percentile to achieve avgtimeDE, pupper
p <- c(plower,0.7236,pupper)
# calculate distribution parameters shape and rate
## p=c(2,0.1) are start values for shape and rate, draw vertical lines at p
solution <- nlm(f=errorGamma, p=c(2,0.1), quantiles=p,
                exp=c(timelowerDE, timemeanDE, timeupperDE),print.level=0);
shape.timeDE <- solution$estimate[1]
rate.timeDE  <- solution$estimate[2]
plotGamma(shape.timeDE, rate.timeDE, p=p)
```

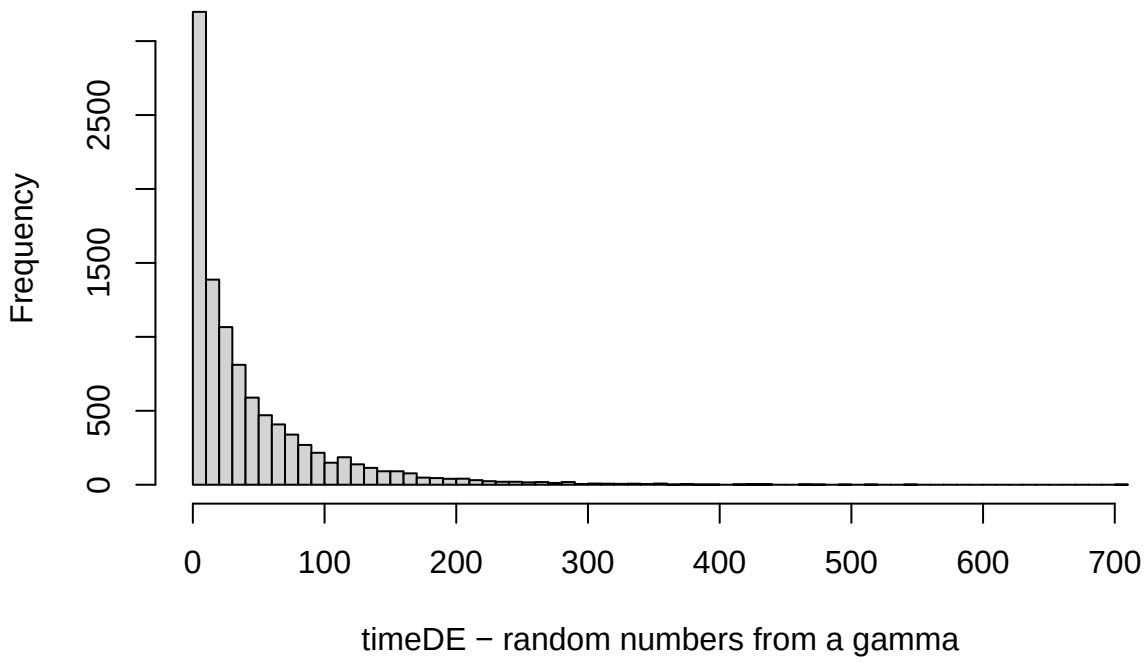



```
# use Gamma-distribution for random draws (two way from distribution for one way)
set.seed(3)
timeDE <- rgamma(numberrandom,shape.timeDE,rate.timeDE);
timeDE <- 2*timeDE
meantimeDE <- mean(timeDE)
```

```
meantimeDE
```

```
## [1] 44.0016
```

```
# plot histogram
hist(timeDE, breaks=80, main="", xlab="timeDE - random numbers from a gamma")
```




```
png("imgtimeDE.png")
hist(timeDE, breaks=80, main="", xlab="timeDE - random numbers from a gamma")
```

3.5.2 Commuting time US (avgTimeUS)

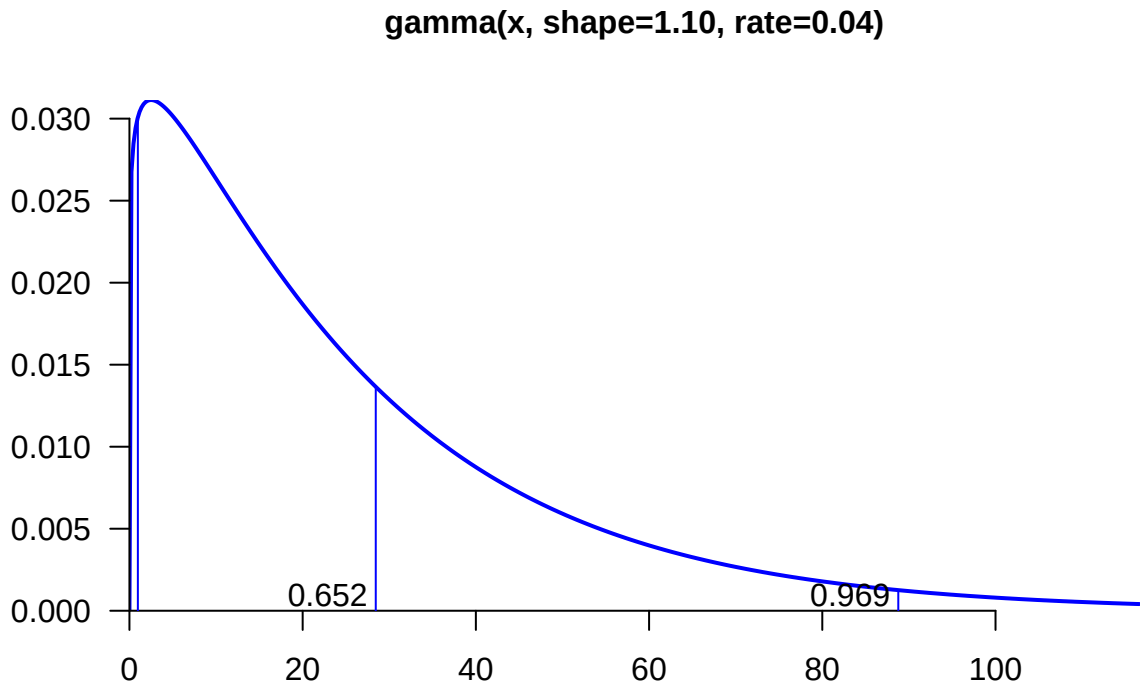
First estimate: Average one-way commuting time is 27.6 minutes in 2020 (54.2 two-way). 9.8% commute at least one hour (one-way!) (US Department Transport, 2020)

```
# two-way
avgTimeUS <- 54.2

# calculate gamma distribution for one-way commuting distance
timelowerUS <- 5
timemeanUS <- 27.6
timeupperUS <- 89
# interval [lower,upper] whith borders at plower% and pupper% percentile and mean
# with non linear maximizati
# use low as plower, mean as 65.2% to achieve avgTimeUS, and pupper
p <- c(0.027,0.652,0.969)
# p=c(2,0.1) are start values for shape and rate
# draw vertical lines at p
```

Travel time (min)	Percent
≤ 5	2.7
5-9	9.2
10-14	12.9
15-19	14.9
20-24	14.1
25-29	6.6
30-34	13.9
35-39	3.2
40-44	4.1
45-59	8.5
60-89	6.7
≥ 90	3.1
Burd et al. (2021), Table 1, p.3	

Table 3: Travel Time To Work: 2019

```
solution <- nlm(f=errorGamma, p=c(2,0.1), quantiles=p,
                exp=c(timelowerUS, timemeanUS, timeupperUS),print.level=0)
shape.timeUS <- solution$estimate[1]
rate.timeUS  <- solution$estimate[2]
plotGamma(shape.timeUS, rate.timeUS, p=p)
```

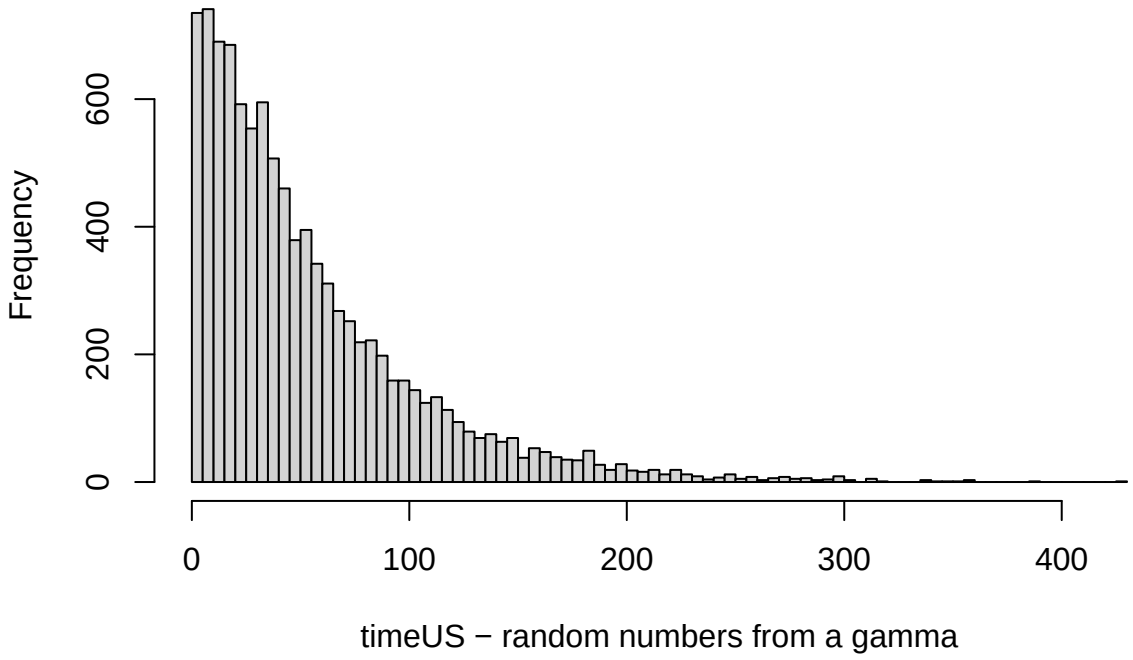



```
# use Gamma-distribution for random draws
set.seed(3)
```

```
timeUS <- 2*rgamma(numberrandom,shape.timeUS,rate.timeUS)
meantimeUS <- mean(timeUS)
meantimeUS
```

```
## [1] 54.19403
```

```
# plot histogram
hist(timeUS, breaks=80, main="", xlab="timeUS - random numbers from a gamma")
```




```
png("imgtimeUS.png")
hist(timeUS, breaks=80, main="", xlab="timeUS - random numbers from a gamma")
```

3.5.3 Average Speed in German Cities (tkmDE)

Average speed in German cities in 2018 (Statista, 2020)

- 11mph: Berlin, Munich, Frankfurt, Leipzig, Bonn
- 12mph: Hamburg, Bremen, Karlsruhe, Augsburg
- 13mph: Köln, Stuttgart, Nürnberg, Hannover, Wiesbaden
- 14mph: Düsseldorf, Wuppertal, Mannheim

SRV for Berlin: average trip with car on a normal day in 2018 is 22.3 min, 7.4 km, 19.8 km/h (Gerike et al., 2020). However, this is average speed. Since commuting is mainly occurring during peak hours, speed is lower. We use the speed provided by (Statista, 2020) which is 17.7 km/h (11 mph).

There is a correlation between commute time and commute distance. It is about 0.78 in the NL in the 1990s (Rietveld et al. 1999). Hence we cannot independently choose time per km and distance. We draw distances from above (MC). Then we calculate for each draw of distance the corresponding average travel time per km by applying the approach of Rietveld et al. (1999).

Average speed is avgtime/avgdistance. Shorter distances takes longer time (speed is 0.6/1 times slower than for distance beyond 25km) (Rietveld et al., 1999).

We use the following information: average speed in large cities, average distance of all commutes (22 km one-way = mean). Distance below are usually traveled at lower speed (Rietveld et al., 1999). According to Rietveld et al. (1999), travel speed increases considerably with distance mainly due to high costs for the first km and the increasing share of highways used at longer commutes.

```
# speed in km/h
speed11 <- kmmile*11
speed12 <- kmmile*12
speed13 <- kmmile*13
speed14 <- kmmile*14
speed18 <- kmmile*18
speed11
```

```
## [1] 11.76274
```

```
speed12
```

```
## [1] 12.83208
```

```
speed18
```

```
## [1] 19.24812
```

```
# for each random draw of distanceDE (tx), draw tkmDE (travel time per km).
# distribution of speed below 25 km depends on speed11--speed14. Use one-way distance
#instead of two-way distance.
# Example: Berlin
# avg. distance of a car trip in Berlin, 2018 = 7.4 km
distcities <- 7.4
#Since distance is 7.4km on average in Berlin, we use the
#first part of the kinked-linear regression line from \citet{Rietveld1999} and calculate
#parameters to meet the city-specific time and speed. This holds for all other
# 11mph-cities.
# Speed within city of Berlin (speed11): Travel time of avg. distanced trip 7.4km
# with average commuting speed in minutes
time11 <- distcities/speed11*60
time11
```

```
## [1] 37.74631
```

```
time18 <- distcities/speed18*60
time18
```

```
## [1] 23.06719
```

```
# set parameters of kinked travel time function time = f(distance):
# For distances below 25km: time = a + b*dist
# We fix a = 8 (about the parameter estimated by Rietveld et al. (1999) for NL).
# Solve for b :  b = (time-a)/dist = (1/speed*60 - a)/dist
# b calculated for one-way trips
parb11 = (time11-8)/distcities
parb11
```

```
## [1] 4.019771
```

```
parb18 = (time18-8)/distcities
parb18
```

```
## [1] 2.036106
```

```
# As a consequence: travel time in Berlin for a trip of length 2 km = 12.6 min,
#7.4 km = 25.1 min, 25 km = 65.7 min


# 11 mph for most congested A-cities, B-cities, C-cities: B, F, M; BN, L; Augsburg
# 14 mph for less congested A-city: D; 7th cong. B-city: Mannheim; 2nd congested
# C-city: Wuppertal
# 18 mph assumed for fastest B- and C-cities.

# Parameter c from Rietveld et al. (1999) = 0.6
# draw from a uniform distribution
# calculate travel time for each distance traveled in minutes
# calculate travel time per km in hours (for distances below 0.1 km we have fixed time of 8 minutes)
set.seed(31)
parambDE <- runif(numberrandom,min=parb18, max=parb11)
timedistDE <- c(1:10000)
timekmDE <- c(10000)
summary(distanceDE)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   1.537   8.650  21.099  27.439 332.863
```

```
for(i in 1:numberrandom) {
  timedistDE[i] <- 8 + parambDE[i] * min(distanceDE[i]/2,25) + 0.6 *
    max(distanceDE[i]/2-25,0)
  if (distanceDE[i]>=0.1) {
    timekmDE[i] <- (timedistDE[i]*60/distanceDE[i])/360
  }
  else
   { timekmDE[i] <- (timedistDE[i]*60)/360
   }
}
timedistDE <- timedistDE*2;
meantimekmDE <- mean(timekmDE)
meantimekmDE
```

```
## [1] 1.039713
```

```
#timekmDE
min(timekmDE)
```

```
## [1] 0.07863005
```

```
max(timekmDE)
```

```
## [1] 13.5141
```

3.5.4 Average Speed in US-Cities (tkmUS)

First estimate: Average speed according to Geotab (2018), average rush-hour speed in major U.S. cities range from 19 mph in Washington D.C. to 41 mph in St. Louis.

```
speed19 <- kmmile*19
speed41 <- kmmile*41
# average travel time US-cities in min:  22.6 (city <- 30000) - 33.1 (>= 5 Mill)
time19 = 33.1
```

```
time41 = 22.6
# calculate avg dist
distcit19 <- speed19 * time19 / 60
distcit41 <- speed41 * time41 / 60
# set parameters of kinked travel time function time = f(distance):
# For distances below 25km: time = a + b*dist
# We fix a = 8 (about the parameter estimated by Rietveld et al. (1999) for NL).
# Solve for b :  b = (time-a)/dist = (1/speed*60 - a)/dist
# b calculated for one-way trips
parb19 = (time19-8)/distcit19
parb19
```

```
## [1] 2.239379
```

```
parb41 = (time41-8)/distcit41
parb41
```

```
## [1] 0.884089
```

```
# Parameter c from Rietveld et al. (1999) = 0.6
# draw from a uniform distribution
# calculate travel time for each distance traveled in minutes
# calculate travel time per km in hours (for distances below 0.1 km we have fixed time of 8 minutes)
set.seed(31)
parambUS <- runif(numberrandom,min=parb41, max=parb19)
timedistUS <- c(1:10000)
timekmUS <- c(10000)
summary(distanceUS)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##   0.0000   0.7004   7.8776  30.6876  35.7674 668.9878
```

```
for(i in 1:numberrandom) {
  timedistUS[i] <- 8 + parambUS[i] * min(distanceUS[i]/2,25) + 0.6 *
    max(distanceUS[i]/2-25,0)
  if (distanceUS[i]>=0.1) {
    timekmUS[i] <- (timedistUS[i]*60/distanceUS[i])/360
  }
  else
   { timekmUS[i] <- (timedistUS[i]*60)/360
   }
}
timedistUS <- timedistUS*2;
meantimekmUS <- mean(timekmUS)
meantimekmUS
```

```
## [1] 1.077083
```

```
#timekmDE
min(timekmUS)
```

```
## [1] 0.05494173
```

```
max(timekmUS)
```

```
## [1] 13.47462
```

```
#set.seed(31)
#timekmUS <- runif(numberrandom,min=1/speed41, max= 1/speed19)
#meantkm <- mean(timekmUS)
#min1 <- 1/speed41
#max2 <- 1/speed19
#meantkm
#min1
#max2
#timekmUS
```

3.6 Avg. hours per workday

3.6.1 Avg. hours per workday Germany

Full-time equivalents * 8 hours day * 33 hours paid leave days (Maye, 2019) * 251 workdays in 2019 (minus weekends) * 251-33 = 218 workdays / 12 month = 18.2 workdays per month

3.6.2 Avg. hours per workday US

- 7.62 h/day (U.S. Bureau of Labor Statistics; BLS)
- 251 workdays in 2019
- US 10 paid leave days and 6 holidays on average Maye (2019)
- 235 workdays per year / 12 = 19.58 workdays per month

3.7 Productivity

Per labor hour (all workers, not only FTEs)

3.7.1 Productivity Germany

- Annual hours worked per worker in 2019: DE 1386 h/year (OECD 2021: (accessed on 22 March 2021)
- Employment DE 2019: 41061 k (OECD 2021: (accessed on 22 March 2021)
- GDP DE 2019: 4644164 million $ (accessed on 22 March 2021)
- Avg. prod per hour worked (own calculation)
- GDP per hour worked: own calculations:

```
# GDP in 2019 in $/year
GDPDE <- 4644164000000*EURDollar
# Avg. hours worked per worker and year (h/year) in 2019
HoursYearDE <- 1386
# Number of employees US in 2019
EmployeesDE <- 41061000
avgprodDE1 <- GDPDE/(HoursYearDE*EmployeesDE)
set.seed(314)
avgprodDE <- runif(numberrandom, min = avgprodDE1, max = avgprodDE1)
meanavgprodDE <- mean(avgprodDE)
meanavgprodDE
```

```
## [1] 68.54789
```

3.7.2 Productivity U.S.

(U.S. Bureau of Labor Statistics; BLS)

GDP per hour worked: Our World in data

- Annual hours worked per worker in 2019: US 1779 h/year (OECD 2021: (accessed on 22 March 2021)
- Employment U.S. in 2019 147194 k (OECD 2021:) (accessed on 22 March 2021)
- OECD (2021), Gross domestic product (GDP) (indicator). U.S. 2019: 21433226 million $ (accessed on 22 March 2021)
- Avg. prod per hour worked (own calculation)

GDP per hour worked is

```
# GDP in 2019 in $/year
GDPUS <- 21433226000000*EURDollar
# Avg. hours worked per worker and year (h/year) in 2019
HoursYearUS <- 1779
# Number of employees US in 2019
EmployeesUS <- 147194000
avgprodUS1 <- GDPUS/(HoursYearUS*EmployeesUS)
avgprodUS1
```

```
## [1] 68.75444
```

```
set.seed(314)
avgprodUS <- runif(numberrandom, min = avgprodUS1, max = avgprodUS1)
```

3.8 AVG: Wage

Only Full-time-equivalent employees (FTE)

3.8.1 Avg. wage and wage distribution Germany

- Average annual gross wage per worker DE: (current prices, OECD) 42421 EUR {, (accessed on 22 March 2021)}
- Avg. gross wage/h per FTE (full-time equivalent) worker in different sectors in Germany, 2019 (see **??**)
- skilled-dependent avg. gross wage/h per FTE in Germany, 2018 (see **??**)
- Minimum wage in 2019: 9.50 EUR/h DeStatis (2021)
- wage distribution in Germany, 2018 DeStatis (2018)
- Graphic below: own graph based on data from table 1.1.4 in DeStatis (2018). Wages of full time workers, in Agriculture, Manufacturing, and Services.

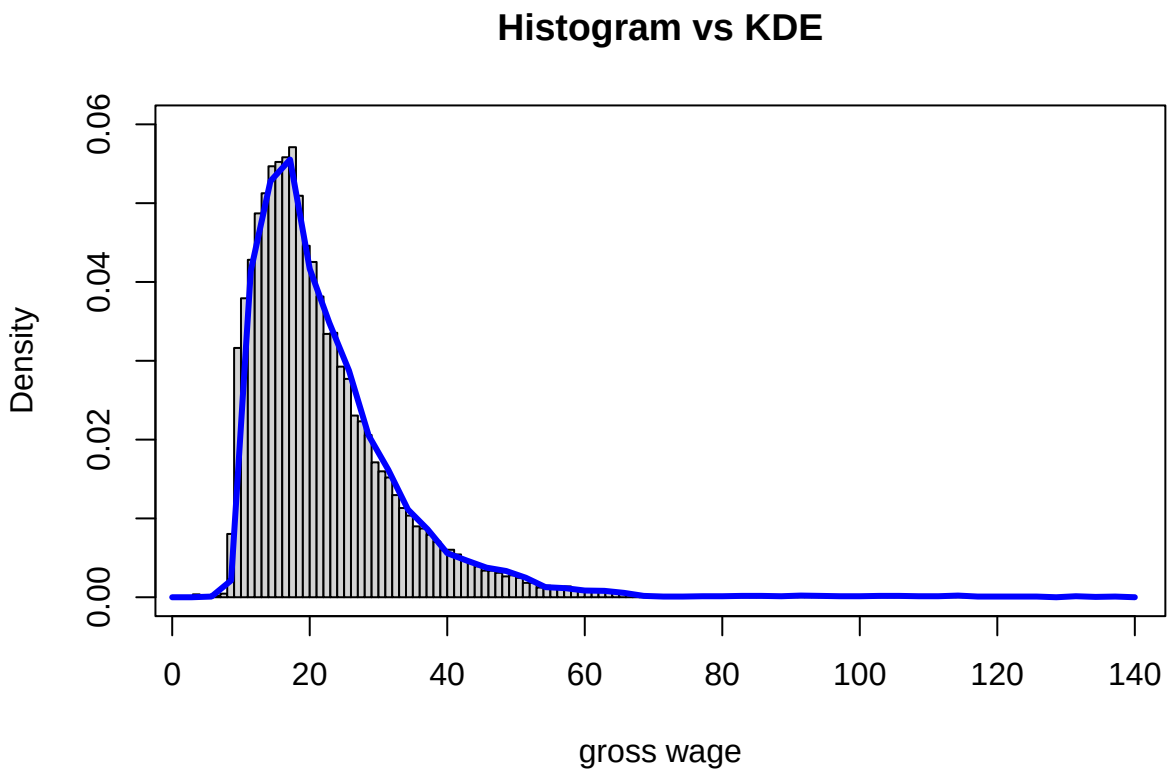
```
# Estimate kernel density of the German gross wage distribution
#(thanks to F. Javier Rubio: Kernel Density and Distribution Estimation for data
# with different supports, 11 August, 2017)

# Data from \citet{DeStatis2018}, Tab 1.1.4
datawageDE <- read.table("WagesHistogramDestatis2018FSTab114.csv", sep = ";", header = TRUE)

#install.packages('kerdiest')
library(kerdiest)

datawageDE2.data <- c(runif(datawageDE[1,2],datawageDE[1,3]-0.4,datawageDE[1,3]+0.4))
for (n1 in 2:135) {
  datawageDE2.data <- c(datawageDE2.data,
                        c(runif(datawageDE[n1,2],datawageDE[n1,3]-0.4,datawageDE[n1,3]+0.4)))
}
# calculate bandwith
# x quantiles
# h,b0 smoothing, bandwith
```

Figure 2: Hourly wages for FTE in Germany, 2018

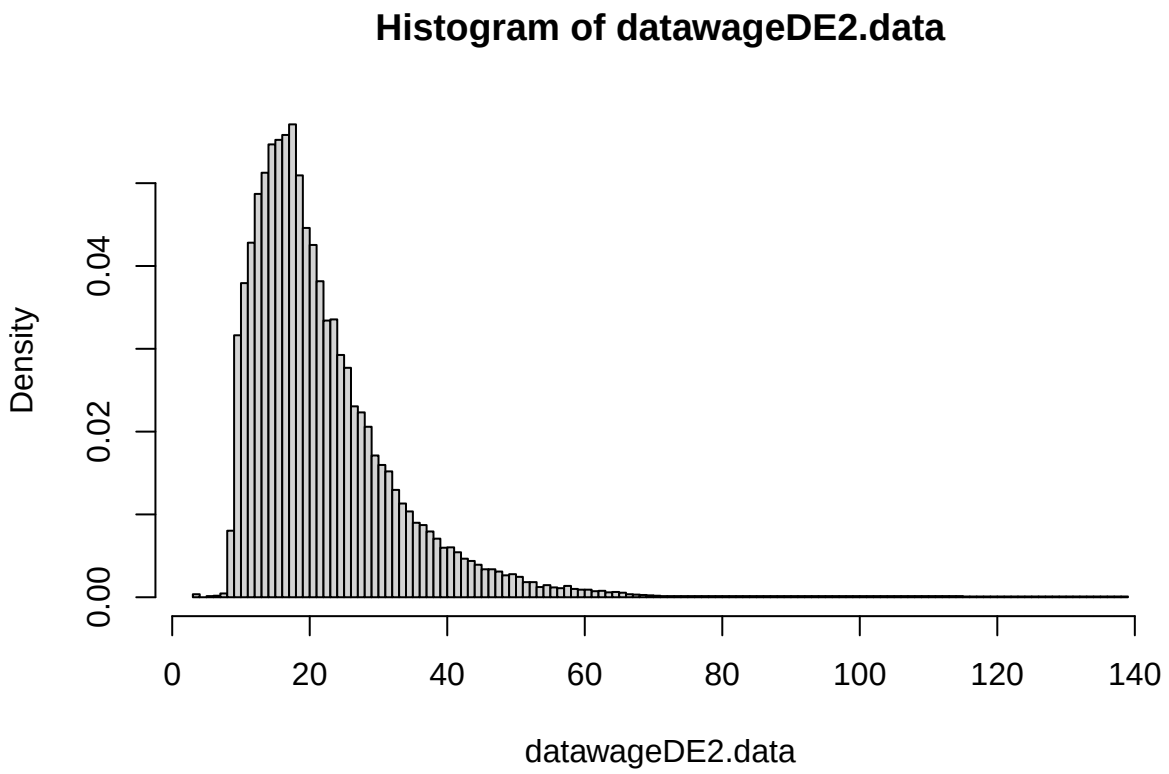
```
kde.dunif <- function(x,h,data) mean( dunif(x-data,0,h))

b0 <- bw.nrd0(datawageDE2.data)
b0
```

```
## [1] 1.070197
```

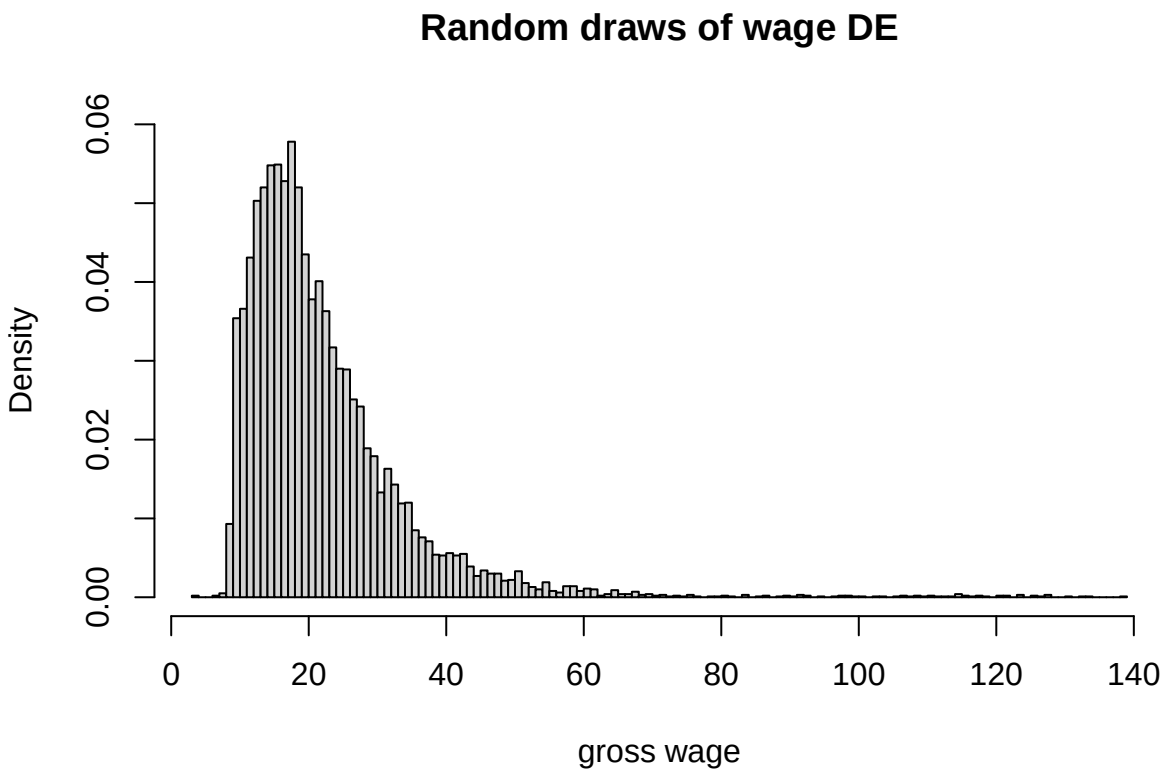
```
kde.dfit <- Vectorize(function(x) kde.dunif(x,b0,datawageDE2.data))
hist(datawageDE2.data,probability = T, breaks = 135, ylim = c(0,0.06), xlab = "gross wage", ylab = "Dens
curve(kde.dfit,0,140,lwd=3,col="blue",add=T,n=50)
box()
```




```
ww <- kde.dunif(100,b0,datawageDE2.data)

# 10000 random draws from the kernel density
wagehist <- hist(datawageDE2.data,breaks=135,freq=FALSE)
```




```
set.seed(17)
samplesize = 10000
bins=with(wagehist,sample(length(mids),samplesize,p=density,replace=TRUE))
# choose a bin
wageDE =runif(length(bins),wagehist$breaks[bins],wagehist$breaks[bins+1])
# sample a uniform in it
hist(wageDE,probability = T, breaks = 100, ylim = c(0,0.06), xlab = "gross wage", ylab = "Density", mai
```




```
#{r wageDE message=FALSE, warning=FALSE}
# # calculate gamma distribution for avg wage
# wageminimumDE <- 9.50
# wagelowerDE <- 11.54
# wagemeanDE <- 25.87
# wageupperDE <- 73.18
# # avg. wage OECD and own calculations ->
# # wagemeanDE <- 42421/HoursYearDE
# # interval [wagelowerDE,wagehigherDE]
# # with non linear maximization
# # use wagelower as 12.1%, wagemean as 54.028%, and wageupper as 99.7% percentiles
# # quantiles
# pp <- c(0.121,0.54028,0.9964)
# #p <- c(0.121,0.66,0.997)
# #p <- c(0.0,0.5,0.95)
# # p=c(2,0.1) are start values for shape and rate
# # draw vertical lines at p
# solution <- nlm(f=errorGamma, p=c(2,0.1), quantiles=pp,
#                 exp=c(wagelowerDE, wagemeanDE, wageupperDE),print.level=0)
# shape.wageDE <- solution$estimate[1]
# rate.wageDE  <- solution$estimate[2]
# plotGamma(shape.wageDE, rate.wageDE, p=p)
# # use Gamma-distribution for random draws
# set.seed(321)
# wageDE <- rgamma(numberrandom,shape.wageDE,rate.wageDE)
# meanwageDE <- mean(wageDE)
```

```
# meanwageDE
# # draw histogram
# hist(wageDE, breaks=60, main="", xlab="wageDE - random numbers from a truncated normal")
# png("imgwageDE.png")
# hist(wageDE, breaks=60, main="", xlab="wageDE - random numbers from a truncated normal")
```

3.8.2 Avg wage and wage distribution US

Average hourly wage of FTE in 2019[a]		
Occupation	wage $/h	rel. s.e.
Full-time business/financ operations occuption Bloomington, IN	31.53	1.5
Full-time business/financ operations occuption (U.S.)	37.65	–
Level 06 management occup (U.S.)	16.31	–
Level 06 accountants and auditors (U.S.)	24.88	2.5
Level 09 accountants and auditors (U.S.)	36.87	2.8
Level 12 accountants and auditors (U.S.)	69.98	12.8
Level 06 computer/math occup. Austin-Round Rock, TX	22.67	5.7
Level 08 computer/math occup. Austin-Round Rock, TX	32.54	2.8
Level 11 computer/math occup. Austin-Round Rock, TX	51.78	1.8
Avg. worker (U.S. - average; Q4)	25.72[b]	1.1
Compensation (soc. benefits; Q4)	11.03	1.5

[a] US BoL (2021a)

[b] US BoL (2021a)

Table 4: Average hourly wage in office jobs,U.S. 2019

Median weekly wage of FTE workers in 2019, U.S.		
Education	$/week	
High school graduates, no college	747	
Bachelor's degree or higher	1382	
Bachelor's degree only	1259	
Advanced degree	1571	

US BoL (2021a)

25 years and over

State and county, avg. wages all workers are available

Avg. hours worked, business/financ. services = 34.4 h/week

Table 5: Average hourly wage in office jobs, 2019

Wage distribution in the U.S.

- Average annual wage per worker U.S.: 65836 USD
  OECD (current prices, 2020 2021) → used for calculating the average wage
- Avg. hourly wage for some occupation groups, FTE U.S., 2019/4 (see 4)
- Avg. hourly wage according to education level, FTE U.S., 2019/4 (see 5)
- minimum wage in 2019: 7.25 USD/h US DoL (see 2021)

We use Table 4 for low and high wages (used as percentiles)

Data on the wage distribution see also Gould (2020).

```
# calculate gamma distribution for avg, wage
wagelowerUS  <- 10.07*EURDollar
wageupperUS <- 67.14*EURDollar
wageminimumUS <- 7.25*EURDollar
```

Percentile	USD/h
10	10.07
20	12.31
30	14.64
40	16.71
50	19.33
60	22.95
70	27.94
80	34.98
90	48.08
95	67.14

Source: (Gould, 2020)
Avgl. wages per percentiles in 2019 in 2019 .

Table 6: Wages US, 2019

```
# OECD annual avg. wage / avg. work hours per year:
wagemeanUS <- 25.72
wagemeanUS
```

```
## [1] 25.72
```

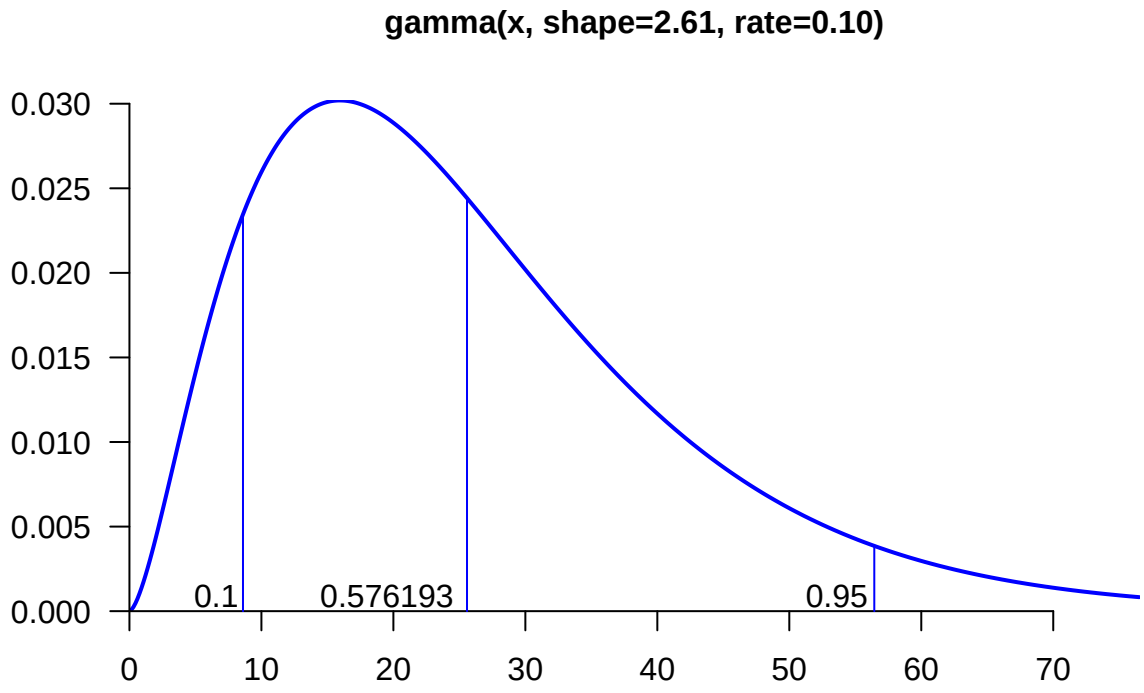
```
wagelowerUS
```

```
## [1] 8.4588
```

```
wageupperUS
```

```
## [1] 56.3976
```

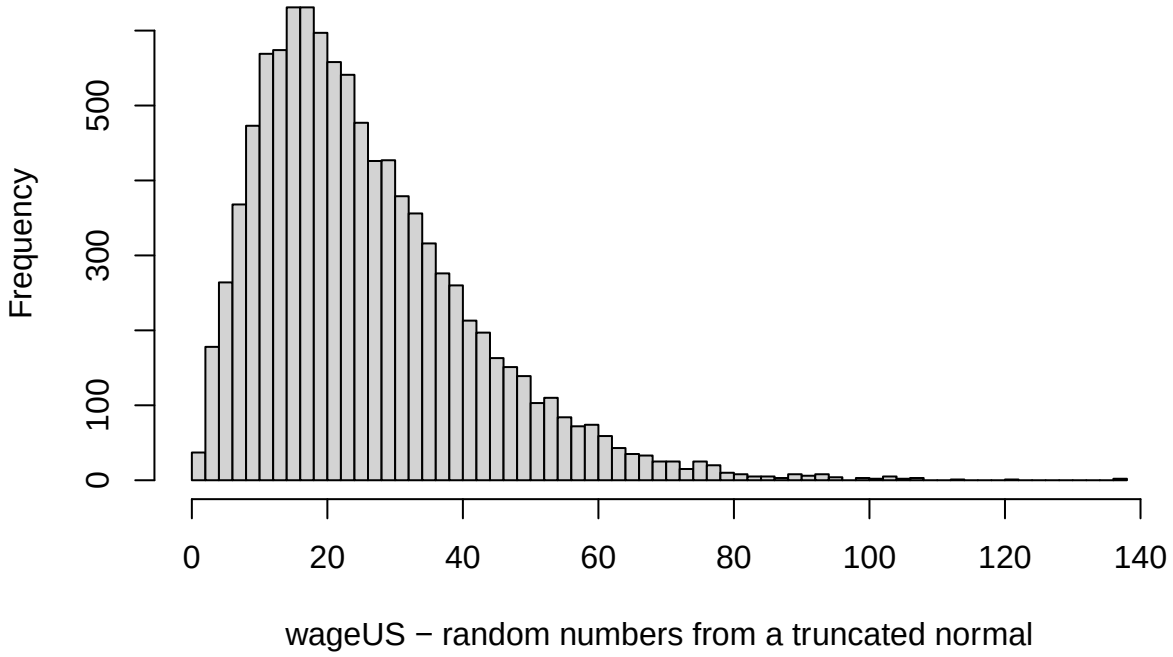
```
# 65836/HoursYearUS*EURDollar
# interval [wagelower,wageupper] whith borders at 10% and 95% percentile
# and mean 0.66 (25.72 is between 60th and 70th percentile)
# with non linear maximization
# use wagelower as 5%, wagemean as 57.6193%, and wageupper as 95%
p <- c(0.10,0.576193,0.95)
# p=c(2,0.1) are start values for shape and rate
solution <- nlm(f=errorGamma, p=c(2,0.1), quantiles=p,
                exp=c(wagelowerUS, wagemeanUS, wageupperUS),print.level=0)
shape.wageUS <- solution$estimate[1]
rate.wageUS  <- solution$estimate[2]
plotGamma(shape.wageUS, rate.wageUS, p=p)
```




```
# use Gamma-distribution for random draws
set.seed(321)
wageUS <- rgamma(numberrandom,shape.wageUS,rate.wageUS)
meanwageUS <- mean(wageUS)
meanwageUS
```

```
## [1] 25.72001
```

```
# plot histogram
hist(wageUS, breaks=60, main="", xlab="wageUS - random numbers from a truncated normal")
```




```
png("imgwageUS.png")
hist(wageUS, breaks=60, main="", xlab="wageUS - random numbers from a truncated normal")
```

3.9 AVG wage tax rate (including social security payments)

Only Full-time-equivalent employees (FTE)

3.9.1 Avg. wage and social security rates on gross-wages Germany

OECD (2020)

- single earner (Avg. income), 42651 USD annual taxable income (gross wage minus employers social insurance contributions). 10037 USD income tax, 10476 USD soc security contributions = 20514 USD payments to government
- family with 2 children: ** 1 earner with avg. wage, 42746 USD, 786 USD income tax, 19346 USD soc sec. ** 2 earner with 100% and 67% income: 55863 USD taxable income, 9807 USD income tax, 17277 USD soc sec. = 27084 USD payments to government

```
# Simply calculate average of a single earner with avg wage and a family with two earners, one earning
wageEURDE = 0.5*(55863 + 42561)*EURDollar
wageEURDE
```

```
## [1] 41338.08
```

```
taxliabilityDE = 0.5 * (20514+27084)*EURDollar
taxliabilityDE
```

```
## [1] 19991.16
```

```
avgtaxrateDE = taxliabilityDE/wageEURDE
avgtaxrateDE
```

```
## [1] 0.4836016
```

3.9.2 Avg. wage and social security rates on gross-wages U.S.

OECD (2020)

- single earner (Avg. income), 57055 USD annual taxable income (gross wage minus employers social insurance contributions). 5727 USD income tax, 3608 states and local taxes, 4365 USD soc security contributions = 13699 USD payments to government
- family with 2 children: ** 2 earner with 100% and 67% income: 95282 USD taxable income, 4118 USD income tax, 5590 state and local taxes, 7289 USD soc sec. = 16997 USD payments to government

```
# Simply calculate average of a single earner with avg wage and a family with two earners, one earning
wageEURUS = 0.5*(57055 + 95282)*EURDollar
wageEURUS
```

```
## [1] 63981.54
```

```
taxliabilityUS = 0.5 * (13699+16997)*EURDollar
taxliabilityUS
```

```
## [1] 12892.32
```

```
avgtaxrateUS = taxliabilityUS/wageEURUS
avgtaxrateUS
```

```
## [1] 0.2015006
```

4. Office Parameters

4.1. Monthly rent per sqm of office space

4.1.1 Germany (rmonthDE)

In Germany, data on office rents are not publicly available. There is a small number of statistics aoffered by private firms (see Table 7). Nothing is known about the distribution of officen rents. We assume, that rents span equally between the lowest rent and highest rent in the seven largest cities (Big 7). Of course, intuitively there is less office space in the upper segment.

We do not have any information on the distribution of the rents paid. Therefore, we use a very simply approach. We assume that rents in each of the i-City-Segment is normally distributed around the mean gross rent but is truncated at the minimum and maximum rent level. We have these in the A-city segment. In the B-City segment there is no minimum and no mean, while the maximum is taken from the city of Essen and the mean from the city of Leipzig. We use the very low minimum from the A-city segment also as minimum for the B-city segment. We have only maximum rents for C- and D-cities. We use 5 EUR as minimum (slightly lower than in the A-cities). Then, we assume equal distribution of rents in C- and D-cities. Eventually, we draw for each i-City segment a share of 10000 random draws from the assumed densities. The share is calculated from the share of each segment on all newly rented sqm in 2019/20. The share of C- and D-cities is calculated with the assumption that C-cities' accumulated share of newly rented sqm is three times as large as the share of D-cities sqm.

```
# Calculate city types' share on newly rented office space
shareNewlyRentedACities <- 4.029/6.1
shareNewlyRentedBCities <- 1.2/6.1
shareNewlyRentedCCities <- 0.75*(6.1-4.029-1.2)/6.1
```

Rents office space 2019-2020 [EUR/m^2]						
City	average EUR/qm	max. EUR/qm	lowest EUR/qm	avg.add. costs sqm	available mill.sqm	newly rented mill.sqm
Berlin	28.70[c]	38,00[k]	9.00[k]	3.68[m]	20.80[k]	0.999[k]
Cologne	15.90[h]	26.00[k]	6.50[k]	3.51[m]	7.84[k]	0.291[k]
Düsseldorf	16.80[f]	28.50[g,k]	7.00[k]	3.57[m]	9.24[k]	0.550[k]
Frankfurt (Main)	22.89[a]	41.50[k]	9.00[k]	3.81[m]	11.63[k]	0.580[k]
Hamburg	17.40[e]	31.00[k]	6.00[k]	3.64[m]	15.14[k]	0.530[k]
Munich	20.50[b]	41.00[k]	9.50[k]	3.82[m]	20.96[b,k]	0.760[k]
Stuttgart	20.40[i]	25.50[k]	9.00[k]	3.51[m]	8.84[k]	0.319[k]
Essen (B-City)	12.00[e]	16.00[e]				
Leipzig (B-City)	10.80[e]	15.50[e]				
A-Cities		34.30[n]				4.03[o]
B-Cities		15.20[n]				1.2[o]
C-Cities		13.40[l]				
D-Cities		10.50[l]				
All 127 cities						6.1[o]

[a] Source: Statista (2021c), Rent 2020
[b] Source: Statista (2021c), Rent 2019
[c] Source: Statista (2021e), Rent 2019
[d] Source: Statista (2021f), Rent 2020
[e] Source: Statista (2021g), Rent 2020
[f] Source: Statista (2021h), Rent 2020
[g] Source: Statista (2021i), Rent 2020
[h] Source: Statista (2021j), Rent 2020
[i] Source: Statista (2021k), Rent 2019
[j] Source: Statista (2021l), Rent 2019
[k] Source: JLL (2021), Rent 2020
[l] Source: BNP (2021), Rent 2020
[m] Source: JLL (2018), 2018
[j] Source: Statista (2021m), 2019, 3% in highest segment
[o] Source: (Feld et al., 2020(@) , 2019

Note A-cities account for about 66% of new rent are in 2019, B-cities for 10%, C-cities for 5%.
(Feld et al., 2020(@)
B-Cities (14): Bochum, Bonn, Bremen, Dresden, Dortmund, Duisburg, Essen, Hannover, Karlsruhe, Leipzig, Mannheim, Nürnberg, Münster, Wiesbaden

Table 7: Office rents of new contracts, Germany, 2019 and 2020

```r
shareNewlyRentedDCities <- 0.25*(6.1-4.029-1.2)/6.1

# Importing the dplyr library (to add columns from manipulation of existing data columns)
library(dplyr)
```

```
##
## Attache Paket: 'dplyr'
```

```
## Die folgenden Objekte sind maskiert von 'package:pastecs':
##
##     first, last
```

```
## Das folgende Objekt ist maskiert 'package:MASS':
##
##     select
```

```
## Die folgenden Objekte sind maskiert von 'package:stats':
##
##     filter, lag
```

```
## Die folgenden Objekte sind maskiert von 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
# data base (Table Office Rents Germany)
# note: addcosts of B-C cities is assumed to be 3.5
dfRentDE = data.frame(
  "City" = c("Berlin","Cologne","Dusseldorf","Frankfurt","Hamburg", "Munich","Stuttgart","ACities","BCi
  "avgrent" = c(28.700, 15.900, 16.80, 22.89, 17.40, 20.50, 20.400, NaN, NaN, NaN, NaN),
  "upperrent" = c(38.000, 26.000, 28.50, 41.50, 31.00, 41.00, 25.500, 34.30, 15.2, 13.4, 10.5),
  "lowestrent" = c(9.000, 6.500, 7.00, 9.00, 6.00, 9.50, 9.000, NaN, NaN, NaN, NaN),
  "addcosts" = c(3.680, 3.510, 3.50, 3.81, 3.64, 3.82, 3.510, 3.5, 3.5, 3.5,  3.5),
  "available" = c(20.800, 7.840, 9.24, 11.60, 15.14, 20.96,  8.840, NaN,  NaN,  NaN,  NaN),
  "newlyrented" = c(0.999,  0.291,  0.55, 0.58, 0.53, 0.76, 0.319,  4.029, 1.2,  NaN,   NaN)
)
dfRentDE
```

```
##          City avgrent upperrent lowestrent addcosts available newlyrented
## 1      Berlin   28.70      38.0        9.0     3.68     20.80       0.999
## 2     Cologne   15.90      26.0        6.5     3.51      7.84       0.291
## 3  Dusseldorf   16.80      28.5        7.0     3.50      9.24       0.550
## 4   Frankfurt   22.89      41.5        9.0     3.81     11.60       0.580
## 5     Hamburg   17.40      31.0        6.0     3.64     15.14       0.530
## 6      Munich   20.50      41.0        9.5     3.82     20.96       0.760
## 7   Stuttgart   20.40      25.5        9.0     3.51      8.84       0.319
## 8     ACities     NaN      34.3        NaN     3.50       NaN       4.029
## 9     BCities     NaN      15.2        NaN     3.50       NaN       1.200
## 10    CCities     NaN      13.4        NaN     3.50       NaN         NaN
## 11    DCities     NaN      10.5        NaN     3.50       NaN         NaN
```

```r
# gross rents
avggross <- dfRentDE$avgrent + dfRentDE$addcosts
upgross <- dfRentDE$uperrent + dfRentDE$addcosts
lowgross <- dfRentDE$lowerrent + dfRentDE$addcosts
avggross
```

```
##  [1] 32.38 19.41 20.30 26.70 21.04 24.32 23.91   NaN   NaN   NaN   NaN
```

```
lowgross
```

```
## numeric(0)
```

```
upgross
```

```
## numeric(0)
```

```
newavggross <- dfRentDE$newlyrented * avggross
dfRentDE <- mutate(dfRentDE, avgGrossRent = avgrent + addcosts, upperGrossRent = upperrent + addcosts,
dfRentDE
```

```
##          City avgrent upperrent lowestrent addcosts available newlyrented
## 1      Berlin   28.70      38.0        9.0     3.68     20.80       0.999
## 2     Cologne   15.90      26.0        6.5     3.51      7.84       0.291
## 3  Dusseldorf   16.80      28.5        7.0     3.50      9.24       0.550
## 4   Frankfurt   22.89      41.5        9.0     3.81     11.60       0.580
## 5     Hamburg   17.40      31.0        6.0     3.64     15.14       0.530
## 6      Munich   20.50      41.0        9.5     3.82     20.96       0.760
## 7   Stuttgart   20.40      25.5        9.0     3.51      8.84       0.319
## 8     ACities     NaN      34.3        NaN     3.50       NaN       4.029
## 9     BCities     NaN      15.2        NaN     3.50       NaN       1.200
## 10    CCities     NaN      13.4        NaN     3.50       NaN         NaN
## 11    DCities     NaN      10.5        NaN     3.50       NaN         NaN
##    avgGrossRent upperGrossRent lowestGrossRent
## 1         32.38          41.68           12.68
## 2         19.41          29.51           10.01
## 3         20.30          32.00           10.50
## 4         26.70          45.31           12.81
## 5         21.04          34.64            9.64
## 6         24.32          44.82           13.32
## 7         23.91          29.01           12.51
## 8           NaN          37.80             NaN
## 9           NaN          18.70             NaN
## 10          NaN          16.90             NaN
## 11          NaN          14.00             NaN
```

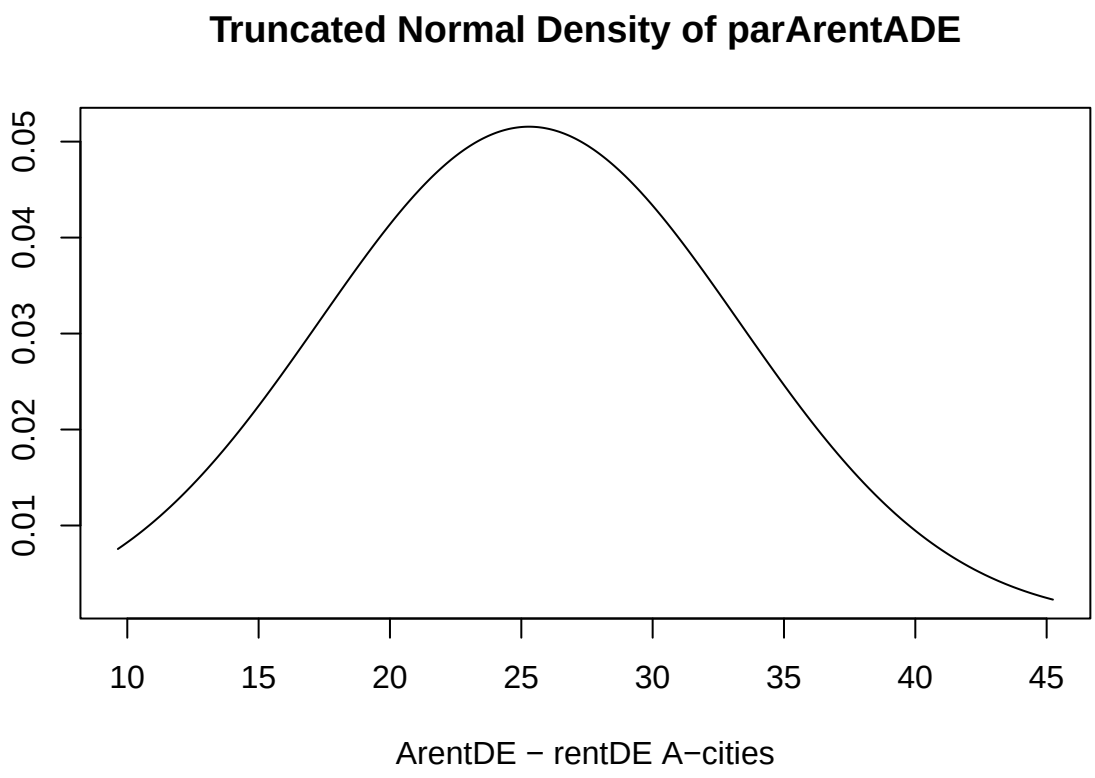
```
### Calculations for A-city type
# an A-city's share of new rented office space
idx <- seq(from = 1, to = 7, by = 1)
dfRentDE <- mutate(dfRentDE, shareNewly = newlyrented/sum(newlyrented[idx],na.rm=TRUE))
# A-cities's average gross rent
SumavgGrossRentACities <- sum(dfRentDE$avgGrossRent[idx] * dfRentDE$shareNewly[idx],na.rm=TRUE)

# Calculate truncated normal for A-cities
lower1 <- min(dfRentDE$lowestGrossRent[idx],na.rm=TRUE)
lowerA <- lower1
upper1 <- max(dfRentDE$upperGrossRent[idx],na.rm=TRUE)
mean1 <- SumavgGrossRentACities
# lower1 determines the 2.5% percentile(1.96);
up1 = mean1 + (mean1-lower1)
sd1 <- (lower1-mean1)/qnorm(0.025)
sd1
```

```
## [1] 7.986767
```

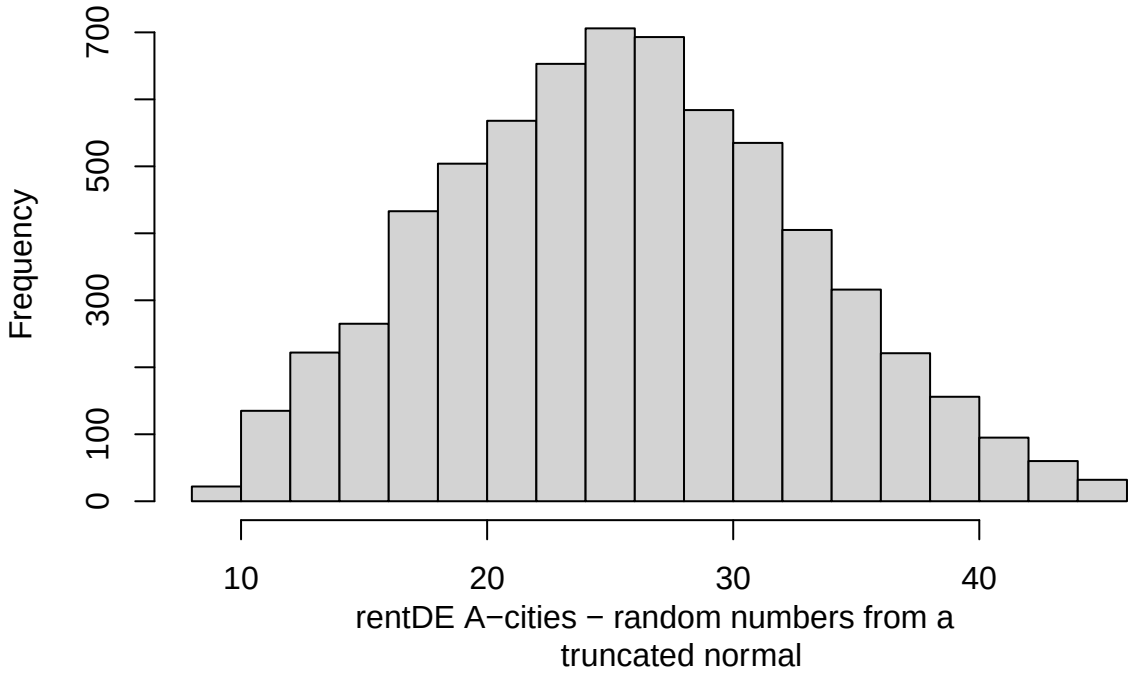
```r
# 2. 10000 random draws for share of A-city type
numberA <- round(numberrandom * shareNewlyRentedACities, digits=0)
#checkno <- numberrandom - (numberA + numberB + numberC + numberD)
#### random numbers and plots for A city
set.seed(423)
upper1
```

```
## [1] 45.31
```

```r
parArentDE <- rtruncnorm(numberA, a = lower1, b=upper1, mean = mean1, sd =sd1)
## sequence of values between lower and upper bounds
xgx <- seq(from=lower1, to=upper1, by=0.1)
#xgx
## vector of values of the heigth of the probability distribution for each value of x
ygx=dtruncnorm(xgx, a=lower1, b=upper1,mean=mean1,sd=sd1)
#ygx
## plot density
plot(xgx, ygx, type = "l", xlab ="ArentDE - rentDE A-cities",ylab = "",
     main="Truncated Normal Density of parArentADE")
```



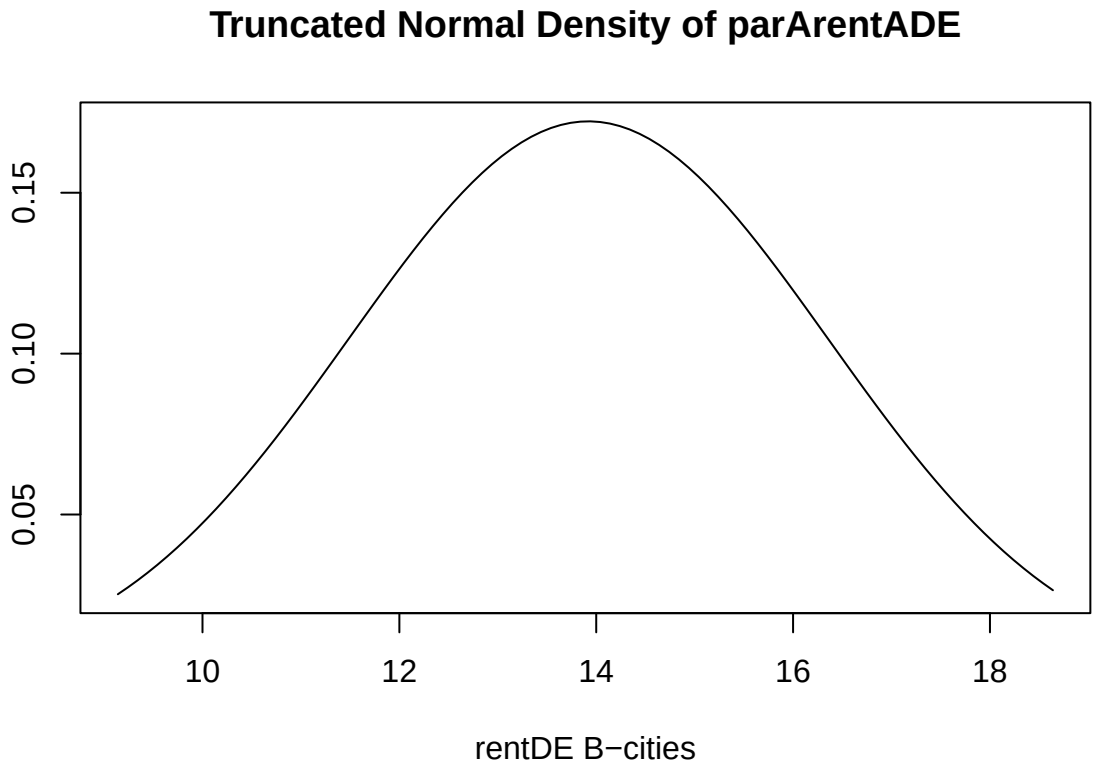

```r
# plot histogram
hist(parArentDE, main="", xlab="rentDE A-cities - random numbers from a
     truncated normal")
```



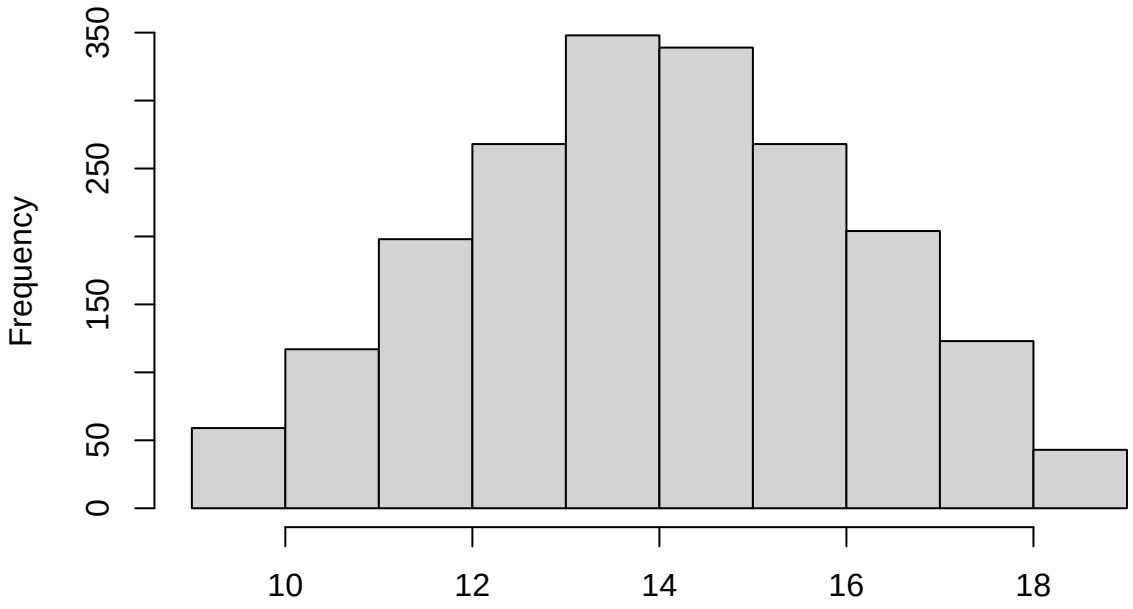

```
### Calculations for B-city type

# Calculate truncated normal for B-cities
lower1 <- min(lowerA-0.50,na.rm=TRUE)
upper1 <- max(dfRentDE$upperGrossRent[9],na.rm=TRUE)
mean1 <- 1/2 * (upper1 + lower1)
# lower1 determines the 2.5% percentile of a normal distribution
sd1 <- (lower1-mean1)/qnorm(0.025)
sd1
```

```
## [1] 2.43882
```

```
# 2. 10000 random draws for share of B-city type
numberB <- round(numberrandom * shareNewlyRentedBCities, digits=0)
#### random numbers and plots for B cities
set.seed(123)
parBrentDE <- rtruncnorm(numberB, a = lower1, b=upper1, mean = mean1, sd =sd1)
## sequence of values between lower and upper bounds
xgx <- seq(from=lower1, to=upper1, by=0.1)
#xgx
## vector of values of the heigth of the probability distribution for each value of x
ygx=dtruncnorm(xgx, a=lower1, b=upper1,mean=mean1,sd=sd1)
#ygx
## plot density
plot(xgx, ygx, type = "l", xlab ="rentDE B-cities",ylab = "",
     main="Truncated Normal Density of parArentADE")
```



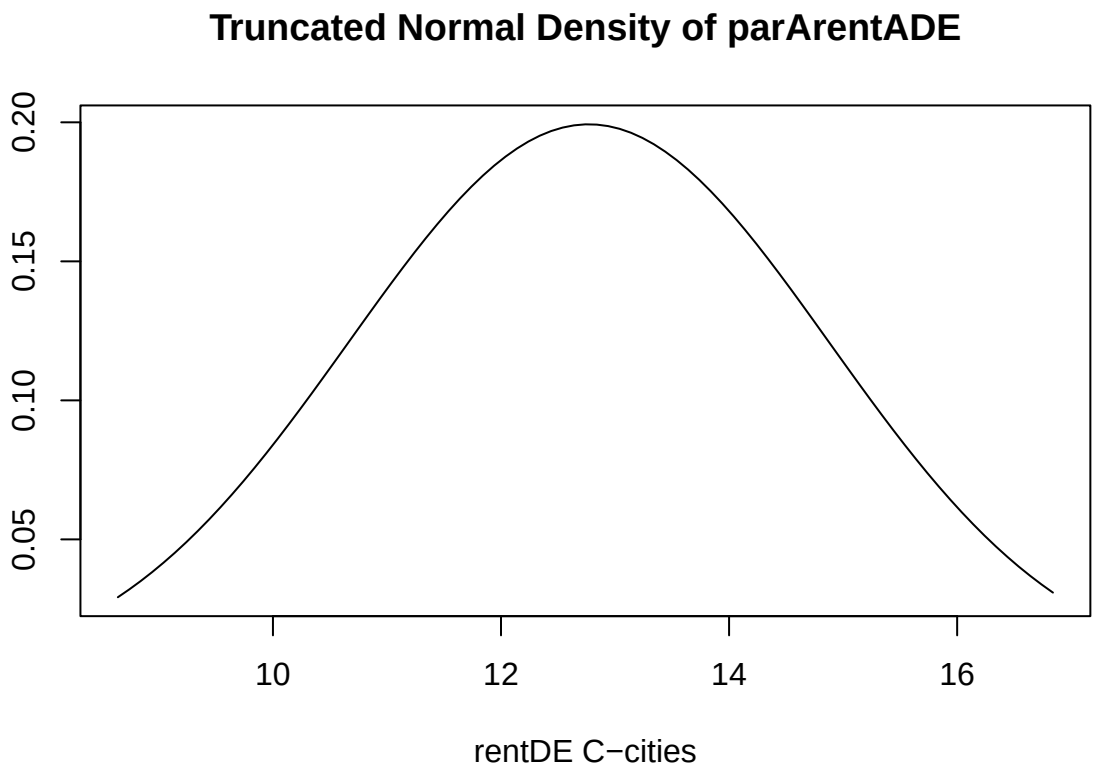

```
# plot histogram
hist(parBrentDE, main="", xlab="rentDE B-cities - random numbers from a truncated normal")
```



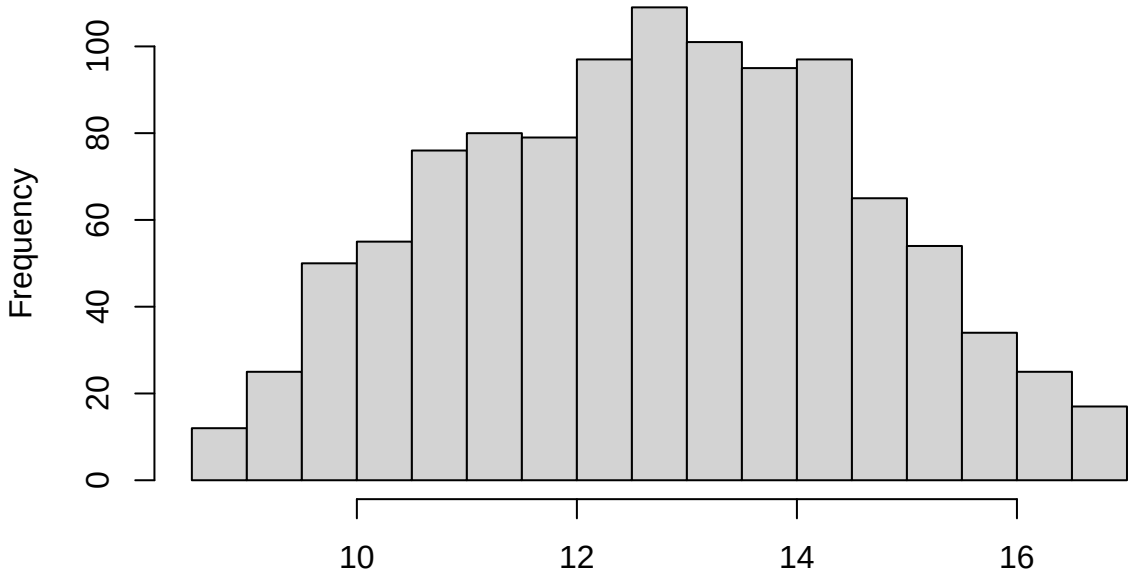

```
# Calculate truncated normal for C-cities
lower1 <- min(lowerA-1,na.rm=TRUE)
upper1 <- max(dfRentDE$upperGrossRent[10],na.rm=TRUE)
mean1 <- 1/2 * (upper1 + lower1)
# lower1 determines the 2.5% percentile of a normal distribution;
sd1 <- (lower1-mean1)/qnorm(0.025)
sd1
```

```
## [1] 2.107182
```

```
# 2. 10000 random draws for share of C-city type
numberC <- round(numberrandom * shareNewlyRentedCCities,digits=0)
#### random numbers and plots for C cities
set.seed(423)
parCrentDE <- rtruncnorm(numberC, a = lower1, b=upper1, mean = mean1, sd =sd1)
## sequence of values between lower and upper bounds
xgx <- seq(from=lower1, to=upper1, by=0.1)
#xgx
## vector of values of the heigth of the probability distribution for each value of x
ygx=dtruncnorm(xgx, a=lower1, b=upper1,mean=mean1,sd=sd1)
#ygx
## plot density
plot(xgx, ygx, type = "l", xlab ="rentDE C-cities",ylab = "",
     main="Truncated Normal Density of parArentADE")
```




```
# plot histogram
hist(parCrentDE, main="", xlab="rentDE C-cities - random numbers from a truncated normal")
```



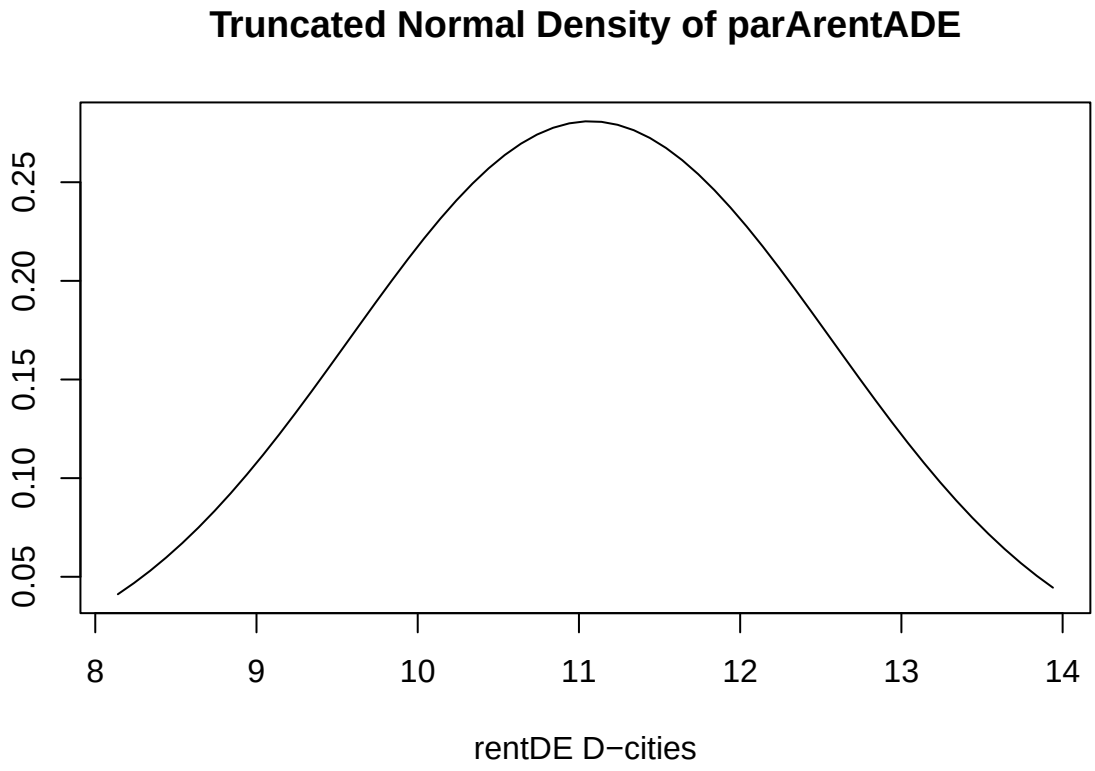

```
# Calculate truncated normal for D-cities
lower1 <- min(lowerA-1.5,na.rm=TRUE)
lower1
```

```
## [1] 8.14
```

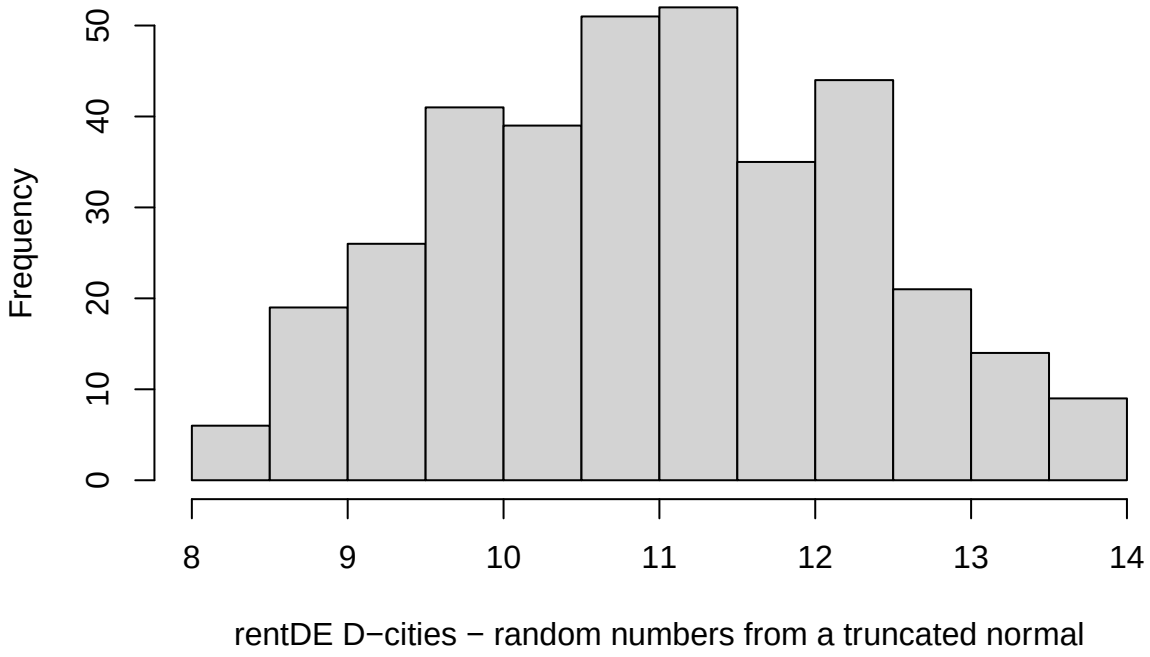
```
upper1 <- max(dfRentDE$upperGrossRent[11],na.rm=TRUE)
mean1 <- 1/2 * (upper1 + lower1)
# lower1 determines the 2.5% percentile of a normal distribution;
sd1 <- (lower1-mean1)/qnorm(0.025)
sd1
```

```
## [1] 1.494925
```

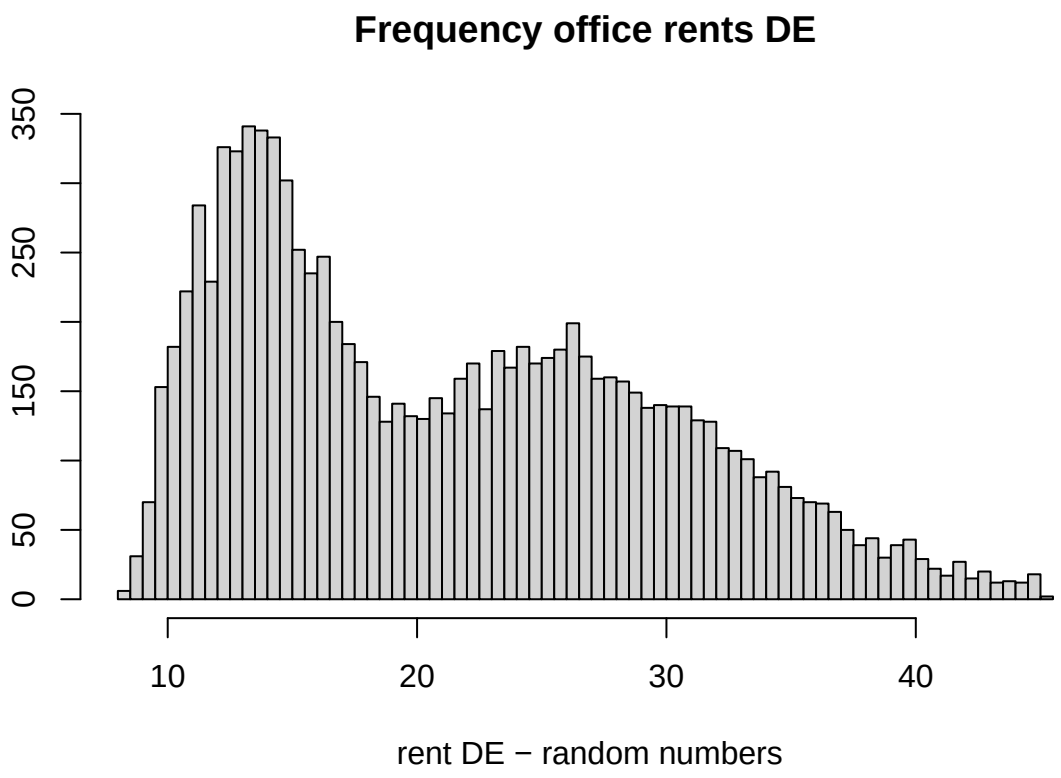
```
# 2. 10000 random draws for share of D-city type
numberD <- numberrandom - numberA - numberB - numberC
#### random numbers and plots for D cities
set.seed(423)
parDrentDE <- rtruncnorm(numberD, a = lower1, b=upper1, mean = mean1, sd =sd1)
## sequence of values between lower and upper bounds
xgx <- seq(from=lower1, to=upper1, by=0.1)
#xgx
## vector of values of the heigth of the probability distribution for each value of x
ygx=dtruncnorm(xgx, a=lower1, b=upper1,mean=mean1,sd=sd1)
#ygx
## plot density
plot(xgx, ygx, type = "l", xlab ="rentDE D-cities",ylab = "",
     main="Truncated Normal Density of parArentADE")
```




```
# plot histogram
hist(parDrentDE, main="", xlab="rentDE D-cities - random numbers from a truncated normal")
```




```
###
parrentDE <- c(parArentDE,parBrentDE,parCrentDE,parDrentDE)
hist(parrentDE, breaks=60, main="Frequency office rents DE", xlab=" rent DE - random numbers", ylab ="")
```




```
png("imgrentDE.png")
hist(parrentDE, breaks=60, main="Frequency office rents DE", xlab=" rent DE - random numbers", ylab ="")
summary(parrentDE)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   8.274  13.991  20.087  21.391  27.676  45.285
```

```
# random draws from parrentde
set.seed(423)
parRentDE <- sample(parrentDE)
mean2 <- mean(parRentDE)
mean2
```

```
## [1] 21.39064
```

4.1.2 U.S. (rmonthUS)

```
# sqm per sqft

dfRentUS = data.frame(
  "RankNational" = c(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25),
  "City" = c("New York Mid Manh", "New York Mid South Manh", "San Francisco Silicon Valley",
             "San Francisco city", "Boston Cambridge", "New York Downtown Manh",
             "Los Angeles Westside", "Boston city", "Washington D.C. Downtown",
             "Austin city", "Seattle Downtown", "Houston Downtown", "Seattle Suburban",
             "Chicago Downtown", "Houston Suburban", "Dallas Downtown",
             "Los Angeles Downtown", "Denver Downtown", "Atlanta Buckhead & Midtown",
```

Rank world	Rank national	City	District	max gross cost[a] USD/sqft p.a.	net rent[b] USD/sqft p.a.	avg.rent A[d] USD/sqft p.a.	avg.rent 2020[e] USD/sqft p.a.
4	1	New York	Midtown Manhattan	212[b]	170[c]	85[f]	83.82[f]
7	2	New York	Midtown South Manh.	182[b]	152[c]	95[f]	88.47[f]
11	3	San Francisco	Silicon Valley	162[b]	135[c]	69.95	85.87[g]
13	4	San Francisco	city	118[b]	85[c]	104.79	85.87[g]
17	5	Boston	Cambridge	103[b]	85[c]		
18	6	New York	Downtown Manhattan	107[b]	87[c]	68[f]	64.64[f]
20	7	Los Angeles	Westside	104[b]	86[c]	39.82	
17	8	Boston	city	100[b]	78[c]	71.20	44.06[g]
22	9	Washington D.C.	Downtown	88[b]	65[c]	60.39	59.42[g]
	10	Austin	city	87[b]	60[c]	52.85	49.93
42	11	Seattle	Downtown	66.47	51.50		45.59[g]
45	12	Houston	Downtown	64.60		45.67	31.97[g]
46	13	Seattle	Suburban	64.20	43.16		45.59[g]
	14	Chicago	Downtown	76[b]	50[c]	48.81	43.99
	15	Houston	Suburban	56.88	32[c]	32.66	31.97[g]
	16	Dallas	Downtown	56.25	42[b]	29.21	31.08[g]
	17	Los Angeles	Downtown	69[b]	48[c]	46.69	
	18	Denver	Downtown	51.50	40[c]		31.87[g]
	19	Atlanta	Buckhead and Midtown	50.14		37.60	29.35[g]
	20	Boston	Suburans	48.06		31.49	44.06[g]
	21	Dallas	Suburban	45.93		31.83	31.08[g]
	22	Auckland	city	40[b]	30[c]		
	23	Atlanta	Suburban	37.76		29.41	29.35[g]
	24	Chicago	Suburban	30.50			25.93
	25	Denver	Suburban	30.25		28.56	31.87[g]

Occupancy cost include service charges, taxes. They are standardized on a net internal area basis.
Prime office occupance costs are costs for the highest-quality office space in a prime location
Top 1: Hongkong 208.67, Top 2: London (West End) 222.70, Top 50 Frankfurt: 61.77
[a] Source: CBRE (2019)
Source: [b] JLL (2019)
Source: [c] Approximation from JLL (2019)
Source: [d] Colliers (2019)
Source: [d] JLL (2020a)
Source: [f] JLL (2020b) in 2020
Source: [g] average over the whole city

Table 8: Prime Office Occupancy Cost - top U.S. City Districts

```
            "Boston Suburban", "Dallas Suburban", "Auckland city", "Atlanta Suburban",
            "Chicago Suburban", "Denver Suburban"),
  "occupancycost" = c(212, 182, 162, 118, 103, 107, 104, 100,  88, 87, 66.47, 64.60,
                      64.20, 76, 56.88, 56.25, 69,
  51.50, 50.14, 48.06, 45.93, 40, 37.76, 30.50, 30.25))
  "net rent" = c(170, 152, 135, 85, 85, 87, 86, 78, 65, 60, NaN, NaN, NaN, 50, 32,
                 42, 48, 40, NaN, NaN, NaN, 30, NaN, NaN, NaN)
dfRentUS
```

```
##    RankNational                         City occupancycost
## 1             1            New York Mid Manh        212.00
## 2             2      New York Mid South Manh        182.00
## 3             3 San Francisco Silicon Valley        162.00
## 4             4           San Francisco city        118.00
## 5             5             Boston Cambridge        103.00
## 6             6       New York Downtown Manh        107.00
## 7             7         Los Angeles Westside        104.00
## 8             8                 Boston city        100.00
## 9             9     Washington D.C. Downtown         88.00
## 10           10                 Austin city         87.00
## 11           11             Seattle Downtown         66.47
## 12           12             Houston Downtown         64.60
## 13           13             Seattle Suburban         64.20
## 14           14             Chicago Downtown         76.00
## 15           15             Houston Suburban         56.88
## 16           16              Dallas Downtown         56.25
## 17           17         Los Angeles Downtown         69.00
## 18           18              Denver Downtown         51.50
## 19           19   Atlanta Buckhead & Midtown         50.14
## 20           20              Boston Suburban         48.06
## 21           21              Dallas Suburban         45.93
## 22           22                Auckland city         40.00
## 23           23             Atlanta Suburban         37.76
## 24           24             Chicago Suburban         30.50
## 25           25              Denver Suburban         30.25
```

```
# rents in EUR per sqm per month
dfRentUS <- mutate(dfRentUS, maxGrossRent = occupancycost * sqmsqft * EURDollar/12)
dfRentUS$maxGrossRent
```

```
##  [1] 159.73628 137.13209 122.06263  88.90981  77.60772  80.62161  78.36119
##  [8]  75.34730  66.30562  65.55215  50.08335  48.67436  48.37297  57.26395
## [15]  42.85754  42.38286  51.98964  38.80386  37.77914  36.21191  34.60701
## [22]  30.13892  28.45114  22.98093  22.79256
```

```
lower1 <- min(dfRentUS$maxGrossRent)
upper1 <- max(dfRentUS$maxGrossRent)
med1 <- median(dfRentUS$maxGrossRent)
med1
```

```
## [1] 50.08335
```

```
mean1 <- 1/2 * (upper1 + lower1)
lower1
```

```
## [1] 22.79256
```

```r
upper1
```

```
## [1] 159.7363
```

```r
mean1
```

```
## [1] 91.26442
```

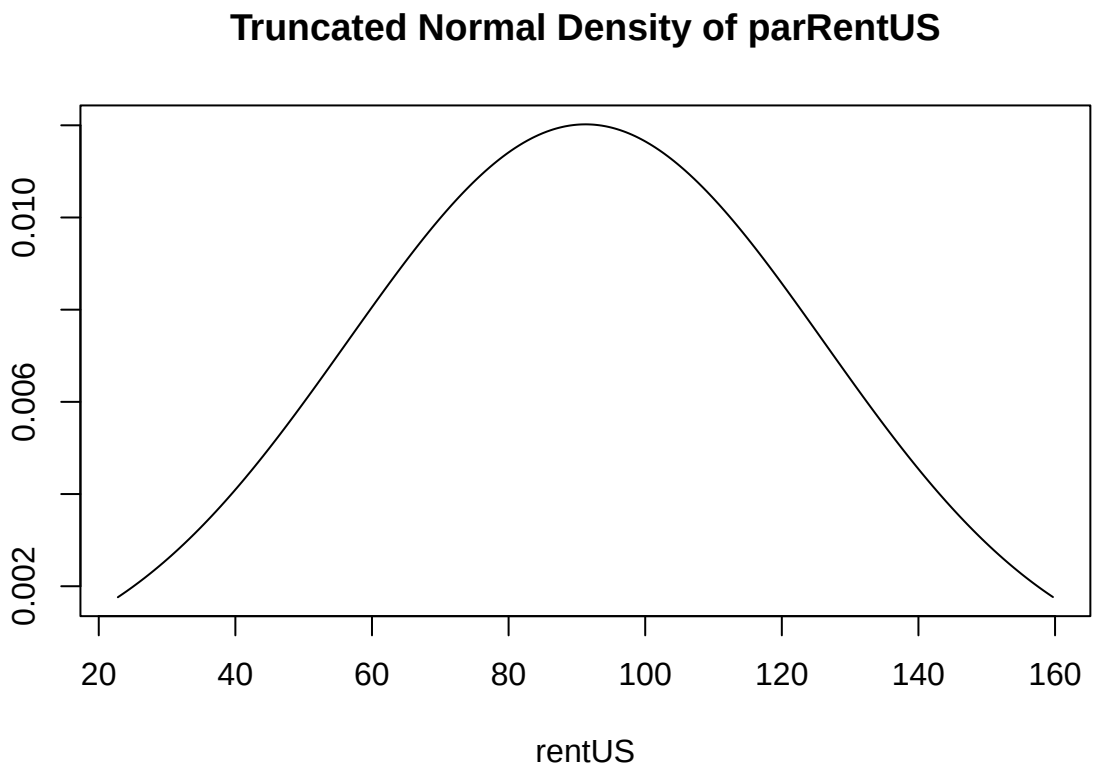
```r
# lower1 determines the 2.5% percentile of a normal distribution;
sd1 <- (lower1-mean1)/qnorm(0.025)
sd1
```

```
## [1] 34.93526
```

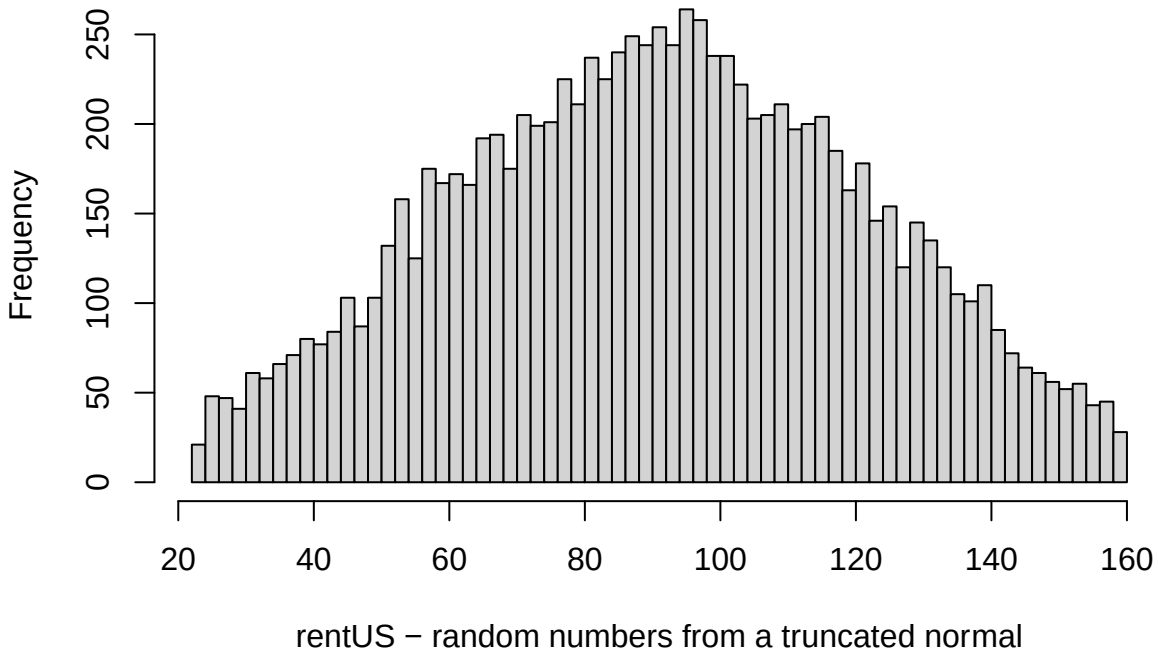
```r
# 2. 10000 random draws for share of D-city type
#### random numbers and plots for D cities
set.seed(423)
parRentUS <- rtruncnorm(numberrandom, a = lower1, b=upper1, mean = mean1, sd =sd1)
meanparRentUS <- mean(parRentUS)
meanparRentUS
```

```
## [1] 91.12458
```

```r
## sequence f values between lower and upper bounds
xgx <- seq(from=lower1, to=upper1, by=0.1)
#xgx
## vector of values of the heigth of the probability distribution for each value of x
ygx=dtruncnorm(xgx, a=lower1, b=upper1,mean=mean1,sd=sd1)
#ygx
## plot density
plot(xgx, ygx, type = "l", xlab ="rentUS",ylab = "",
     main="Truncated Normal Density of parRentUS")
```




```
# plot histogram
hist(parRentUS, breaks=60, main="", xlab="rentUS - random numbers from a truncated normal")
```




```
png("imgrentUS.png")
hist(parRentUS, breaks=60, main="", xlab="rentUS - random numbers from a truncated normal")
```

4.2 sqmemployee (sqm of office space per employee)

4.2.1 Germany (sqmemployeeDE)

Henger et al. (2017)

m^2 Nutzfläche je Beschäftigte[a]

Branchen	Kreistyp[b]			
	A-Städte	B-Städte	sonst. Großstädte	Durchschnitt
Banken/Finanz	31	24,6	23	24,1
Bau,Immobilien	32,3	25,6	24	25,1
EDV	28,4	22,5	21,1	22
Industrie	25,3	20,1	18,8	19,7
Transport	26,2	20,8	19,5	20,4
Dienstleistung	33,5	26,6	24,9	26
Handel	34,9	27,7	25,9	27,1
Durchschnitt	30,9	24,5	22,9	24

[a] see Henger et al. (2017)

[b] A-Städte (wichtigste Bürostandorte): Berlin, Düsseldorf, Frankfurt (Main), Hamburg, Köln, München, Stuttgart. B-Städte: u.a. Bonn, Dortmund, Dresden, Essen, Hannover, Leipzig, Mannheim, Wiesbaden. sonst. Großstädte: weitere kreisfreie Städte > 100000 Einwohner

Table 9: Office space per employee in German cities

```
# random numbers for office space per employee
set.seed(555)
sqmemployeeDE <- runif(numberrandom, min = 18.8, max = 34.9)
mean1 <- mean(sqmemployeeDE)
mean1
```

```
## [1] 26.83006
```

4.2.2 U.S. (sqmemployeeUS)

U.S. (statista} 138 square ft. (12.8 square meter) in the Americas in 2017, in 2012 it was about 176 square feet according to CoreNetGlobal (cited in Cipolla, 2020). CoreNet Global expected decline to 100 square ft (cited in PR Newswire, 2012)

The real eastate firm Austin Tenant Advisors guess that 150-175 square ft. per employee is US average (Nathan, 2020) .

100-300 sqf (9.29-27.87 sqm) suggested by Housont Installation (Twardowski, 2019)

Trend in coworking - strong increase.

We use a uniform distribution over 12-15 sqm

```
# random numbers for office space per employee
set.seed(555)
sqmemployeeUS <- runif(numberrandom, min = 9.29, max = 27.87)
mean1 <- mean(sqmemployeeUS)
mean1
```

```
## [1] 18.55699
```

5. Other Components: Delay Costs

- delayA (delay costs with contract A)

6. Non-data driven Parameters

- delayB (delay costs with contract B - not from data)
- beta (productivity of in-vehicle work - not from data)
- s_drive (share of driving during in-vehicle work - not from data)

```
#  Generate GDX-File only from data calculated in R
idvar <- c(1:10000)
# matrix of random data - database of random data
numb = 10000;
idlist <- list(name='id_r',uels=list(idvar),ts='MC run',type='set')
mc_gkm <- list(name='mc_gkm',val=as.array(pargkmDE), uels=list(idvar),dim=1,
               form='full',ts='gross monetary avg. travel costs employee [EUR:km]',
            type='parameter',domains='id_r')
mc_gs <- list(name='mc_gs',val=as.array(pargsDE), uels=list(idvar),dim=1,
              form='full',ts='gross monetary avg. travel costs employee [EUR:km]',
               type='parameter',domains='id_r')
mc_gd <- list(name='mc_gd',val=as.array(pargdDE), uels=list(idvar),dim=1,
              form='full',ts='gross monetary avg. travel costs employee [EUR:km]',
               type='parameter',domains='id_r')
mc_xbar <- list(name='mc_xbar',val=as.array(distanceDE), uels=list(idvar),dim=1,
                form='full',ts='gross monetary avg. travel costs employee [EUR:km]',
```

```
               type='parameter',domains='id_r')
mc_parambDE <- list(name='mc_parambDE',val=as.array(parambDE), uels=list(idvar),dim=1,
                    form='full',ts='parameter to deterine tkm [h:km]',
                    type='parameter',domains='id_r')
mc_sqmemployee <- list(name='mc_sqmemployee',val=as.array(sqmemployeeDE), uels=list(idvar),dim=1,form='
               type='parameter',domains='id_r')
mc_rmonth <- list(name='mc_rmonth',val=as.array(parRentDE), uels=list(idvar),dim=1,
                  form='full',ts='gross monetary avg. travel costs employee [EUR:km]',
               type='parameter',domains='id_r')
mc_avgproduct <- list(name='mc_avgproduct',val=as.array(avgprodDE), uels=list(idvar),
                      dim=1,form='full',ts='gross monetary avg. travel costs employee [EUR:km]',
               type='parameter',domains='id_r')
mc_wage <- list(name='mc_wage',val=as.array(wageDE), uels=list(idvar),dim=1,form='full',
                ts='gross monetary avg. travel costs employee [EUR:km]',
               type='parameter',domains='id_r')
wgdx.lst('MC_paramDE',list(idlist,mc_gkm,mc_gs,mc_gd,mc_xbar,mc_parambDE,
                           mc_sqmemployee,mc_rmonth,mc_avgproduct,mc_wage))
gdxInfo('MC_paramDE',dump=FALSE,returnList=TRUE, returnDF=TRUE)
```

```
## Cannot return both a list and a data frame: setting returnList FALSE
```

```
## $gdxLibraryVer
## [1] "GDX Library      34.3.0 rac355f3 Released Feb 25, 2021 WEI x86 64bit/MS Window"
##
## $gdxFileVer
## [1] "GDX Library      34.3.0 rac355f3 Released Feb 25, 2021 WEI x86 64bit/MS Window"
##
## $producer
## [1] "GDXRRW:wgdx"
##
## $symCount
## [1] 10
##
## $uelCount
## [1] 10000
##
## $sets
##   name index dim  card   text doms domnames
## 1 id_r     1   1 10000 MC run    0        *
##
## $parameters
##             name index dim  card
## 1         mc_gkm     2   1 10000
## 2          mc_gs     3   1 10000
## 3          mc_gd     4   1 10000
## 4        mc_xbar     5   1 10000
## 5    mc_parambDE     6   1 10000
## 6 mc_sqmemployee     7   1 10000
## 7      mc_rmonth     8   1 10000
## 8  mc_avgproduct     9   1 10000
## 9        mc_wage    10   1 10000
##                                                 text doms domnames
## 1 gross monetary avg. travel costs employee [EUR:km]    0     id_r
## 2 gross monetary avg. travel costs employee [EUR:km]    0     id_r
```

```
## 3 gross monetary avg. travel costs employee [EUR:km]    0     id_r
## 4 gross monetary avg. travel costs employee [EUR:km]    0     id_r
## 5                   parameter to deterine tkm [h:km]    0     id_r
## 6 gross monetary avg. travel costs employee [EUR:km]    0     id_r
## 7 gross monetary avg. travel costs employee [EUR:km]    0     id_r
## 8 gross monetary avg. travel costs employee [EUR:km]    0     id_r
## 9 gross monetary avg. travel costs employee [EUR:km]    0     id_r
##
## $variables
## [1] name     index    dim      card     text     doms     domnames
## <0 Zeilen> (oder row.names mit Länge 0)
##
## $equations
## [1] name     index    dim      card     text     doms     domnames
## <0 Zeilen> (oder row.names mit Länge 0)
##
## $aliases
## [1] name  index base
## <0 Zeilen> (oder row.names mit Länge 0)
```

```
# means
id_l = 1
idmvar = id_l
am_wage <- list(name='am_wage',val=as.array(mean(wageDE)), uels=list(idmvar),
                dim=1,form='full',ts='mean gross monetary avg. travel costs employee [EUR:km]',
               type='parameter',domains='id_l')
mwage <- mean(wageDE)*8
```

```
idvar <- c(1:10000)
# matrix of random data - database of random data
#  Generate GDX-File only from data calculated in R
idvar <- c(1:10000)
# matrix of random data - database of random data
numb = 10000;
idlist <- list(name='id_r',uels=list(idvar),ts='MC run',type='set')
mc_gkmUS <- list(name='mc_gkmUS',val=as.array(pargkmUS), uels=list(idvar),dim=1,
                 form='full',ts='gross monetary avg. travel costs employee [EUR:km]',
            type='parameter',domains='id_r')
mc_gsUS <- list(name='mc_gsUS',val=as.array(pargsUS), uels=list(idvar),dim=1,
                form='full',ts='gross monetary avg. travel costs employee [EUR:km]',
               type='parameter',domains='id_r')
mc_gdUS <- list(name='mc_gdUS',val=as.array(pargdUS), uels=list(idvar),dim=1,
                form='full',ts='gross monetary avg. travel costs employee [EUR:km]',
               type='parameter',domains='id_r')
mc_xbarUS <- list(name='mc_xbarUS',val=as.array(distanceUS), uels=list(idvar),dim=1,
                  form='full',ts='gross monetary avg. travel costs employee [EUR:km]',
               type='parameter',domains='id_r')
mc_parambUS <- list(name='mc_parambUS',val=as.array(timekmUS), uels=list(idvar),dim=1,
                    form='full',ts='parameter to determine tkm [h/km]',
               type='parameter',domains='id_r')
mc_sqmemployeeUS <- list(name='mc_sqmemployeeUS',val=as.array(sqmemployeeUS), uels=list(idvar),dim=1,fo
               type='parameter',domains='id_r')
mc_rmonthUS <- list(name='mc_rmonthUS',val=as.array(parRentUS), uels=list(idvar),dim=1,
                    form='full',ts='gross monetary avg. travel costs employee [EUR:km]',
               type='parameter',domains='id_r')
```

```
mc_avgproductUS <- list(name='mc_avgproductUS',val=as.array(avgprodUS), uels=list(idvar),dim=1,form='ful
               type='parameter',domains='id_r')
mc_wageUS <- list(name='mc_wageUS',val=as.array(wageUS), uels=list(idvar),dim=1,
                  form='full',ts='gross monetary avg. travel costs employee [EUR:km]',
               type='parameter',domains='id_r')
wgdx.lst('MC_paramUS',list(idlist,mc_gkmUS,mc_gsUS,mc_gdUS,mc_xbarUS,mc_parambUS,
                           mc_sqmemployeeUS,mc_rmonthUS,mc_avgproductUS,mc_wageUS))
gdxInfo('MC_paramUS',dump=FALSE,returnList=FALSE, returnDF=TRUE)
```

```
## $gdxLibraryVer
## [1] "GDX Library      34.3.0 rac355f3 Released Feb 25, 2021 WEI x86 64bit/MS Window"
##
## $gdxFileVer
## [1] "GDX Library      34.3.0 rac355f3 Released Feb 25, 2021 WEI x86 64bit/MS Window"
##
## $producer
## [1] "GDXRRW:wgdx"
##
## $symCount
## [1] 10
##
## $uelCount
## [1] 10000
##
## $sets
##   name index dim  card   text doms domnames
## 1 id_r     1   1 10000 MC run    0        *
##
## $parameters
##               name index dim  card
## 1         mc_gkmUS     2   1 10000
## 2          mc_gsUS     3   1 10000
## 3          mc_gdUS     4   1 10000
## 4        mc_xbarUS     5   1 10000
## 5      mc_parambUS     6   1 10000
## 6 mc_sqmemployeeUS     7   1 10000
## 7      mc_rmonthUS     8   1 10000
## 8  mc_avgproductUS     9   1 10000
## 9        mc_wageUS    10   1 10000
##                                                 text doms domnames
## 1 gross monetary avg. travel costs employee [EUR:km]    0     id_r
## 2 gross monetary avg. travel costs employee [EUR:km]    0     id_r
## 3 gross monetary avg. travel costs employee [EUR:km]    0     id_r
## 4 gross monetary avg. travel costs employee [EUR:km]    0     id_r
## 5                  parameter to determine tkm [h/km]    0     id_r
## 6 gross monetary avg. travel costs employee [EUR:km]    0     id_r
## 7 gross monetary avg. travel costs employee [EUR:km]    0     id_r
## 8 gross monetary avg. travel costs employee [EUR:km]    0     id_r
## 9 gross monetary avg. travel costs employee [EUR:km]    0     id_r
##
## $variables
## [1] name     index    dim      card     text     doms     domnames
## <0 Zeilen> (oder row.names mit Länge 0)
##
```

```
## $equations
## [1] name     index    dim      card     text     doms     domnames
## <0 Zeilen> (oder row.names mit Länge 0)
##
## $aliases
## [1] name  index base
## <0 Zeilen> (oder row.names mit Länge 0)
```

```
mwage <- mean(wageUS)*8
```

```
# summary statistics
# generate database
paramDE.frame <- data.frame(pargkmDE,pargsDE,pargdDE,distanceDE,
                            sqmemployeeDE,parRentDE,wageDE,timekmDE,avgprodDE)
paramUS.frame <- data.frame(pargkmUS,pargsUS,pargdUS,distanceUS,
                            sqmemployeeUS,parRentUS,wageUS,timekmUS,avgprodUS)
#install.packages("stargazer")
library(stargazer)
```

```
##
## Please cite as:
##  Hlavac, Marek (2018). stargazer: Well-Formatted Regression and Summary Statistics Tables.
##  R package version 5.2.2. https://CRAN.R-project.org/package=stargazer
```

```
correlation.matrix <- cor(paramDE.frame[,c('pargkmDE','pargsDE','pargdDE','distanceDE',
                          'sqmemployeeDE','parRentDE','wageDE','timekmDE','avgprodDE')])
stargazer(correlation.matrix, title="Correlation Matrix DE")
```

% Table created by stargazer v.5.2.2 by Marek Hlavac, Harvard University. E-mail: hlavac at fas.harvard.edu
% Date and time: Do, Mrz 03, 2022 - 18:42:01

Table 10: Correlation Matrix DE

	pargkmDE	pargsDE	pargdDE	distanceDE	sqmemployeeDE	parRentDE	wageDE	t
pargkmDE	1	-0.004	-0.006	-0.021	0.012	0.004	0.008	
pargsDE	-0.004	1	0.001	0.012	-0.013	0.003	-0.003	
pargdDE	-0.006	0.001	1	-0.013	-0.004	0.001	0.007	
distanceDE	-0.021	0.012	-0.013	1	-0.006	0.009	-0.0004	
sqmemployeeDE	0.012	-0.013	-0.004	-0.006	1	-0.001	-0.005	
parRentDE	0.004	0.003	0.001	0.009	-0.001	1	0.007	
wageDE	0.008	-0.003	0.007	-0.0004	-0.005	0.007	1	
timekmDE	-0.006	-0.007	-0.004	-0.297	0.007	-0.007	-0.007	
avgprodDE								

```
correlation.matrix <- cor(paramUS.frame[,c('pargkmUS','pargsUS','pargdUS','distanceUS',
                          'sqmemployeeUS','parRentUS','wageUS','timekmUS','avgprodUS')])
stargazer(correlation.matrix, title="Correlation Matrix US")
```

% Table created by stargazer v.5.2.2 by Marek Hlavac, Harvard University. E-mail: hlavac at fas.harvard.edu
% Date and time: Do, Mrz 03, 2022 - 18:42:01

References

A.A.A., 2020. Your Driving Costs 2020.

Table 11: Correlation Matrix US

	pargkmUS	pargsUS	pargdUS	distanceUS	sqmemployeeUS	parRentUS	wageUS	tim
pargkmUS	1	-0.004	0.001	0.006	0.012	0.010	0.002	-
pargsUS	-0.004	1	-0.004	-0.005	-0.013	0.005	0.009	-
pargdUS	0.001	-0.004	1	-0.016	0.002	-0.007	-0.012	-
distanceUS	0.006	-0.005	-0.016	1	-0.011	-0.010	0.005	-
sqmemployeeUS	0.012	-0.013	0.002	-0.011	1	-0.008	0.0003	-
parRentUS	0.010	0.005	-0.007	-0.010	-0.008	1	0.007	-
wageUS	0.002	0.009	-0.012	0.005	0.0003	0.007	1	-
timekmUS	-0.002	-0.001	-0.003	-0.268	-0.001	-0.008	-0.012	
avgprodUS								

ADAC (2020a) from "Kostenvergleich e-Fahrzeuge + Plug-in Hybride gegen Benziner und Diesel", 2020. ADAC2020a

ADAC, 2020b. Berechnungsgrundlagen für die standardisierte Autokostenberechnung.

BNP Paribas Real Estate, 2021. Property Report: Büromarkt Deutschland BNP2021

Bureau of Transportation Statistiscs (BTS), 2017. National Household Travel Survey 'Stretch Commute' Quick Facts. Quick Facts 2017 accessed, 23.03.2021

Bureau of Transportation Statistics (BTS), 2003. Quick Facts. Quick Facts 2003 accessed, 23.03.2021

Burd, C., Burrows, M., McKenzie, B., 2021. Travel Time to Work in the United States: 2019. American Community Survey Reports, U.S. Census Bureau.

Cipolla J, 2020. How much office space per employee do you need?%Cipolla 2020

Compostella, J., Fulton, L.M., De Kleine, R., Kim, H.C., Wallington, T.J., 2020. Near- (2020) and long-term (2030/2035) costs of automated, electrified, and shared mobility in the United States. Transport Policy 85, 54–66.

Colliers, 2019. Q4 2019 Office Market Outlook. U.S. Research Outlook.

Dauth W, Haller P, 2018. Berufliches Pendeln zwischen Wohn- und Arbeitsort: Klarer Trend zu längeren Pendeldistanzen. iab (10/2018), 13.

Deloitte, 2019. Urbane Mobilität und autonomes Fahren im Jahr 2035 – Welche Veränderungen durch Robotaxis auf Automobilhersteller, Städte, und Politik zurollen.

DeStatis, 2021. Earnings. Minimum wages. DeStatis minimum wage accessed 24.03.2021

DeStatis, 2018. Verdienststrukturerhebung – Ergebnisse für Deutschland. Fachserie 16, Heft 1 2018. Destatis FS16.1 2018 accessed 24.03.2021

DeStatis, 2017. Verdienste auf einen Blick. DeStatis Verdienste accessed 24.03.2021

Dingel, J.I., Neiman, B., 2021. How many jobs can be done from home? NBER working paper 26948.

)] Feld LP, Schulten A, Simons H, Wandzik C, Gerling M, 2020. Frühjahrsgutachten der Immobilienwirtschaft 2020 des Rates der Immobilienweisen. Im Auftrag von ZIA, Berlin. ZIA2020 accessed 24.03.2021

Gerike, R., Hubrich, S., Ließke, F., Wittig, S., Wittwer, R., 2020. Tabellenbericht zum Forschungsprojket 'Mobilität in Städten – SRV 2018' in Berlin. TU Dresden. accessed 31.05.2021

Geotab, 2018. Gridlocked cities: traffic patterns revealed across 20 major U.S. cities. GEOTAB 2018 accessed 31.05.2021

Giménez-Nadal J.I., Molina J.A., Velilla J., 2020. Trends in commuting time of European workers: A cross-country analysis. IZA Working Paper DP No. 12916.

Gould, E., 2020. State of Working America Wages 2019. A story of slow, uneven, and unequal wage growth over the last 40years. Economic Policy Institute, Washington D.C. https://epi.org/183498 accessed 24.03.2021

Henger R, Scheunemann H, Barthauer M, Giesemann C, Hude M, Seipelt B, Toschka A, 2017. Büroimmobilien: Energetischer Zustand und Anreize zur Steigerung der Energieeffizienz, Tech. Rep., Deutsche Energie-Agentur GmbH.

IEA, 2020. Key World Energy Statistics 2020, p. 81.

JLL, 2021. Büromarktüberblick Big 7/4.Quartal 2020. JLL 2021 accessed 24.03.2021

JLL, 2020a. Office Outlook JLL 2020a accessed 24.03.2021

JLL, 2020a. New York Q4 2019, Quarterly Office Outlook JLL 2020b accessed 24.03.2021

JLL, 2019. Global Premium Office Rent Tracker. JLL 2019 accessed 24.03.2021

JLL, 2018. Büro-Nebenkosten leicht rückläufig – OSCAR Analyse von JLL. JLL 2018 accessed 24.03.2021

KraftStG 2002. Einzelnorm (n.d.). KraftStg 2002 accessed 24.03.2021

Maye, A., 2019. No-vacation nation, revised. CEPR

Nathan, 2020. What is the average square footage of office space per person. (Nathan 2020) accessed 25.05.2021

OECD.Stat () accessed 22.03.2021

OECD, 2020. Taxing Wages 2020.

OECD, 2018. OECD Employment Outlook 2018, OECD. OECD 2018 accessed 24.03.2021

PR Newswire, 2012. Office space per worker will drop to 100 square feet of below for many companies within five years, according to new research from CoreNet Global. (cited in:
rnewswire 2012: news from CoreNet Global) accessed 25.05.2021

Rietveld, P., Zwart, B., van Wee, B., van den Hoorn, T., 1999. On the relationship between travel time and travel distance of commuters. Annals of Regional Science 33, 269–287.

Statista (2021a) O average, how long is your daily commute to work/school/university (one way?) Statista 2021a accessed 24.03.2021

Statista (2021b) Durchschnittsmiete in den Top 5-Bürozentren in Deutschland im 4.Quartal der Jahre 2014 und 2015. Statista 2021b accessed 24.03.2021

Statista (2021c) Mietpreise für Büroflächen in Frankfurt am Main bis 2020 Statista 2021c accessed 25.03.2021

Statista (2021d) Entwicklung der Durchschnittsmieten für Büroflächen in München von 2004 bis 2019 Statista 2021d accessed 25.03.2021

Statista (2021e) Entwicklung der Durchschnittsmieten für Büroflächen in Berlin Statista 2021e accessed 25.03.2021

Statista (2021f) Entwicklung der Spitzenmiete für Büroflächen in Berlin von 2008 bis 2020 Statista 2021f accessed 25.03.2021

Statista (2021g) Entwicklung der Mietpreise für Büroflächen in Hamburg von 2003 bis 2020 Statista 2021g accessed 25.03.2021

Statista (2021h) Entwicklung der Durchschnittsmiete für Büroflächen in Düsseldorf bis 2020 Statista 2021h accessed 25.03.2021

Statista (2021i) Entwicklung der Spitzenmiete für Büroflächen in Düsseldorf von 2009 bis zum 1. Halbjahr 2020 Statista 2021i accessed 25.03.2021

Statista (2021j) Entwicklung der Mietpreise für Büroflächen in Köln von 2003 bis 2020 Statista 2021j accessed 25.03.2021

Statista (2021k) Entwicklung der Durchschnittsmiete für Büroflächen in Stuttgart von 2001 bis 2019 Statista 2021k accessed 25.03.2021

Statista (2021l) Entwicklung der Spitzenmiete für Büroflächen in Stuttgart von 2001 bis 2019 Statista 2021l accessed 25.03.2021

Statista (2021m) Büroflächen – Bestand nach Großestädten in Deutschland 2019/2020 Statista2021m.com accessed 25.03.2021

Statistia (2019) Durchschnittliche Geschwindkeiten im Automobilverkehr in ausgewählten deutschen Städten im Jahre 2018. Statista 2019 accessed 27.05.2021

StromStG 2020. StromStG 2020 accessed 24.03.2021

Twardowski T, 2019. How much office space do you need? Twardowski 2019 accessed: 25.05.2021.

U.S. Bureau of Labor Statistics, BoL, 2020.
BoL 2020 accessed: 22.03.2021.

U.S. Bureau of Labor Statistics, BoL, 2020.

U.S. Department of Labor (US DoL), 2021. DoL accessed 24.03.2021

U.S. Department of Transportation, BoTS, 2020. Transportation Statistics Annual Report 2020. U.S. BoTS 2020 accessed: 20.03.2021.

CBRE, 2019. 2019 Global prime office occupancy costs. and \href{http://cbre.vo.llnwd.net/grgservices/secure/2019%20CBRE%20POOC_Q9bx.pdf?e=1617101862&h=d62ccb3196eeda4e0b1adeb756c3d94d}{CBRE 2019} accessed, 30.03.2021