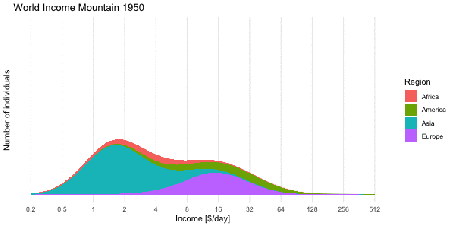
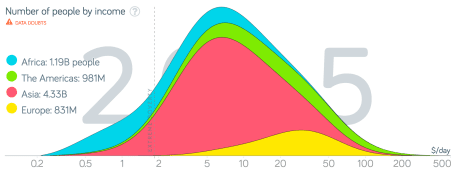
**Abstract:**

We work out the math behind the so called income mountain plots used in the book "Factfulness" by Hans Rosling and use these insight to generate such plots using tidyverse code. The trip includes a mixture of log-normals, the density transformation theorem, histogram vs. density and then skipping all those details again to make nice moving mountain plots.



**Introduction**

I got interested in the so called [**income mountain plots**](https://www.gapminder.org/data/documentation/income-mountains-dataset/), which show how incomes are distributed within individuals of a population:



Screenshot of the 2010 income mountain plot. Free material from [www.gapminder.org](https://www.gapminder.org/).

One notices that the "mountains" are plotted on a log-base-2 x-axis and without a y-axis annotation. Why? Furthermore, world income data usually involve mean income per country, so I got curious how/if these plots were made without access to finer granularity level data? Aim of this blog post is to answer these questions by using Gapminder data freely available from their webpage. The answer ended up as a nice tidyverse exercise and could serve as motivating application for basic probability course content.

**Data Munging Gapminder**

Data on income, population and Gini coefficient were needed to analyse the above formulated questions. We start by downloading the GDP data, which is the annual gross domestic product per capita by Purchasing Power Parities (PPP) measured in [international dollars](https://en.wikipedia.org/wiki/Geary%E2%80%93Khamis_dollar), fixed 2011 prices. Hence, the inflation over the years and differences in the cost of living between countries is accounted for and can thus be compared .– We download the [data from Gapminder](https://github.com/Gapminder-Indicators/gdppc_cppp/raw/master/gdppc_cppp-by-gapminder.xlsx) where they are available in *wide format* as Excel-file. For tidyverse handling we reshape them into *long format*.

##Download gdp data from gapminder - available under a CC BY-4 license.

if (!file.exists(file.path(fullFigPath, "gapminder-gdp.xlsx"))) {

download.file("https://github.com/Gapminder-Indicators/gdppc\_cppp/raw/master/gdppc\_cppp-by-gapminder.xlsx", destfile=file.path(fullFigPath,"gapminder-gdp.xlsx"))

}

gdp\_long <- readxl::read\_xlsx(file.path(fullFigPath, "gapminder-gdp.xlsx"), sheet=2) %>%

rename(country=`geo.name`) %>%

select(-geo,-indicator,-indicator.name) %>%

gather(key="year", value="gdp", -country,) %>%

filter(!is.na(gdp))

Furthermore, we rescale GDP per year to daily income, because this is the unit used in the book.

gdp\_long %<>% mutate(gdp = gdp / 365.25)

Code Chunks –

```{r,include=FALSE,echo=FALSE,message=FALSE}

##If default fig.path, then set it.

if (knitr::opts\_chunk$get("fig.path") == "figure/") {

knitr::opts\_knit$set( base.dir = '/Users/hoehle/Sandbox/Blog/')

knitr::opts\_chunk$set(fig.path="figure/source/2018-07-02-factfulness/")

}

fullFigPath <- paste0(knitr::opts\_knit$get("base.dir"),knitr::opts\_chunk$get("fig.path"))

filePath <- file.path("","Users","hoehle","Sandbox", "Blog", "figure", "source", "2018-07-02-factfuless")

knitr::opts\_chunk$set(echo = TRUE,fig.width=8,fig.height=4,fig.cap='',fig.align='center',echo=FALSE,dpi=72\*2) # autodep=TRUE

options(width=150)

suppressPackageStartupMessages(library(tidyverse))

suppressPackageStartupMessages(library(magrittr))

suppressPackageStartupMessages(library(knitr))

## Packages used for this post

suppressPackageStartupMessages(require(reldist))

suppressPackageStartupMessages(require(openxlsx))

suppressPackageStartupMessages(require(gridExtra))

suppressPackageStartupMessages(require(grid))

suppressPackageStartupMessages(require(scales))

suppressPackageStartupMessages(require(animation))

##Configuration

options(knitr.table.format = "html")

theme\_set(theme\_minimal())

#if there are more than n rows in the tibble, print only the first m rows.

options(tibble.print\_max = 10, tibble.print\_min = 5)

```

## Abstract:

We work out the math behind the so called income mountain plots used in the book "Factfulness" by Hans Rosling and use these insight to generate such plots using

tidyverse code. The trip includes a mixture of log-normals, the density transformation theorem, histogram vs. density and then skipping all those details again to make nice moving mountain plots.

<center>

```{r,results='asis',echo=FALSE,fig.cap="Animated Income Mountains"}

cat(paste0("![]({{ site.baseurl }}/",knitr::opts\_chunk$get("fig.path"),"moving-mountains.gif"),")")

```

</center>

{% include license.html %}

## Introduction

What is really enjoyable is that the

[Gapminder web page](https://www.gapminder.org) allows you to study

many of the graphs from the book interactively and contains the data for download. Being a fan of \*\*transparency\*\* and \*\*reproducibility\*\*, I got interested in the so called

[\*\*income mountain plots\*\*](https://www.gapminder.org/data/documentation/income-mountains-dataset/), which show

how incomes are distributed within individuals of a population:

<p>

<center>

```{r,results='asis',echo=FALSE,fig.cap="Income mountain from Gapminder."}

cat(paste0("![]({{ site.baseurl }}/",knitr::opts\_chunk$get("fig.path"),"gapminder-income-mountain.png"),")")

```

</center>

<FONT COLOR="bbbbbb">Screenshot of the 2010 income mountain plot. Free material from [www.gapminder.org](https://www.gapminder.org).</FONT>

<p>

One notices that the "mountains" are plotted on a log-base-2 x-axis and without a y-axis annotation. Why? Furthermore, world income data usually involve mean income per country, so I got curious how/if these plots were made without access to finer granularity level data? Aim of this blog post is to answer these questions by using Gapminder data freely available from their webpage. The answer ended up as a nice `tidyverse` exercise and could serve as motivating application for basic probability course content.

## Data Munging Gapminder

Data on income, population and Gini coefficient were needed to analyse

the above formulated questions. I have done this previously in order to visualize the [Olympic Medal Table Gapminder Style](http://staff.math.su.se/hoehle/blog/2016/08/21/gapMedal.html). We start by downloading the GDP data, which is the annual gross domestic product per capita by Purchasing Power Parities (PPP) measured in

[international dollars](https://en.wikipedia.org/wiki/Geary–Khamis\_dollar), fixed 2011 prices. Hence, the inflation over

the years and differences in the cost of living between countries is

accounted for and can thus be compared - see the [Gapminder documentation](https://www.gapminder.org/data/documentation/gd001/) for further details. We download the

[data from Gapminder](https://github.com/Gapminder-Indicators/gdppc\_cppp/raw/master/gdppc\_cppp-by-gapminder.xlsx) where they are available in \*wide format\* as Excel-file. For tidyverse handling we reshape them into \*long format\*.

```{r DATA\_GDPLOAD, echo=TRUE, message=FALSE}

##Download gdp data from gapminder - available under a CC BY-4 license.

if (!file.exists(file.path(fullFigPath, "gapminder-gdp.xlsx"))) {

download.file("https://github.com/Gapminder-Indicators/gdppc\_cppp/raw/master/gdppc\_cppp-by-gapminder.xlsx", destfile=file.path(fullFigPath,"gapminder-gdp.xlsx"))

}

gdp\_long <- readxl::read\_xlsx(file.path(fullFigPath, "gapminder-gdp.xlsx"), sheet=2) %>%

rename(country=`geo.name`) %>%

select(-geo,-indicator,-indicator.name) %>%

gather(key="year", value="gdp", -country,) %>%

filter(!is.na(gdp))

```

Furthermore, we rescale GDP per year to daily income, because this is the unit used in the book.

```{r, echo=TRUE}

gdp\_long %<>% mutate(gdp = gdp / 365.25)

```

Similar code segments are written for (see the [code](`r paste0("https://raw.githubusercontent.com/hoehleatsu/hoehleatsu.github.io/master/\_source/",current\_input())`) on github for details)

\* the gini (`gini\_long`) and population (`pop\_long`) data

\* the regional group (=continent) each country belongs two (`group`)

```{r DATA\_LOAD, echo=FALSE, results='hide', message=FALSE}

if (!file.exists(file.path(fullFigPath, "gapminder-gini-v2.xlsx"))) {

download.file("https://docs.google.com/spreadsheets/d/1V9ueokiba2KKO0Un8UwJ73rBPr2Zub8j7bfT6Fi222E/export?format=xlsx", destfile=file.path(fullFigPath,"gapminder-gini-v2.xlsx"))

}

gini <- readxl::read\_xlsx(file.path(fullFigPath, "gapminder-gini-v2.xlsx"), sheet=2) %>%

rename(country=`Row Labels`) %>% select(-`Geo code`, -indicator) %>% as.tbl()

gini\_long <- gini %>%

gather(key="year", value="gini", -country) %>%

mutate(year=as.character(as.numeric(year)))

if (!file.exists(file.path(fullFigPath,"gapminder-pop.csv"))) {

download.file("https://docs.google.com/spreadsheet/pub?key=phAwcNAVuyj0XOoBL\_n5tAQ&output=csv", destfile=file.path(fullFigPath, "gapminder-pop.csv"))

}

pop <- readr::read\_csv(file=file.path(fullFigPath, "gapminder-pop.csv")) %>%

rename(country=`Total population`)

pop\_long <- pop %>% gather(key="year", value="population", -country) %>%

filter(!is.na(population))

##Regions, see https://www.gapminder.org/gsdev

if (!file.exists(file.path(fullFigPath,"gapminder-countrygroups.xls"))) {

download.file("https://www.gapminder.org/gsdev/files/popReg/en/list\_country\_groups\_en.xls", destfile=file.path(fullFigPath, "gapminder-countrygroups.xls"))

}

groups <- readxl::read\_xls(file.path(fullFigPath, "gapminder-countrygroups.xls"), sheet=1) %>%

rename(country=Entity, region=Region, code=ID)

```

The four data sources are then joined into one long tibble `gm`. For each year we also compute the fraction a country's population makes up of the world population that year (column `w`) as well as the fraction within the year and region the population makes up (column `w\_region`) :

```{r DATA\_MERGE}

##Merge data together

gm <- inner\_join(gini\_long, gdp\_long, by=c("country","year")) %>%

inner\_join(pop\_long, by=c("country","year")) %>%

inner\_join(groups, by=c("country")) %>%

select(country, region, code, everything())

##Add weights (by year) column and by year and region

gm %<>% group\_by(year) %>%

mutate(w=population/sum(population)) %>%

group\_by(year, region) %>%

mutate(w\_region = population/sum(population)) %>%

ungroup

gm

```

## Income Mountain Plots

The construction of the income mountain plots is thoroughly described

on the

[Gapminder webpage](https://www.gapminder.org/data/documentation/income-mountains-dataset/), but without mathematical detail. With

respect to the math it says: \*"Bas van Leeuwen shared his formulas

with us and explained how to the math from ginis and mean income, to

accumulated distribution shapes on a logarithmic scale."\*

Unfortunately, the formulas are not shared with the reader. It's not

black magic though: The income distribution of a country is assumed to

be [log-normal](https://en.wikipedia.org/wiki/Log-normal\_distribution) with a given mean $\mu$ and standard deviation $\sigma$

on the log-scale, i.e. $X \sim \operatorname{LogN}(\mu,\sigma^2)$.

From knowing the mean income

$\overline{x}$ of the distribution as well as the Gini index $G$ of the distribution, one can show that it's possible to directly infer $(\mu, \sigma)$ of the log-normal distribution.

Because the

[Gini index](https://en.wikipedia.org/wiki/Gini\_coefficient) of the log-normal distribution is given by

$$

G = 2\Phi\left(\frac{\sigma}{\sqrt{2}}\right)-1,

$$

where $\Phi$ denotes the CDF of the

standard normal distribution, and by knowing that the expectation of the log-normal is $E(X) = \exp(\mu + \frac{1}{2}\sigma^2)$, it is possible to determine $(\mu,\sigma)$ as:

$$

\sigma = \sqrt{2}\> \Phi^{-1}\left(\frac{G+1}{2}\right)

\quad\text{and}\quad

\mu = \log(\overline{x}) - \frac{1}{2} \sigma^2.

$$

We can use this to determine the parameters of the log-normal for every country in each year.

```{r XBARGINI2MUSIGMA}

## Compute the parameters of the log-normal distribution from a specified

## mean and gini-index.

xbarg\_2\_musigma <- function(xbar,g) {

sigma <- sqrt(2)\*qnorm((g+1)/2)

mu <- log(xbar) - 1/2\*sigma^2

return(c(mu=mu, sigma=sigma))

}

##Determine parameters of the log-normal modelling the gdp per country and year

gm <- gm %>% group\_by(country,year) %>% do({

##Determine parameters of the log-normal distrib

theta <- xbarg\_2\_musigma(xbar=.$gdp, g=.$gini)

data.frame(., meanlog=theta[1], sdlog=theta[2], median=exp(theta[1]))

}) %>% ungroup

```

```{r}

n\_country <- length(unique(gm$country))

```

### Mixture distribution

The income distribution of a \*\*set of countries\*\* is now given as a

[Mixture distribution](https://en.wikipedia.org/wiki/Mixture\_distribution) of log-normals, i.e. one component for each of the countries in the set with a weight proportional to the population of the country. As an example, the world income distribution would be a mixture of the `r n\_country` countries in the Gapminder dataset, i.e.

$$

f\_{\text{mix}}(x) = \sum\_{i=1}^{`r n\_country`} w\_i \>\cdot

\>f\_{\operatorname{LogN}}(x; \mu\_i, \sigma\_i^2), \quad\text{where}

\quad w\_i = \frac{\text{population}\_i}{\sum\_{j=1}^{`r n\_country`} \text{population}\_j},

$$

and $f\_{\operatorname{LogN}}(x; \mu\_i, \sigma\_i^2)$ is the density of the log-normal distribution with country specific parameters. Note that we could have equally used the mixture approach to define the income of, e.g., a continent region. With the above definition we define standard R-functions for computing the PDF

(`dmix`), CDF (`pmix`), quantile function (`qmix`) and a function for sampling from the distribution (`rmix`) - see the [github code](`r paste0("https://raw.githubusercontent.com/hoehleatsu/hoehleatsu.github.io/master/\_source/",current\_input())`) for details.

```{r MIXFUNCS, echo=FALSE}

#' Generate a sample of size n from the mixture distribution.

#'

#' @param x Value where to evaluate the density (can be a vector)

#' @param meanLog Vector containing the individual meanlog parameters of each component

#' @param sdLog Vector containing the individual sdlog parameters of each component

#' @param Weight vector, should be the same length as meanlog and sum to one.

#' @return A vector of length n.

rmix <- function(n, meanlog, sdlog, w) {

##Sanity checks

stopifnot( (length(meanlog) == length(sdlog)) & (length(meanlog) == length(w)) )

##Sample component and then from its density

i <- sample(seq\_len(length(meanlog)), size=n, prob=w, replace=TRUE)

rlnorm(n, meanlog=meanlog[i], sdlog=sdlog[i])

}

#' Density of the log-normal mixture

dmix <- function(x, meanlog, sdlog, w) {

##Sanity check

stopifnot( (length(meanlog) == length(sdlog)) & (length(meanlog) == length(w)) )

one\_x <- function(x) sum(w\*dlnorm(x, meanlog=meanlog, sdlog=sdlog))

sapply(x, one\_x)

}

#' Cumulative density of the log-normal mixture

pmix <- function(q, meanlog, sdlog, w) {

##Sanity check

stopifnot( (length(meanlog) == length(sdlog)) & (length(meanlog) == length(w)) )

one\_q <- function(q) sum(w\*plnorm(q, meanlog=meanlog, sdlog=sdlog))

sapply(q, one\_q)

}

##Quantile function of the log-normal mixture

qmix <- function(p, meanlog, sdlog, w) {

##Sanity check

stopifnot( (length(meanlog) == length(sdlog)) & (length(meanlog) == length(w)) )

##Find one quantile numerically

one\_p <- function(p) {

target <- function(x) {

pmix(x, meanlog=meanlog, sdlog=sdlog, w=w) - p

}

uniroot(target,lower=0, upper=1e99)$root

}

sapply(p, one\_p)

}

```

```{r GMRECENT, results='hide'}

##Restrict to year 2015

gm\_recent <- gm %>% filter(year == 2015) %>% ungroup

```

We use the mixture approach to compute the density of the world income distribution obtained by "mixing" all `r n\_country`

log-normal distributions. This is shown below for the World income distribution of

the year `r mean(as.numeric(gm\_recent$year), na.rm=TRUE)`. Note the $\log\_2$ x-axis. This presentation

is \*Factfulness\*' preferred way of illustrating the skew income distribution.

```{r DMIXWORLD, echo=TRUE}

<<GMRECENT>>

##Make a data frame containing the densities of each region for

##the gm\_recent dataset

df\_pdf <- data.frame(log2x=seq(-2,9,by=0.05)) %>%

mutate(x=2^log2x)

pdf\_region <- gm\_recent %>% group\_by(region) %>% do({

pdf <- dmix(df\_pdf$x, meanlog=.$meanlog, sdlog=.$sdlog, w=.$w\_region)

data.frame(x=df\_pdf$x, pdf=pdf, w=sum(.$w), population=sum(.$population), w\_pdf = pdf\*sum(.$w))

})

## Total is the sum over all regions - note the summation is done on

## the original income scale and NOT the log\_2 scale. However, one can show that in the special case the result on the log-base-2-scale is the same as summing the individual log-base-2 transformed densities (see hidden CHECKMIXTUREPROPERTIES chunk).

pdf\_total <- pdf\_region %>% group\_by(x) %>%

summarise(region="Total",w=sum(w), pdf = sum(w\_pdf))

## Expectation of the distribution

mean\_mix <- gm\_recent %>%

summarise(mean=sum(w \* exp(meanlog + 1/2\*sdlog^2))) %$% mean

## Median of the distribution

median\_mix <- qmix(0.5, gm\_recent$meanlog, gm\_recent$sdlog, gm\_recent$w)

## Mode of the distribution on the log2-scale (not transformation invariant!)

mode\_mix <- pdf\_total %>%

mutate(pdf\_log2x = log(2) \* x \* pdf) %>%

filter(pdf\_log2x == max(pdf\_log2x)) %$% x

```

```{r CHECKMIXTUREPROPERTIES, results='hide'}

## Sanity check - this should be approximately 1!

pdf\_total2 <- pdf\_total %>%

mutate(xm1 = if\_else(is.na(lag(x)),0, lag(x)), width=x-xm1,

# Check that computation in one go is the same as adding the

# densities of the continent components on the original scale

pdf\_direct=dmix(x, meanlog=gm\_recent$meanlog, sdlog=gm\_recent$sdlog, w=gm\_recent$w),

pdf\_the\_same = isTRUE(all.equal(pdf, pdf\_direct)))

##This should be close to one (might need to expand the grid to get closer to one)

sum(pdf\_total2$pdf \* pdf\_total2$width)

##This should be true all along the way

stopifnot(all(pdf\_total2$pdf\_the\_same))

## Check summation property on the y-axis instead

pdf\_total\_log2x <- pdf\_total2 %>%

mutate(xm1 = if\_else(xm1==0,1e-2, xm1)) %>%

mutate(log2x = log2(x), log2xm1 = log2(xm1), width=log2x - log2xm1, pdf = log(2) \* x \* pdf)

##This should be close to one, depenends on what is used for xm1 =0.

sum(pdf\_total\_log2x$pdf \* pdf\_total\_log2x$width)

############################################################

## Investigate how big the difference is, if we sum the transformed densities on the log\_2 scale opposite to summing

## the untransformed densities onthe original scale.

## Conclusion: Doesn't make a difference as can be seen from the change of variable formula stated above:

##

## f\_Y(y) = 2^y log(2) f\_mix(2^y) =

## = 2^y log(2) \sum\_{i} w\_i \* f\_i(2^y)

## = \sum\_{i} w\_i \* (2^y log(2) \* f\_i(2^y))

## = \sum\_{i} w\_i f\_{Y,i}(y)

##

## because f\_{Y,i}(y) = (2^y log(2) \* f\_i(2^y)).

###

############################################################

df\_total <- pdf\_region %>% group\_by(x) %>%

summarise(region="Total",w=sum(w),

pdfx\_sum\_dens\_on\_x = sum(w\_pdf),

pdflog2\_sum\_dens\_on\_log2 = sum(log(2)\*x\*w\_pdf)) %>%

mutate(pdflog2\_sum\_dens\_on\_x = log(2) \* x \* pdfx\_sum\_dens\_on\_x,

isTheSame = isTRUE(all.equal(pdflog2\_sum\_dens\_on\_log2,pdflog2\_sum\_dens\_on\_x)))

all(df\_total$isTheSame)

```

```{r DENSITYPLOTS}

p1 <- ggplot(pdf\_region %>% rename(Region = region), aes(x=x, y=log(2)\*x\*pdf, color=Region)) + geom\_line() +

scale\_x\_continuous(trans='log2',

breaks = trans\_breaks("log2", function(x) 2^x),

labels = trans\_format("log2", math\_format(2^.x))) +

xlab("Income [$/day]") +

ylab(expression(log(2) %.% x %.% pdf)) +

NULL

##Weighted + summed

p2 <- ggplot(pdf\_region %>% rename(Region=region), aes(x=x, y=log(2)\*x\*w\_pdf, color=Region)) +

geom\_line(data=pdf\_total, aes(x=x, y=log(2)\*x\*pdf), color="steelblue", lwd=1.5) +

geom\_line() +

scale\_x\_continuous(trans='log2',

breaks = trans\_breaks("log2", function(x) 2^x,n=11),

labels = trans\_format("log2", function(x) ifelse(x<0, sprintf("%.1f",2^x), sprintf("%.0f",2^x)))) +

xlab("Income [$/day]") +

ylab(expression(w[region] %.% log(2) %.% x %.% pdf)) +

guides(color=FALSE) +

NULL

grid.arrange(p1,p2, ncol=2)

```

For illustration we compute a mixture distribution for each region using all countries within region. This is shown in the left pane. Note: because a log-base-2-transformation is used for the x-axis, we need to perform a [change of variables](https://en.wikipedia.org/wiki/Probability\_density\_function#Dependent\_variables\_and\_change\_of\_variables), i.e. we compute the density for $Y=\log\_2(X)=g(X)$ where $X\sim f\_{\text{mix}}$, i.e.

$$

f\_Y(y) = \left| \frac{d}{dy}(g^{-1}(y)) \right| f\_X(g^{-1}(y)) = \log(2) \cdot 2^y \cdot f\_{\text{mix}}( 2^y) = \log(2) \cdot x \cdot f\_{\text{mix}}(x), \text{ where } x=2^y.

$$

In the right pane we then show the region specific densities each weighted by their population fraction. These are then summed up to yield the world income shown as a thick blue line.

The median of the resulting world income distribution is at `r sprintf("%.1f",median\_mix)` \$/day, whereas the mean of the mixture is at an income of `r sprintf("%.1f",mean\_mix)`\$/day and the mode (on the log-base-2 scale) is `r sprintf("%.1f",mode\_mix)`\$/day. Note that the later is not transformation invariant, i.e. the value is not the mode of the income distribution, but of $\log\_2(X)$.

To get the income mountain plots as shown in \*Factfulness\*, we additionally need to obtain number of people on the $y$-axis and not density. We do this by partitioning the x-axis into non-overlapping intervals and then compute the number of individuals expected to fall into a given interval with limits $[l, u]$. Under our model this expectation is

$$n \cdot (F\_{\text{mix}}(u)-F\_{\text{mix}}(l)),$$

where $F\_{\text{mix}}$ is the CDF of the mixture

distribution and $n$ is the total world population. The mountain plot below shows this for a given partition with $n=`r format(sum(gm\_recent$population), big.mark=",")`$.

Note that $2.5\cdot

10^8$ corresponds to 250 mio people. Also note the $\log\_2$ x-axis, and hence (on the linear scale) unequally wide intervals of the partitioning. Contrary to \*Factfulness\*', I prefer to make this more explicit by indicating the intervals explicitly on the x-axis of the mountain plot, because it is about number of people in certain \*\*income brackets\*\*.

```{r PREPAREMOUNTAINDF, echo=TRUE}

##Function to prepare the data.frame to be used in a mountain plot

make\_mountain\_df <- function(gm\_df, log2x=seq(-2,9,by=0.25)) {

##Make a data.frame containing the intervals with appropriate annotation

df <- data.frame(log2x=log2x) %>%

mutate(x=2^log2x) %>%

mutate(xm1 = lag(x), log2xm1=lag(log2x)) %>%

mutate(xm1=if\_else(is.na(xm1),0,xm1),

log2xm1=if\_else(is.na(log2xm1),0,log2xm1),

mid\_log2 = (log2x+log2xm1)/2,

width = (x-xm1),

width\_log2 = (log2x-log2xm1)) %>%

##Format the interval character representation

mutate(interval=if\_else(xm1<2, sprintf("[%6.1f-%6.1f]",xm1,x), sprintf("[%4.0f-%4.0f]",xm1,x)),

interval\_log2x=sprintf("[2^(%4.1f)-2^(%4.1f)]",log2xm1,log2x))

##Compute expected number of individuals in each bin.

people <- gm\_df %>% group\_by(region) %>% do({

countries <- .

temp <- df %>% slice(-1) %>% rowwise %>%

mutate(

prob\_mass = diff(pmix(c(xm1,x), meanlog=countries$meanlog, sdlog=countries$sdlog, w=countries$w\_region)),

people = prob\_mass \* sum(countries$population)

)

temp %>% mutate(year = max(gm\_df$year))

})

##Done

return(people)

}

##Create mountain plot data set for gm\_recent with default spacing.

(people <- make\_mountain\_df(gm\_recent))

```

This can then be plotted with `ggplot2`:

```{r TRUEMOUNTAINPLOT, echo=FALSE, fig.height=5}

ggplot(people %>% rename(Region=region),

aes(x=interval,y=people, fill=Region)) + geom\_col() +

geom\_col(width=1) +

ylab("Number of individuals") + xlab("Income [$/day]") +

theme\_minimal() +

theme(axis.text.x = element\_text(angle = 90, hjust = 1))

```

In light of all the talk about gaps, it can also be healthy to plot the income distribution on the linear scale. From this it becomes obvious that linearly there indeed are larger absolute differences in income, but -as argued in the book- the exp-scale (base 2) incorporates peoples perception about the worth of additional income.

```{r LINEARSCALE-MOUNTAINPLOT, warning=FALSE}

ggplot(people %>% rename(Region=region),

aes(x=xm1,y=people, fill=Region, width=width)) +

geom\_col() +

ylab("Number of individuals") + xlab("Income [$/day]") +

NULL

```

Because the intervals are not equally wide, only the height of the bars should be interpreted in this plot. However, the eye perceives area, which in this case is misguiding. Showing histograms with unequal bin widths is a constant dilemma between area, height, density and perception. The recommendation would be that if one wants to use the linear-scale, then one should use equal width linear intervals or directly plot the density. As a consequence, plots like the above are not recommended, but they make obvious the tail behaviour of the income distribution - a feature which is somewhat hidden by the log-base-2-scale plots.

```{r LINEARDMIX, eval=FALSE}

##Weighted + summed

ggplot(pdf\_region %>% rename(Region=region), aes(x=x, y=w\_pdf, color=Region)) +

geom\_line(data=pdf\_total, aes(x=x, y=pdf), color="steelblue", lwd=1.5) +

geom\_line() +

xlab("Income [$/day]") +

ylab(expression(w[region] %.% pdf)) +

#guides(color=FALSE) +

NULL

```

Of course none of the above plots looks as nice as the Gapminder plots, but they have proper x and y-axes annotation and, IMHO, are clearer to interpret, because they do not mix the concept of density with the concept of individuals falling into income bins. As the bin-width converges to zero, one gets the density multiplied by $n$, but this complication of infinitesimal width bins is impossible to communicate. In the end this was the talent of Hans Rosling and Gapminder - to make the complicated easy and intuitive! We honor this by skipping the math^[Shown is the expected number of individuals in thin bins of size 0.01 on the log-base-2-scale. As done in Factfulness we also skip the interval annotation on the x-axis and, as a consequence, do without y-axis tick marks, which would require one to explain the interval widths.] and celebrate the result as the \*\*art\*\* it is!

```{r ARTISTICMOUNTAINPLOT, echo=TRUE, fig.height=4, results="hide"}

##Make mountain plot with smaller intervals than in previous plot.

ggplot\_oneyear\_mountain <- function(people, ymax=NA) {

##Make the ggplot

p <- ggplot(people %>% rename(Region=region), aes(x=mid\_log2,y=people, fill=Region)) +

geom\_col(width=min(people$width\_log2)) +

ylab("Number of individuals") +

xlab("Income [$/day]") +

scale\_x\_continuous(minor\_breaks = NULL, trans="identity",

breaks = trans\_breaks("identity", function(x) x,n=11),

labels = trans\_format(trans="identity", format=function(x) ifelse(x<0, sprintf("%.1f",2^x), sprintf("%.0f",2^x)))) +

theme(axis.text.y=element\_blank(), axis.ticks.y=element\_blank()) +

scale\_y\_continuous(minor\_breaks = NULL, breaks = NULL, limits=c(0,ymax)) +

ggtitle(paste0("World Income Mountain ",max(people$year))) +

NULL

#Show it and return it.

print(p)

invisible(p)

}

##Create the mountain plot for 2015

gm\_recent %>%

make\_mountain\_df(log2x=seq(-2,9,by=0.01)) %>%

ggplot\_oneyear\_mountain()

```

## Discussion

Our replicated mountain plots do not exactly match those made by Gapminder (c.f. the screenshot). It appears as if our distributions are located slightly more to the right. It is not entirely clear why there is a deviation, but one possible problem could be that we do the translation into income per day differently? I'm not an econometrician, so this could be a trivial blunder on my side, however, the values in this post are roughly of the same magnitude as the graph on p. 45 in @vanzanden\_etal2011 mentioned in the [Gapminder documentation page](https://www.gapminder.org/data/documentation/income-mountains-dataset/), whereas the Gapminder curves appear too far to the left. It might be worthwhile to check [individual country data](https://docs.google.com/spreadsheets/d/1939CzZ5HHoLreb0YyopaWfNjJ9mnN27IhywI6-TuwZs/edit#gid=501532268) underlying the graphs to see where the difference is. <br>

\*\*Edit 2018-07-05:\*\* I checked this and read the [documentation](https://www.gapminder.org/data/documentation/income-mountains-dataset/) again more carefully: Apparently, Gapminder uses \*mean household income (or consumption) per person per day (measured in PPP$ 2011)\* opposite to the \*GDP/capita\* used in the scientific literature they quote and for which you can download the data from their website. To me this was not clear when reading Factfulness and, unfortunately, there is no documentation for how exactly their mean household income value per individual is computed from the GDP/capita.^[The Gapminder [Household Income v1](http://www.gapm.io/ioihhinc) description is currently (as of 2018-07-05) blank and so is the link specified as reference [Gapminder [3]](gapm.io/elev) in \*Factfulness\*. The detailed data just contain the column `household\_income` without further explanation. Altogether, a somewhat disappointing number of links in the book are currently still under construction. Reading the document [Data Sources used in Don’t Panic — End Poverty](https://www.gapminder.org/news/data-sources-dont-panic-end-poverty/) it appears to me that the GDP/capita are converted to household incomes by scaling all countries GDPs per capita until the global income log-normal mixture distribution is such that \*11.3% of world population are below the extreme poverty line of 1.85$/day (in PPP 2011) in year 2015\*. When I tried this rather ad-hoc approach I got a scale parameter of approximately 0.379 for the GDP, which corresponds to a shift of 1.402 to the left on the log-base-2 scale. This worked ok for the single benchmark of Sweden in 1970 that I tested. Furthermore, Gapminder uses something they call \*\*log-normal-topping\*\* per country in order to get better tail behaviour. [Adventurous Excel-files](https://drive.google.com/drive/folders/11\_k8\_sTa7ycuprJjaORotbGVQyX1b\_tx) not directly linked to in any explanation are used for the calculation and can be consulted for further details. The authors note themselves that they hope to convert these computations to python soon...`r emo::ji("smile")`] For the sake of illustrating the dynamics in the world income the difference in scale is not that important, though.

We end the post by animating the dynamics of the income mountains since 1950 using `gganimate`. To put it in possibilistic terms:

Let the [world move forward](https://youtu.be/hVimVzgtD6w?t=8m12s)! It is not as bad as it seems. Facts matter.

```{r ANIMATE, cache=TRUE, results='hide', message=FALSE}

##Make mountain plot with smaller intervals than in previous plot.

people\_all <- gm %>% group\_by(year) %>% do({

make\_mountain\_df(., log2x=seq(-2,9,by=0.01))

})

##Helper function to do the animation for a set of years.

animation <- function(years) {

##Largest bin to show

total <- people\_all %>%

group\_by(year,as.factor(mid\_log2)) %>%

summarise(people=sum(people))

ymax <- max(total %$% people)

##Show all on the same y-scale

for (theYear in years) {

people\_all %>% filter(year == theYear) %>%

ggplot\_oneyear\_mountain(ymax=ymax)

}

invisible()

}

##Years to show

years <- gm %>% filter(year >= 1950) %>% distinct(year) %$% year

ani.options(interval=0.2)

##Make an animate

animation::saveGIF(animation(years),

movie.name=file.path(fullFigPath, "moving-mountains.gif"),

ani.width=600,ani.height=300)

```

<center>

```{r,results='asis',echo=FALSE,fig.cap="Animated Income Mountains"}

cat(paste0("![]({{ site.baseurl }}/",knitr::opts\_chunk$get("fig.path"),"moving-mountains.gif"),")")

```

</center>

```{r FINDSCALEFACTOR, eval=FALSE}

##Determine parameters of the log-normal modelling the gdp per country and year

f <- function(scale, gm\_df, target=0.113, poverty\_limit=1.85) {

##Scale all GDP in a very ad-hoc approach to get a more fitting proportion

##below poverty limit.

foo <- gm\_df %>% mutate(gdp=gdp \* scale)

foo <- foo %>% select(-meanlog,-sdlog) %>% group\_by(country,year) %>% do({

##Determine parameters of the log-normal distrib

theta <- xbarg\_2\_musigma(xbar=.$gdp, g=.$gini)

data.frame(., meanlog=theta[1], sdlog=theta[2], median=exp(theta[1]))

})

pmix(poverty\_limit, meanlog=foo$meanlog, sdlog=foo$sdlog, w=foo$w) - target

}

scale <- uniroot(f, gm\_df=gm %>% filter(year==2015), interval=c(0.001,0.999))$root

scale

sprintf("%.3f",scale)

sprintf("%.3f",log2(scale))

scale \* 22054/365.25

log2(scale)

```

```{r STORERESULTS}

save(list=c("n\_country", "gm", "rmix", "pmix", "dmix", "qmix"), file=file.path(fullFigPath, "factfulness.RData"))

```

## Literature

* the gini (gini\_long) and population (pop\_long) data
* the regional group (=continent) each country belongs two (group)

The four data sources are then joined into one long tibble gm. For each year we also compute the fraction a country’s population makes up of the world population that year (column w) as well as the fraction within the year and region the population makes up (column w\_region) :

## # A tibble: 15,552 x 9

## country region code year gini gdp population w w\_region

##

## 1 Afghanistan Asia AFG 1800 0.305 1.65 3280000 0.00347 0.00518

## 2 Albania Europe ALB 1800 0.389 1.83 410445 0.000434 0.00192

## 3 Algeria Africa DZA 1800 0.562 1.96 2503218 0.00264 0.0342

## 4 Andorra Europe AND 1800 0.4 3.28 2654 0.00000280 0.0000124

## 5 Angola Africa AGO 1800 0.477 1.69 1567028 0.00166 0.0214

## # ... with 1.555e+04 more rows

**Income Mountain Plots**

The construction of the income mountain plots is thoroughly described on the [Gapminder webpage](https://www.gapminder.org/data/documentation/income-mountains-dataset/), but without mathematical detail. With respect to the math it says: *"Bas van Leeuwen shared his formulas with us and explained how to the math from ginis and mean income, to accumulated distribution shapes on a logarithmic scale."* Unfortunately, the formulas are not shared with the reader. It’s not black magic though: The income distribution of a country is assumed to be [log-normal](https://en.wikipedia.org/wiki/Log-normal_distribution) with a given mean \(\mu\) and standard deviation \(\sigma\) on the log-scale, i.e. \(X \sim \operatorname{LogN}(\mu,\sigma^2)\). From knowing the mean income \(\overline{x}\) of the distribution as well as the Gini index \(G\) of the distribution, one can show that it’s possible to directly infer \((\mu, \sigma)\) of the log-normal distribution.

Because the [Gini index](https://en.wikipedia.org/wiki/Gini_coefficient) of the log-normal distribution is given by \[  
G = 2\Phi\left(\frac{\sigma}{\sqrt{2}}\right)-1,  
\] where \(\Phi\) denotes the CDF of the standard normal distribution, and by knowing that the expectation of the log-normal is \(E(X) = \exp(\mu + \frac{1}{2}\sigma^2)\), it is possible to determine \((\mu,\sigma)\) as:

\[  
\sigma = \frac{2}{\sqrt{2}} \Phi^{-1}\left(\frac{G+1}{2}\right)  
\quad\text{and}\quad  
\mu = \log(\overline{x}) – \frac{1}{2} \sigma^2.  
\]

We can use this to determine the parameters of the log-normal for every country in each year.

**Mixture distribution**

The income distribution of a **set of countries** is now given as a [Mixture distribution](https://en.wikipedia.org/wiki/Mixture_distribution) of log-normals, i.e. one component for each of the countries in the set with a weight proportional to the population of the country. As an example, the world income distribution would be a mixture of the 192 countries in the Gapminder dataset, i.e.

\[  
f\_{\text{mix}}(x) = \sum\_{i=1}^{192} w\_i \>\cdot  
\>f\_{\operatorname{LogN}}(x; \mu\_i, \sigma\_i^2), \quad\text{where}  
\quad w\_i = \frac{\text{population}\_i}{\sum\_{j=1}^{192} \text{population}\_j},  
\] and \(f\_{\operatorname{LogN}}(x; \mu\_i, \sigma\_i^2)\) is the density of the log-normal distribution with country specific parameters. Note that we could have equally used the mixture approach to define the income of, e.g., a continent region. With the above definition we define standard R-functions for computing the PDF (dmix), CDF (pmix), quantile function (qmix) and a function for sampling from the distribution (rmix) .

We use the mixture approach to compute the density of the world income distribution obtained by "mixing" all 192 log-normal distributions. This is shown below for the World income distribution of the year 2015. Note the \(\log\_2\) x-axis. This presentation is *Factfulness*‘ preferred way of illustrating the skew income distribution.

##Restrict to year 2015

gm\_recent <- gm %>% filter(year == 2015) %>% ungroup

##Make a data frame containing the densities of each region for

##the gm\_recent dataset

df\_pdf <- data.frame(log2x=seq(-2,9,by=0.05)) %>%

mutate(x=2^log2x)

pdf\_region <- gm\_recent %>% group\_by(region) %>% do({

pdf <- dmix(df\_pdf$x, meanlog=.$meanlog, sdlog=.$sdlog, w=.$w\_region)

data.frame(x=df\_pdf$x, pdf=pdf, w=sum(.$w), population=sum(.$population), w\_pdf = pdf\*sum(.$w))

})

## Total is the sum over all regions - note the summation is done on

## the original income scale and NOT the log\_2 scale. However, one can show that in the special case the result on the log-base-2-scale is the same as summing the individual log-base-2 transformed densities (see hidden CHECKMIXTUREPROPERTIES chunk).

pdf\_total <- pdf\_region %>% group\_by(x) %>%

summarise(region="Total",w=sum(w), pdf = sum(w\_pdf))

## Expectation of the distribution

mean\_mix <- gm\_recent %>%

summarise(mean=sum(w \* exp(meanlog + 1/2\*sdlog^2))) %$% mean

## Median of the distribution

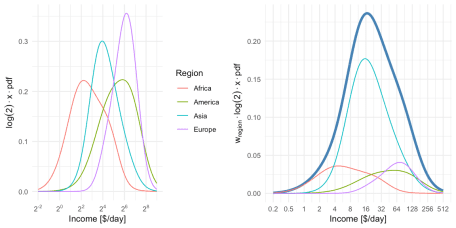
median\_mix <- qmix(0.5, gm\_recent$meanlog, gm\_recent$sdlog, gm\_recent$w)

## Mode of the distribution on the log2-scale (not transformation invariant!)

mode\_mix <- pdf\_total %>%

mutate(pdf\_log2x = log(2) \* x \* pdf) %>%

filter(pdf\_log2x == max(pdf\_log2x)) %$% x



For illustration we compute a mixture distribution for each region using all countries within region. This is shown in the left pane. Note: because a log-base-2-transformation is used for the x-axis, we need to perform a [change of variables](https://en.wikipedia.org/wiki/Probability_density_function#Dependent_variables_and_change_of_variables), i.e. we compute the density for \(Y=\log\_2(X)=g(X)\) where \(X\sim f\_{\text{mix}}\), i.e. \[  
f\_Y(y) = \left| \frac{d}{dy}(g^{-1}(y)) \right| f\_X(g^{-1}(y)) = \log(2) \cdot 2^y \cdot f\_{\text{mix}}( 2^y) = \log(2) \cdot x \cdot f\_{\text{mix}}(x), \text{ where } x=2^y.  
\]

In the right pane we then show the region specific densities each weighted by their population fraction. These are then summed up to yield the world income shown as a thick blue line. The median of the resulting world income distribution is at 20.0 $/day, whereas the mean of the mixture is at an income of 39.9$/day and the mode (on the log-base-2 scale) is 17.1$/day. Note that the later is not transformation invariant, i.e. the value is not the mode of the income distribution, but of \(\log\_2(X)\).

To get the income mountain plots as shown in *Factfulness*, we additionally need to obtain number of people on the \(y\)-axis and not density. We do this by partitioning the x-axis into non-overlapping intervals and then compute the number of individuals expected to fall into a given interval with limits \([l, u]\). Under our model this expectation is

\[n \cdot (F\_{\text{mix}}(u)-F\_{\text{mix}}(l)),\]

where \(F\_{\text{mix}}\) is the CDF of the mixture distribution and \(n\) is the total world population. The mountain plot below shows this for a given partition with \(n=7,305,116,647\). Note that \(2.5\cdot 10^8\) corresponds to 250 mio people. Also note the \(\log\_2\) x-axis, and hence (on the linear scale) unequally wide intervals of the partitioning. Contrary to *Factfulness*‘, I prefer to make this more explicit by indicating the intervals explicitly on the x-axis of the mountain plot, because it is about number of people in certain **income brackets**.

##Function to prepare the data.frame to be used in a mountain plot

make\_mountain\_df <- function(gm\_df, log2x=seq(-2,9,by=0.25)) {

##Make a data.frame containing the intervals with appropriate annotation

df <- data.frame(log2x=log2x) %>%

mutate(x=2^log2x) %>%

mutate(xm1 = lag(x), log2xm1=lag(log2x)) %>%

mutate(xm1=if\_else(is.na(xm1),0,xm1),

log2xm1=if\_else(is.na(log2xm1),0,log2xm1),

mid\_log2 = (log2x+log2xm1)/2,

width = (x-xm1),

width\_log2 = (log2x-log2xm1)) %>%

##Format the interval character representation

mutate(interval=if\_else(xm1<2, sprintf("[%6.1f-%6.1f]",xm1,x), sprintf("[%4.0f-%4.0f]",xm1,x)),

interval\_log2x=sprintf("[2^(%4.1f)-2^(%4.1f)]",log2xm1,log2x))

##Compute expected number of individuals in each bin.

people <- gm\_df %>% group\_by(region) %>% do({

countries <- .

temp <- df %>% slice(-1) %>% rowwise %>%

mutate(

prob\_mass = diff(pmix(c(xm1,x), meanlog=countries$meanlog, sdlog=countries$sdlog, w=countries$w\_region)),

people = prob\_mass \* sum(countries$population)

)

temp %>% mutate(year = max(gm\_df$year))

})

##Done

return(people)

}

##Create mountain plot data set for gm\_recent with default spacing.

(people <- make\_mountain\_df(gm\_recent))

## # A tibble: 176 x 13

## # Groups: region [4]

## region log2x x xm1 log2xm1 mid\_log2 width width\_log2 interval interval\_log2x prob\_mass people year

##

## 1 Africa -1.75 0.297 0.25 -2 -1.88 0.0473 0.25 [ 0.2- 0.3] [2^(-2.0)-2^(-1.8)] 0.00134 1586808. 2015

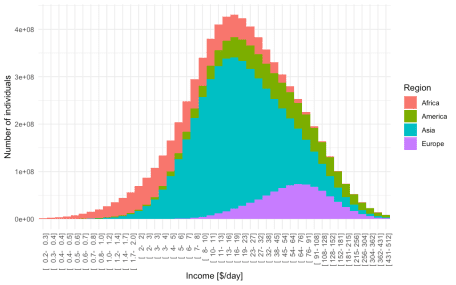
## 2 Africa -1.5 0.354 0.297 -1.75 -1.62 0.0563 0.25 [ 0.3- 0.4] [2^(-1.8)-2^(-1.5)] 0.00205 2432998. 2015

## 3 Africa -1.25 0.420 0.354 -1.5 -1.38 0.0669 0.25 [ 0.4- 0.4] [2^(-1.5)-2^(-1.2)] 0.00307 3639365. 2015

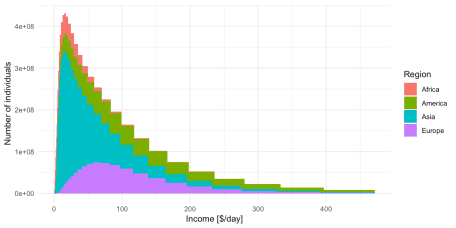
## 4 Africa -1 0.5 0.420 -1.25 -1.12 0.0796 0.25 [ 0.4- 0.5] [2^(-1.2)-2^(-1.0)] 0.00448 5305674. 2015

## 5 Africa -0.75 0.595 0.5 -1 -0.875 0.0946 0.25 [ 0.5- 0.6] [2^(-1.0)-2^(-0.8)] 0.00636 7537067. 2015

## # ... with 171 more rows

This can then be plotted with ggplot2: 

In light of all the talk about gaps, it can also be healthy to plot the income distribution on the linear scale. From this it becomes obvious that linearly there indeed are larger absolute differences in income, but -as argued in the book- the exp-scale (base 2) incorporates peoples perception about the worth of additional income.

Because the intervals are not equally wide, only the height of the bars should be interpreted in this plot. However, the eye perceives area, which in this case is misguiding. Showing histograms with unequal bin widths is a constant dilemma between area, height, density and perception. The recommendation would be that if one wants to use the linear-scale, then one should use equal width linear intervals or directly plot the density. As a consequence, plots like the above are not recommended, but they make obvious the tail behaviour of the income distribution – a feature which is somewhat hidden by the log-base-2-scale plots.

Of course none of the above plots looks as nice as the Gapminder plots, but they have proper x and y-axes annotation and, IMHO, are clearer to interpret, because they do not mix the concept of density with the concept of individuals falling into income bins. As the bin-width converges to zero, one gets the density multiplied by \(n\), but this complication of infinitesimal width bins is impossible to communicate. In the end this was the talent of Hans Rosling and Gapminder – to make the complicated easy and intuitive! We honor this by skipping the math[1](http://staff.math.su.se/hoehle/blog/2018/07/02/factfulness.html#fn1) and celebrate the result as the **art** it is!

##Make mountain plot with smaller intervals than in previous plot.

ggplot\_oneyear\_mountain <- function(people, ymax=NA) {

##Make the ggplot

p <- ggplot(people %>% rename(Region=region), aes(x=mid\_log2,y=people, fill=Region)) +

geom\_col(width=min(people$width\_log2)) +

ylab("Number of individuals") +

xlab("Income [$/day]") +

scale\_x\_continuous(minor\_breaks = NULL, trans="identity",

breaks = trans\_breaks("identity", function(x) x,n=11),

labels = trans\_format(trans="identity", format=function(x) ifelse(x<0, sprintf("%.1f",2^x), sprintf("%.0f",2^x)))) +

theme(axis.text.y=element\_blank(), axis.ticks.y=element\_blank()) +

scale\_y\_continuous(minor\_breaks = NULL, breaks = NULL, limits=c(0,ymax)) +

ggtitle(paste0("World Income Mountain ",max(people$year))) +

NULL

#Show it and return it.

print(p)

invisible(p)

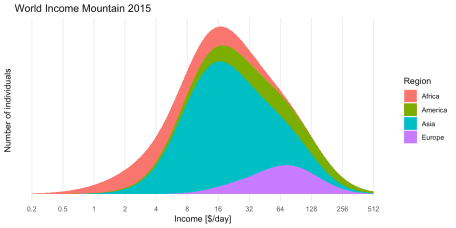
}

##Create the mountain plot for 2015

gm\_recent %>%

make\_mountain\_df(log2x=seq(-2,9,by=0.01)) %>%

ggplot\_oneyear\_mountain()



**Discussion**

Our replicated mountain plots do not exactly match those made by Gapminder (c.f. the screenshot). It appears as if our distributions are located slightly more to the right. It is not entirely clear why there is a deviation, but one possible problem could be that we do the translation into income per day differently?

