Homework 1: Introduction to R programming

Code ▾

Louis Sirugue

10/2021

This homework covers the material from the first 5 lectures. I encourage you start **working on it progressively** as we go through the course. Homeworks should be done **individually**. You are welcome to help each other by sharing what you understand, but write your homework by yourself, **do not share your code**. Contrarily to the online quizzes, this homework is meant to be challenging. **Partial answers will be taken into account**, so if you cannot do everything, **write down your thought process**.

The grading system is available here (https://louissirugue.github.io/data-analysis-course/homework1/grading.html), and the material for this homework is available here (https://louissirugue.github.io/data-analysis-course/homework1/data.zip) Make sure that your answers are unambiguous and that your code is annotated extensively. You can write either in English or in French.

I. Describe an earnings distribution

The dataset for this first part contains the sex and annual earnings of 64,336 individuals. This data was extracted from the 2020 Annual Social and Economic (ASEC) Supplement to the US Current Population Survey (CPS), conducted by the Bureau of the Census for the Bureau of Labor Statistics. Only the sex and salary of working individuals with positive earnings are kept in this extract. The full dataset is available here (https://www.census.gov/data/datasets/time-series/demo/cps/cps-asec.html) and the full documentation here (https://www2.census.gov/programs-surveys/cps/techdocs/cpsmar20.pdf).

1) Read the file asec_2020.csv and store it in tibble format.

The `tibble()` format comes with the `tidyverse` package. We first need to load this package. To open a dataset we can use the `read` function that corresponds to its format. In our case the relevant function is `read.csv()`. To change the format of an object we can use the `as` function that corresponds to its format. In our case the relevant function is `as_tibble()`

Hide

```
library(tidyverse)
asec <- as_tibble(read.csv("asec_2020.csv"))
```

2) Print the first 10 rows of the dataset. The output should document the class of each variable. How do you interpret the class associated with each of these two variables?

The `head()` function allows to print the `n` first rows of a dataset. The fact that the data is in `tibble()` format ensures that the class of each variable will be displayed under its name.

Hide

```
head(asec, n = 10)
```

```
## # A tibble: 10 x 2
##    Sex    Earnings
##    <chr>     <int>
##  1 Female    52500
##  2 Male      34000
##  3 Female    40000
##  4 Male       8424
##  5 Male      58000
##  6 Female    42000
##  7 Male      55000
##  8 Female    28000
##  9 Male        200
## 10 Male      25000
```

The variable `Sex` is a character variable, meaning that it contains text. The variable Earnings is an integer variable, meaning that is contains round numbers.

3) Use the summary() function to describe *only* the earnings variable. Write a small description of the earnings distribution based on the output.

To make sure that the summary() function only outputs a description of the variable of interest, we can subset a single variable from the data with the `$` operator.

Hide

```
summary(asec$Earnings)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##       2   25000   45000   62132   75000 1999999
```

The earnings distribution in this sample ranges from $2 to $1999999. The earnings of the middle 50% of the sample lie between $25000 and $75000. Individuals earn on average $62132, but the median individual earns $45000. This means that half of the sample earns less than this amount, and half of the sample earns more. The fact the the median is lower than the mean suggests that the earnings distribution is skewed to the right.

4) Compute the same statistics separately for males and females using the summarise() function of the dplyr grammar. Is the range (max-min) an adequate statistic to compare these two distributions?

The `summarise()` function allows to aggregate variables into a set of group-level statistics. In our case there are two groups indicated by the sex variable. We should thus group the data according to this variable before summarizing the earnings variable.

The output must include the following statistics about the group-level earnings distribution:

- Minimum: obtained with `min()`
- First quartile: second element of the output of `quantile()`
- Mean: obtained with `mean()`
- Median: obtained with `median()` or third element of the output of `quantile()`
- Third quartile: fourth element of the output of `quantile()`
- Maximum: obtained with `max()`

Hide

```
asec %>%
  group_by(Sex) %>%
  summarise(`Min.` = min(Earnings),
            `1st Qu.` = quantile(Earnings)[2],
            Median = median(Earnings),
            Mean = mean(Earnings),
            `3rd Qu.` = quantile(Earnings)[4],
            `Max.` = max(Earnings)) %>%
  kable(., caption = "Descriptive statistics by sex")
```

Descriptive statistics by sex

Sex	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
Female	2	21900	38400	50915.03	60200	1999999
Male	2	30000	52000	72527.36	86098	1269999

In this case the range is a bit misleading. It indicates that the minimum of both distributions is $2 and that while the maximum value is $1269999 for males it amounts to $1999999 for females. This is not very representative of how the two distributions compare as any other statistic is higher for males than for females. In this case the inter-quartile range allows for a more adequate comparison: the earnings of the middle 50% of the female sample range from $21900 to $60200, and those of the middle 50% of the male sample range from $30000 to $86098.

5) Superimpose the earnings distribution of males and females with different fill colors on a graph and modify the x-axis labels to express earnings in thousands of dollars. Does it look like a common distribution?

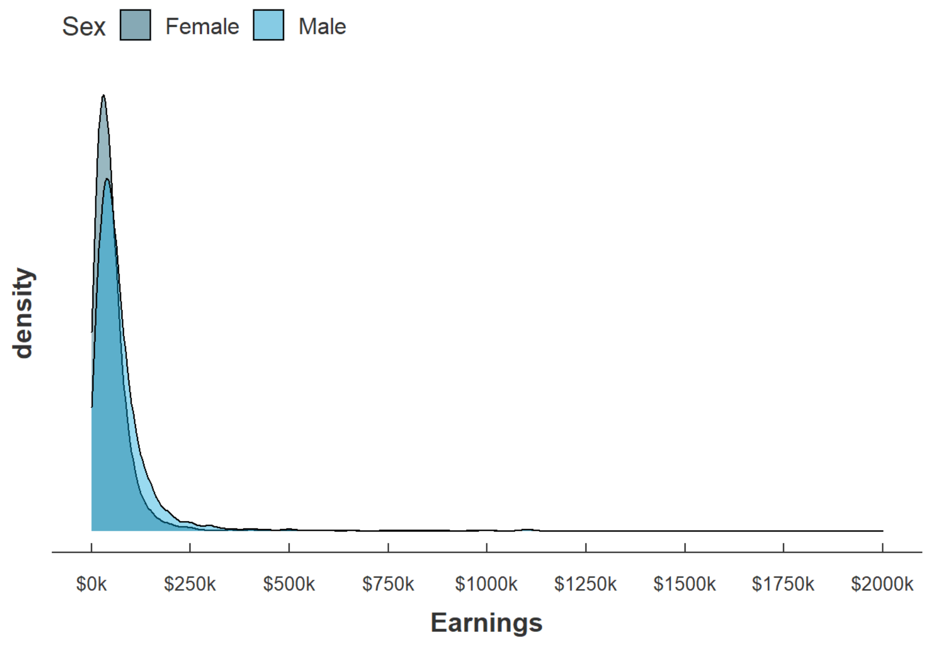
For the density of males and females to be plotted separately with different fill colors, the `sex` variable should be assigned to the `fill=` argument of `aes()`.

`geom_density()` is the geometry that corresponds to the density. As the two distributions will be superimposed, the opacity should be set with the `alpha` parameter which ranges from 0 (full transparency) to 1 (full opacity).

Modifications of an axis scale should be indicated in a `scale` function. As we want to modify the x-axis and that the x variable is continuous, the corresponding function is `scale_x_continuous()`.

Hide

```
ggplot(asec, aes(x = Earnings, fill = Sex)) +
  geom_density(alpha = .4, bw = 15000) +
  scale_x_continuous(breaks = seq(0, 2000000, 250000),
                     labels = paste0("$", seq(0, 2000, 250),"k")) +
  theme(axis.text.y = element_blank())
```



Both densities seem to follow a log-normal distribution.

II. Visualize intergenerational persistence

Intergenerational persistence refers to the association between individuals' socio-economic status across generations. A higher intergenerational persistence (or low intergenerational mobility) in income means that children are very likely to reach an income level that is similar to that of their parents. Conversely, a low intergenerational persistence means that their is no strong association between the income of parents an that their children will earn as adults.

The academic article this second part is about, *'Where is the Land of Opportunity? The Geography of Intergenerational Mobility in the United States'*, documents the characteristics of intergenerational persistence/mobility in the United States. This section focuses on a specific set of statistics gathered in *transition matrices*. For every quantile of the parent income distribution (it can be quintiles, deciles, centiles, ...), the transition matrix documents share of children falling in each quantile of their income distribution. Consider for instance the following tercile by tercile transition matrix.

3x3 Transition matrix

	Parent T1	**Parent T2**	**Parent T3**
Child T1	49.4%	30.3%	20.3%
Child T2	32.4%	36.7%	30.9%
Child T3	18.2%	33.0%	48.8%

It reads as follows. 48.8% of children born to parents in the third tercile of the income distribution also belong to the third tercile of their income distribution once adults. In other words, conditional on being born to parents in the third tercile, a child as 48.8% chances to be in the third tercile as well once adult. Note that if child income was completely disconnected from parent income, this probability would be of one third.

The dataset for this exercise is a 100 $\times$ 100 transition matrix estimated on US data for children born in the 1980-82 birth cohorts. The data and the article attached to this homework are publicly available at opportunityinsights.org (https://opportunityinsights.org/). One subtlety with this transition matrix is about the 6.1% zero-income earners in the children income distribution. Because they all do earn the same income, it does not make sense to allocate them in different quantiles. Thus, 'centile' 1 corresponds to 0.3% of negative child income, and centile 4 corresponds to the mass of children earning zero income. Consequently, the seventh bin of the child income distribution is not a full percentile group but contains $(0.07 - 0.061 - 0.003) = 0.6\%$ of children). Percentiles 8 to 100 of the child income distribution are regular percentiles. There is no issue of weakly negative income for the parent income distribution.

1) Use the package readxl to read the sheet 'Online Data Table 1' in the file transition_matrix.xls. Omit the first 9 rows of the sheet and make sure that the first row is not used for variable names. Rename the first column 'child_quantile'.

The readxl package provides functions to import excel files into R. In our case the file is a .xls file, so the dedicated function is `read_xls()`. The name of the sheet to import should be indicated in the `sheet=` argument of the function. The number of rows to omit at the top of the file should be indicated in the `skip=` argument. Whether or not to assign the values of the first row to variable names can be indicated by setting the `col_names=` argument to either `TRUE` or `FALSE`.

To access the variable names of dataset in order to change them, we can use the `colnames()` function. The indices of columns to which new variable names should be assigned must be specified within brackets `[]`.

Hide

```
library(readxl)
tmat <- read_xls("transition_matrix.xls", sheet = "Online Data Table 1",
                 skip = 9, col_names = F)
colnames(tmat)[1] <- "child_quantile"
```

2) How would you compute the expected rank for children born to parents in a given quantile? In other words, how can you get the average quantile for children whose parents belong to the nth quantile? Consider quantile as a continuous variable for this question onward.

Hint: Keep in mind that the first column lists the quantiles of the child income distribution. Every other column corresponds to a quantile of the parent income distribution, and values in these columns indicate the share of children born to parents in this quantile who attain each quantile of the child income distribution. For instance, 1.87% of children whose parents are in the 2nd quantile reach the 10th quantile. You can think of the expected rank as a weighted average of quantiles.

The expected rank of an individual conditional on being born to parents in a given quantile is equal to the sum of each possible quantile weighted by their corresponding probability.

Take as an example the 3x3 transition matrix above. The expected tercile for a child born to parents in the first tercile is given by:

$$\frac{(49.4\% \times 1) + (32.4\% \times 2) + (18.2\% \times 3)}{49.4\% + 32.4\% + 18.2\%} = \frac{(.494 \times 1) + (.324 \times 2) + (.182 \times 3)}{1} = 1.688$$

In that case, children born to parents in the first tercile would on average reach the 1.688th tercile, or, discretizing the results, would on average fall in the second quantile.

In practice, we can thus obtain the expected quantile of a child by multiplying each quantile with the corresponding probability, and summing the resulting vector.

3) Compute and plot the expected centile of children for each parent centile. Comment the graph.

Hint: Start by creating a new dataset in long format. To do so, pivot the shares of children per parent quantile in long format and recode the resulting parent quantile variable. You can check that your results are correct by comparing the graph you obtain with Figure II.A. in the article attached to this homework.

The function `pivot_longer()` allows to reshape the data from the wide format to the long format. The first argument should indicate the variables that have to be put in long format. As we need to pivot all variables except that of the child quantile, we can just write `-child_quantile`. Then we must name the variable in which the names of the variables to pivot should be stored, and the name of the variable in which the values of the variable to pivot should be stored.

Because the variable name are of the form `...n` where `n` is the column index, the long-format parent quantile variable should be recoded. As the first column is allocated to the child quantile, names of parent quantile variables range from `...2` to `...101`. We can thus start by extracting only the fourth to the last character using the `substr()` function. The index of the last character of a string can be obtained with the function `nchar()`, for *Number of CHARacters*. The operation `substr(parent_quantile, 4, nchar(parent_quantile)` thus transforms the parent quantile variable into a variable ranging from 2 to 101. The last step to recode it properly is thus to set the variable as numeric with the corresponding `as` function, and to subtract 1.

Hide

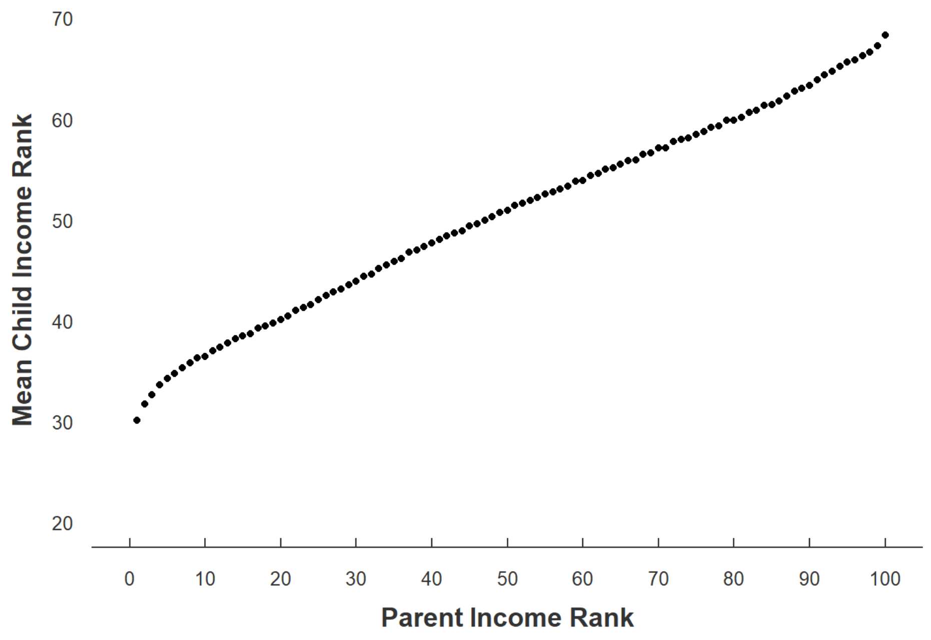
```
tmat_long <- tmat %>%
  pivot_longer(-child_quantile, names_to = "parent_quantile", values_to = "share") %>%
  mutate(
    parent_quantile = as.numeric(substr(parent_quantile, 4, nchar(parent_quantile))) - 1
    )
```

That being done, we can apply the procedure described in the previous question separately for each parent quintile: summing the product between child quantiles and their probability. Such an aggregation can be done easily with the `summarise()` function. It requires to first group the data according to variable indicating the groups for which the operation should be done separately, here parent quantile, with the `group_by()` function. Then we generate the expected quantile variable within `summarise()` to which we assign the sum of the product between child quantile and the corresponding probabilities.

The resulting data contains two variables: the parent quantile and the expected quantile for children born to parents in these quantiles. We can plot the relationship between the two variable with `ggplot()` by assigning parent quantiles to the x axis and child expected quantiles to the y axis in `aes()` and by using a the `geom_point()` geometry.

Hide

```
tmat_long %>%
  group_by(parent_quantile) %>%
  summarise(exp_quantile = sum(child_quantile * share)) %>%
  ggplot(., aes(x = parent_quantile, y = exp_quantile)) + geom_point() +
  scale_x_continuous(name = "Parent Income Rank", limits = c(0, 100),
                     breaks = seq(0, 100, 10)) +
  scale_y_continuous(name = "Mean Child Income Rank", limits = c(20, 70),
                     breaks = seq(20, 70, 10))
```



The resulting figure documents a positive relationship between parent income percentile and child expected income percentile: the higher the income percentile of one's parent, the higher her expected percentile rank. Had there been no relationship at all between the two variables, the expected income percentile for children would be the 50th whatever the parent centile rank. Had there been a perfect relationship between the two variables, the expected income centile for children would always be equal to that of their parents. This figure reveals a relationship that lies in between these two extreme theoretical cases: children income centiles are not fully determined by that of their parents, but there is a sizable intergenerational persistence.

4) Write a function that allows to reduce the 100 $\times$ 100 matrix into an q $\times$ q matrix. It should take two arguments: output_format and q. When output_format = "wide" the function should return the reduced matrix in wide format, as the original data. Otherwise it should be returned in long format. The q argument should control the number of quantiles of the reduced matrix.

Hint: Start from the data in long format that you should have computed in the previous question. Note that the ntile function is not appropriate to aggregate quantiles in that case given the allocation of negative- and zero-income earners. If your function works fine, setting output_format to "wide" and q to 5 should output the same numbers as in Table II in the article attached to this homework.

To illustrate how we should proceed, let's take a more tractable example. Consider that we want to reduce a 6 $\times$ 6 transition matrix into a 2 $\times$ 2.

6x6 Transition matrix

	1	**2**	**3**	**4**	**5**	**6**
1	a	g	m	s	B	H
2	b	h	n	t	C	I
3	c	i	o	u	D	J
4	d	j	p	v	E	K
5	e	k	q	w	F	L
6	f	l	r	A	G	M

We want to aggregate this 6 × 6 = 32-cell matrix into a 2 × 2 = 4-cell matrix. We should thus aggregate the 3 × 3 = 9-cell squares at each corner of the initial matrix. Take the top-left 3 × 3 square for instance. We should aggregate the cells (a, g, m, b, h, n, c, i, o) into a single one. But we cannot simply sum their values. Indeed, as each column sums to 100%, the three halves of column would sum to 150% on expectation, while we would like each new cell of 2 × 2 matrix to sum to 50% so that a full column sums to 100%. Thus we need to weight the sum of the cells by 50%\150%, which actually corresponds to the ratio between the new quantile dimension and the initial one, 2/6. This reasoning is the same for each 3 × 3 corner square of the initial matrix. The matrix reduction then writes as follows.

Reduced matrix

	1	**2**
1	(2/6)x(a+g+m+b+h+n+c+i+o)	(2/6)x(s+B+H+t+C+I+u+D+J)
2	(2/6)x(d+j+p+e+k+q+f+l+r)	(2/6)x(v+E+K+w+F+L+A+G+M)

More generally, the procedure to reduce the matrix goes as follows.

1. Replace the initial percentiles by the corresponding desired quantiles
2. Sum every cell belonging to any given pair of quantiles
3. Multiply everything by the ratio between the initial and the new number of quantiles.

The last steps can be done easily with the functions `group_by()` and `summarise()`. But because the `ntile()` function is not appropriate to aggregate quantiles in that case given the allocation of negative- and zero-income earners, we should find a way to do it manually. To rescale a variable that range from 1 to 100 to another variable that should range from 1 to q, we can simply divide this variable by 100 and multiply it by q. For instance, into which quintile would the 34th centile fall? 0.34 × 5 = 1.2. It would fall into the second quintile. Indeed, values from 0 to 1 should be in the first quintile, from 1 to 2 in the second quintile, and so on up to values ranging from 4 to 5 that should be in the fifth quintile. Not that because the higher bound of the last range should be included, it should be as well for the higher bound of each quantile while lower bounds should be excluded. To obtain the actual value of the quintile and not a location on a scale from q/100 to q, we should consider the first integer that comes afterwards. In our example, instead of 1.2 the result should be 2 because 1.2 falls into the second quantile. The smallest integer greater than a given value is called the ceiling of that value. We can obtain it with the function `ceiling()`. Thus, to convert a centile variable into a `q`-tile variable, we should take the ceiling of that centile variable multiplied by `q`/100.

Hide

```
q_aggreg <- function(q, centile_var) {
  return(ceiling(centile_var * (q / 100)))
}
```

We now have all the required elements to complete steps 1 to 3. Once these are completed, we should use `if {} else {}` to return the reduced matrix either in long or in wide format depending on the value of the `output_format` argument.

Hide

```
reduce_matrix <- function(output_format, q) {

  reduced_matrix <- tmat_long %>%
    mutate_at(c("child_quantile", "parent_quantile"), ~q_aggreg(q, .)) %>%
    group_by(parent_quantile, child_quantile) %>%
    summarise(share = (q / 100) * sum(share)) %>%
    ungroup()

  if (output_format == "long") {
    return(reduced_matrix)
  } else {
    return(reduced_matrix %>%
             pivot_wider(names_from = "parent_quantile", values_from = "share") %>%
             select(-child_quantile))
  }
}

kable(reduce_matrix("wide", 5),
      caption = "5x5 transition matrix",
      row.names = T) %>%
  column_spec(1, bold = T)
```

5x5 transition matrix

	1	2	3	4	5
1	0.34	0.24	0.18	0.13	0.11
2	0.28	0.24	0.20	0.16	0.12
3	0.18	0.22	0.22	0.21	0.17
4	0.12	0.18	0.22	0.24	0.24
5	0.08	0.12	0.18	0.25	0.37

5) Plot the 10 $\times$ 10 transition matrix generated by this function with the geom_tile geometry. Use a continuous color palette from the viridis package. Comment the plot.

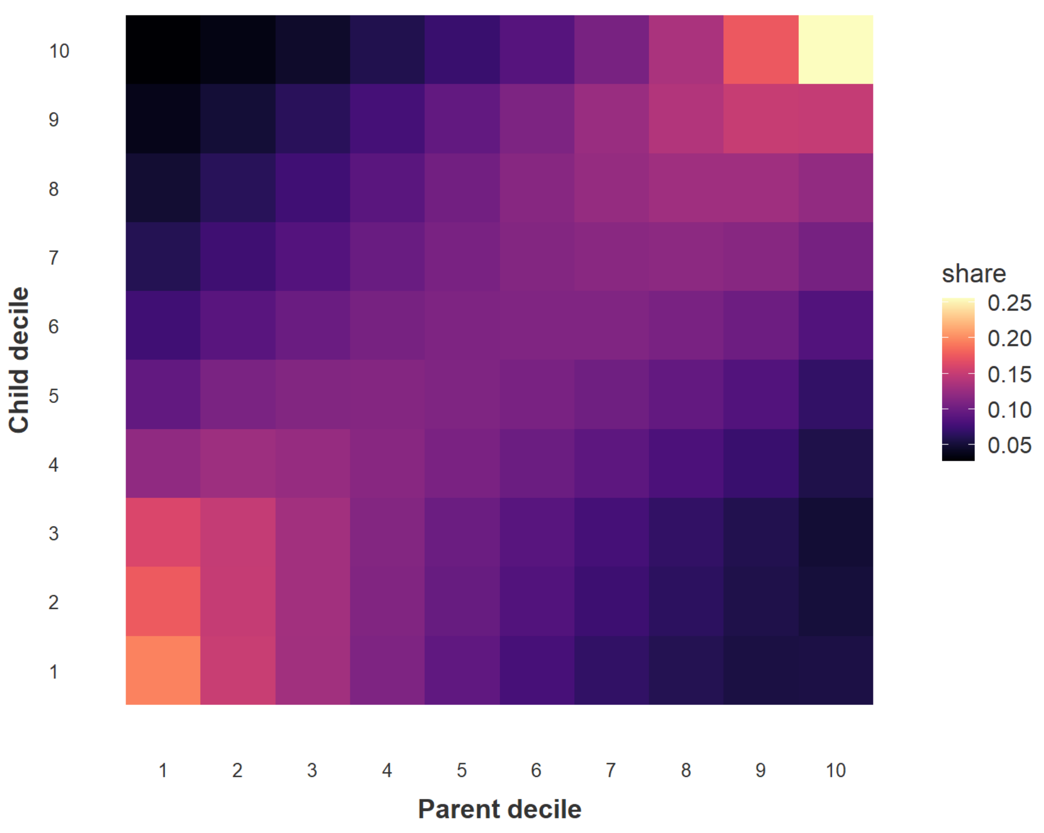
The `geom_tile()` geometry allows to plot a matrix as grid whose cells are filled with a color that depends on their value. To use `geom_tile()` the data should be in long format. To have a 10 $\times$ 10 transition matrix in long format we can use the function from the previous question with the argument `output_format` set to `"long"` and the argument `q` set to 10. We then need to assign the parent quantile variable to the x axis, the child quantile variable to the y axis and the corresponding probabilities to the `fill` argument of `aes()`.

To assign a color palette from the `viridis` package we first need to load that package. Then, the palette should be assigned we the dedicated `scale` function, `scale_fill_viridis` in our case. The color of the gradient can then be chosen with the `option` argument of this `scale` function.

Hide

```
library(viridis)

ggplot(reduce_matrix("long", 10),
       aes(x = parent_quantile, y = child_quantile, fill = share)) +
  geom_tile() + scale_fill_viridis(option = "A") +
  scale_x_continuous(name = "Parent decile", breaks = 1:10) +
  scale_y_continuous(name = "Child decile", breaks = 1:10) +
  theme(axis.ticks = element_blank(), axis.line = element_blank(),
        legend.position = "right", legend.direction = "vertical")
```



The most striking pattern from this plot is the diagonal gradient. The closer a child quantile to that of its parents, the higher the likelihood she falls into that quantile. This provides another way to visualize intergenerational persistence. Had there been no intergenerational persistence, the value of every cell would be 10% and there would be no color variation on the plot. Another interesting visual element is the fact that the vertical gradient is much more pronounced at the tails of the parent income distribution. A children born to parents in the first or tenth decile of the income distribution is much more likely to end up in the same quantile as her parents than an individual born to parents from the middle of the income distribution. While individuals born to parents in the bottom decile have 3% chance to reach the top decile, individuals born to parents in the top decile have a 26% chance to remain in the top decile.