# Praktikum 2

Importing data dan Visualisasi Data

## 1. Tujuan

1. Memahami cara menavigasi dan mengelola sistem file pada R, termasuk konsep direktori kerja, path penuh, dan path relatif.
2. Menggunakan fungsi dasar R untuk mengimpor serta mengelola file agar analisis data dapat dilakukan secara portabel dan terstruktur.
3. Menerapkan visualisasi data menggunakan **ggplot2** dan melatih keterampilan membuat berbagai jenis plot.

## 2. *Importing data*

Praktikum sebelumnya telah menggunakan dataset yang sudah tersimpan sebagai objek R. Dalam pekerjaan analisis data, jarang sekali kita seberuntung itu; biasanya kita harus mengimpor data ke dalam R dari sebuah file, basis data, atau sumber lain. Saat ini, salah satu cara paling umum untuk menyimpan dan berbagi data untuk analisis adalah melalui spreadsheet elektronik. Spreadsheet menyimpan data dalam baris dan kolom. Pada dasarnya, spreadsheet adalah versi file dari sebuah data frame. Saat menyimpan tabel semacam itu ke dalam file komputer, kita memerlukan cara untuk menentukan kapan sebuah baris atau kolom berakhir dan yang lain dimulai. Hal ini pada gilirannya mendefinisikan sel, yaitu tempat nilai tunggal disimpan.

Praktikum minggu ini kita menguraikan cara memuat data dari file ke dalam R. Pertama, penting untuk mengidentifikasi lokasi file; oleh karena itu, kita membahas jalur file (file path) dan direktori kerja (working directory). Selanjutnya, kita membahas tipe file (teks atau biner) dan encoding (seperti ASCII dan Unicode), yang keduanya penting dalam proses impor data. Lalu, kita memperkenalkan fungsi populer untuk impor data, yang disebut parser. Terakhir, cara menyimpan data di spreadsheet.

### Menavigasi dan mengelola sistem file

Langkah pertama ketika mengimpor data dari spreadsheet adalah menemukan file yang berisi data tersebut. Meskipun kami tidak menyarankannya, Anda bisa menggunakan pendekatan seperti saat membuka file di Microsoft Excel, yaitu dengan mengklik menu “File” di RStudio, lalu “Import Dataset”, dan kemudian menelusuri folder hingga menemukan file yang dimaksud. Namun, biasanya kita menulis kode alih-alih menggunakan pendekatan klik-tombol.

#### Sistem file

Sistem file komputer bisa kita umpamakan sebagai serangkaian folder bertingkat, di mana masing-masing berisi folder lain dan file. Folder disebut direktori. Folder yang berisi semua folder lain disebut direktori root. Sementara itu, direktori tempat kita berada saat ini disebut direktori kerja (working directory). Direktori kerja akan berubah sesuai pergerakan kita melalui folder; anggap saja ini sebagai lokasi kita saat ini.

#### Path relatif dan penuh

Path sebuah file adalah daftar nama direktori yang bisa dianggap sebagai instruksi folder apa yang harus dibuka, dan dalam urutan apa, untuk menemukan file tersebut. Jika instruksi tersebut dimulai dari direktori

root, kita menyebutnya path penuh (full path). Jika instruksi dimulai dari direktori kerja, kita menyebutnya path relatif. Contoh path penuh pada sistem dapat dilihat dengan mengetik:

```
system.file(package = "dslabs")
```

```
## [1] "/Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/library/dslabs"
```

Hasilnya akan berbeda pada tiap komputer. Fungsi `system.file` menemukan path penuh ke file yang ditambahkan ke sistem Anda saat menginstal paket **dslabs**. String yang dipisahkan oleh tanda garis miring adalah nama direktori. Garis miring pertama mewakili direktori root dan kita tahu ini path penuh karena dimulai dengan garis miring.

Kita dapat menggunakan fungsi `list.files` untuk menampilkan nama file dan direktori dalam direktori tertentu. Misalnya:

```
dir <- system.file(package = "dslabs")
list.files(dir)
```

```
##  [1] "data"        "DESCRIPTION" "extdata"     "help"        "html"
##  [6] "INDEX"       "Meta"        "NAMESPACE"   "R"           "script"
```

Perhatikan bahwa nama file tersebut tidak diawali garis miring, yang berarti mereka adalah path relatif. Path relatif ini memberikan lokasi file atau direktori jika path yang tersimpan dalam `dir` dianggap sebagai direktori kerja.

> **Catatan:** Kita tidak akan banyak menggunakan fungsi `system.file` dalam pekerjaan analisis data sehari-hari. Kami memperkenalkannya di sini karena fungsi ini memudahkan berbagi spreadsheet untuk latihan. Spreadsheet tersebut ada dalam direktori **extdata**.

### Direktori kerja

Sangat disarankan hanya menggunakan path relatif di dalam kode. Alasannya adalah path penuh bersifat unik untuk komputer dan kita ingin kode tetap dapat digunakan di komputer lain (portable). Untuk mengetahui path penuh dari direktori kerja, gunakan fungsi `getwd`. Jika ingin mengubah direktori kerja, gunakan fungsi `setwd` atau ubah melalui RStudio dengan mengklik "Session".

Ketika memulai sebuah proyek, pilihlah direktori untuk menyimpan semua file terkait proyek tersebut dan jadikan direktori itu sebagai direktori kerja saat menjalankan analisis. Hal ini memudahkan karena jika kita memberikan path relatif ke fungsi impor, R akan mengasumsikan file dicari dalam direktori kerja. B

### Membuat nama path

Fungsi `file.path` menggabungkan karakter untuk membentuk path lengkap, dengan memastikan kompatibilitas terhadap sistem operasi. Linux dan Mac menggunakan garis miring (/), sedangkan Windows menggunakan garis miring terbalik (\). Fungsi ini bermanfaat karena sering kali kita ingin mendefinisikan path menggunakan variabel.

Contoh:

```
dir <- system.file(package = "dslabs")
file_path <- file.path(dir, "extdata/murders.csv")
```

Kita dapat menyalin file dengan path penuh `file_path` ke direktori kerja menggunakan fungsi `file.copy`:

```
file.copy(file_path, "murders.csv")
```

```
## [1] FALSE
```

Jika penyalinan berhasil, fungsi ini akan mengembalikan **TRUE**. Kita menggunakan nama file yang sama untuk tujuan penyalinan, tetapi bisa juga memberi nama lain. Jika sudah ada file dengan nama tersebut di direktori tujuan, penyalinan akan gagal kecuali argumen `overwrite` diatur.

## Jenis File

Untuk sebagian besar aplikasi analisis data, file umumnya dapat diklasifikasikan menjadi dua kategori: **file teks** dan **file biner**.

### File Teks

File *murders.csv* yang digunakan merupakan file teks. Salah satu keuntungan utama dari file teks adalah kita bisa dengan mudah "melihat" isinya tanpa perlu membeli perangkat lunak khusus atau mengikuti instruksi yang rumit. Editor teks apa pun dapat digunakan untuk memeriksa file teks, termasuk editor gratis seperti RStudio atau *nano*. Untuk mencobanya, buka file csv menggunakan fitur "Open file" di RStudio. Anda seharusnya bisa langsung melihat isinya di editor. Ketika file teks digunakan untuk menyimpan lembar kerja (*spreadsheet*), pemisah baris (*line break*) digunakan untuk memisahkan baris, dan sebuah karakter yang telah ditentukan, disebut **delimiter**, digunakan untuk memisahkan kolom dalam sebuah baris. Delimiter yang paling umum adalah koma (`,`), titik koma (`;`), spasi (), dan tab (`\t`).
Cara membaca file dengan delimiter berbeda ke dalam R juga berbeda, sehingga kita perlu mengetahui delimiter apa yang digunakan. Dalam beberapa kasus, delimiter dapat ditebak dari akhiran nama file (*suffix*). Misalnya, file yang diakhiri dengan **csv** atau **tsv** diharapkan dipisahkan dengan koma dan tab. Namun, lebih sulit menebak delimiter untuk file yang berakhiran **txt**. Karena itu, disarankan untuk melihat isi file secara langsung daripada hanya menebak dari akhiran. Anda dapat melihat sejumlah baris dari file di R menggunakan fungsi `readLines`:

```
readLines("murders.csv", n = 3)
```

```
## [1] "state,abb,region,population,total" "Alabama,AL,South,4779736,135"
## [3] "Alaska,AK,West,710231,19"
```

Hasil tersebut langsung menunjukkan bahwa file dipisahkan oleh koma. Selain itu, terlihat juga bahwa file memiliki *header*: baris pertama berisi nama kolom, bukan data. Hal ini penting untuk diketahui. Sebagian besar *parser* mengasumsikan file dimulai dengan header, tetapi tidak semua file memilikinya.

### File Biner

Membuka file gambar seperti jpg atau png dengan editor teks atau menggunakan `readLines` di R tidak akan menampilkan isi yang dapat dipahami karena file tersebut merupakan file biner. Berbeda dengan file teks yang dirancang agar mudah dibaca manusia dan mengikuti konvensi standar, file biner dapat memiliki banyak format sesuai jenis datanya. Meskipun fungsi `readBin` di R dapat memproses file biner apa pun, menafsirkan hasilnya membutuhkan pemahaman mendalam tentang struktur file tersebut. Kita akan fokus pada format biner yang umum untuk spreadsheet: **xls** dan **xlsx** milik Microsoft Excel.

### Encoding

Masalah yang sering muncul ketika mengimpor data, baik teks maupun biner, adalah salah mengenali *encoding* file. Pada dasarnya, komputer menerjemahkan semua hal ke dalam urutan angka 0 dan 1. **ASCII** adalah sistem *encoding* yang memberikan nomor tertentu pada setiap karakter. Dengan menggunakan 7 bit, ASCII dapat mewakili 2^7=128 simbol unik, cukup untuk semua karakter papan ketik bahasa Inggris. Namun, banyak bahasa di dunia memiliki karakter yang tidak ada dalam ASCII. Misalnya, huruf *é* dalam kata "México" tidak ada dalam katalog ASCII. Untuk mengatasi hal ini, dikembangkan *encoding* yang lebih luas seperti **Unicode**. Unicode memiliki beberapa variasi berdasarkan panjang bit: **UTF-8**, **UTF-16**, dan **UTF-32**. RStudio biasanya menggunakan **UTF-8** sebagai default. Menariknya, ASCII adalah subset dari UTF-8, sehingga jika sebuah file dikodekan dalam ASCII lalu diasumsikan UTF-8, tidak akan menimbulkan masalah. Namun, ada juga *encoding* lain, misalnya **ISO-8859-1** (dikenal juga sebagai *Latin-1*) yang dibuat untuk bahasa Eropa Barat, **Big5** untuk bahasa Tionghoa Tradisional, dan **ISO-8859-6** untuk bahasa Arab. Paket **dslabs** menyertakan sebuah file yang tidak dikodekan dengan UTF-8 sebagai contoh. Perhatikan karakter aneh yang muncul saat mencoba membaca baris pertamanya:

```
fn <- "calificaciones.csv"
file.copy(file.path(system.file("extdata", package = "dslabs"), fn), fn)
```

```
## [1] FALSE
```

```
readLines(fn, n = 1)
```

```
## [1] "\"nombre\",\"f.n.\",\"estampa\",\"puntuaci\xf3n\""
```

## Parser

Fungsi untuk mengimpor data, atau *parser*, tersedia dari **base R**. Namun, fungsi yang lebih kuat dan seringkali lebih cepat tersedia dalam paket **readr**, **readxl**, dan **data.table**.

### Base R

Base R menyediakan beberapa *file parser* seperti `read.csv`, `read.table`, dan `read.delim`. Argumen pertama dapat berupa jalur lengkap atau jalur relatif. Jika jalur relatif diberikan, parser akan mengasumsikan ingin mencari di direktori kerja. Oleh karena itu, untuk membaca file **murders.csv** yang sebelumnya disalin ke direktori kerja, kita cukup mengetik:

```
dat <- read.csv("murders.csv")
```

Fungsi impor R-base yang sering berguna adalah `scan`, karena menyediakan fleksibilitas tinggi. Saat membaca *spreadsheet*, banyak hal bisa saja bermasalah. Misalnya, file mungkin memiliki header multilini atau ada sel yang hilang. Dengan pengalaman, Anda akan belajar bagaimana menangani berbagai tantangan. Membaca *help file* dari fungsi-fungsi yang dibahas di sini akan sangat membantu. Dengan `scan`, kita dapat membaca setiap sel dalam sebuah file. Berikut contohnya:

```
x <- scan("murders.csv", sep = ",", what = "c")
x[1:10]
```

```
##  [1] "state"      "abb"        "region"     "population" "total"
##  [6] "Alabama"    "AL"         "South"      "4779736"    "135"
```

### readr

Paket **readr** mencakup *parser* untuk membaca *spreadsheet* berupa *text file* ke dalam R. Paket ini merupakan bagian dari **tidyverse**, tetapi bisa dimuat secara langsung dengan:

```
library(readr)
```

Fungsi-fungsi berikut tersedia untuk membaca *spreadsheet* dari *text file*:

| Fungsi | Format | Akhiran umum |
|---|---|---|
| `read_table` | nilai dipisahkan spasi | txt |
| `read_csv` | nilai dipisahkan koma | csv |
| `read_csv2` | nilai dipisahkan titik koma | csv |
| `read_tsv` | nilai dipisahkan tab | tsv |
| `read_delim` | format teks umum (delimiter didefinisikan) | txt |

Paket ini juga menyediakan `read_lines` dengan fungsi mirip `readLines`. Contoh membaca **murders.csv**:

```
dat <- read_csv("murders.csv")
```

```
## Rows: 51 Columns: 5
## -- Column specification --------------------------------------------------------
```

```
## Delimiter: ","
## chr (3): state, abb, region
## dbl (2): population, total
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Output akan menunjukkan spesifikasi kolom (tipe data), serta menghasilkan sebuah *tibble* (bukan sekadar *data frame*). Pesan tersebut bisa disembunyikan dengan `show_col_types = FALSE`. Selain itu, **readr** mengizinkan kita menentukan *encoding*, dan memiliki fungsi untuk menebak encoding:

```
guess_encoding("murders.csv")
```

```
## # A tibble: 1 x 2
##   encoding confidence
##   <chr>         <dbl>
## 1 ASCII             1
```

Contoh untuk file dengan karakter asing (**calificaciones.csv**):

```
guess_encoding("calificaciones.csv")
```

```
## # A tibble: 3 x 2
##   encoding   confidence
##   <chr>           <dbl>
## 1 ISO-8859-1       0.92
## 2 ISO-8859-2       0.72
## 3 ISO-8859-9       0.53
```

Setelah tahu encoding, kita dapat menentukannya lewat argumen `locale`:

```
dat <- read_csv("calificaciones.csv", show_col_types = FALSE,
                locale = locale(encoding = "ISO-8859-1"))
names(dat)
```

```
## [1] "nombre"     "f.n."       "estampa"    "puntuación"
```

### readxl

Paket **readxl** menyediakan fungsi untuk membaca format Microsoft Excel.

```
library(readxl)
```

Fungsi utamanya:

| Fungsi | Format terduga otomatis | Akhiran umum |
|---|---|---|
| `read_excel` | deteksi otomatis | xls, xlsx |
| `read_xls` | format lama | xls |
| `read_xlsx` | format baru | xlsx |

File Excel dapat berisi lebih dari satu *spreadsheet* (*sheet*). Secara default, fungsi di atas membaca *sheet* pertama. Kita bisa menggunakan `excel_sheets` untuk melihat semua nama *sheet*, lalu memilih salah satunya dengan argumen `sheet`.

### data.table

Paket **data.table** menyediakan fungsi **fread**, sebuah utilitas yang kuat dan cepat untuk membaca dataset besar. `fread` secara otomatis mendeteksi format input, termasuk file teks terpisah atau bahkan file terkompresi

seperti gzip/zip. Kecepatan `fread` umumnya lebih unggul dibanding parser lain.

```
library(data.table)
dat <- fread("murders.csv")
```

Catatan: `fread` menghasilkan objek bertipe **data.table**.

### Mengunduh file

Banyak data tersedia di internet. Jika data tersebut berbentuk file, kita bisa mengunduhnya lalu mengimpornya, atau bahkan langsung membacanya dari web. Contoh file **murders.csv** dari GitHub:

```
url <- paste0("https://raw.githubusercontent.com/",
              "rafalab/dslabs/master/inst/extdata/murders.csv")
dat <- read.csv(url)
```

Untuk menyimpan salinan lokal:

```
download.file(url, "murders.csv")
```

Catatan: `download.file` akan menimpa file lama tanpa peringatan. Fungsi lain yang berguna:

`tempdir()` → membuat direktori sementara dengan nama unik.

`tempfile()` → membuat string nama file sementara yang unik.

Contoh:

```
tmp_filename <- tempfile()
download.file(url, tmp_filename)
dat <- read_csv(tmp_filename)
```

```
## Rows: 51 Columns: 5
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (3): state, abb, region
## dbl (2): population, total
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
file.remove(tmp_filename)
```

```
## [1] TRUE
```

### Mengorganisasi Data dengan Spreadsheet

Seorang analis data perlu mengumpulkan data, atau bekerja dengan orang lain dalam mengumpulkan data, dengan cara yang paling nyaman disimpan dalam sebuah spreadsheet. Meskipun mengisi spreadsheet secara manual adalah praktik yang sangat kami tidak anjurkan, dan kami lebih merekomendasikan proses ini diotomatisasi sebanyak mungkin, terkadang kita tetap harus melakukannya. Oleh karena itu, pada bagian ini, kami memberikan beberapa rekomendasi tentang bagaimana cara mengorganisasi data dalam sebuah spreadsheet. Walaupun ada paket R yang dirancang untuk membaca spreadsheet Microsoft Excel, umumnya kita ingin menghindari format ini. Sebagai gantinya, kami merekomendasikan Google Sheets sebagai perangkat lunak gratis. Berikut kami rangkum rekomendasi yang dibuat oleh Karl Broman dan Kara Woo. Silakan baca makalah asli mereka untuk detail lebih lanjut.

**Konsisten** – Sebelum mulai memasukkan data, buatlah rencana. Setelah punya rencana, konsistenlah dan patuhi itu.

**Pilih Nama yang Baik untuk Segala Sesuatu** – Pilih nama untuk objek, file, dan direktori yang mudah diingat, mudah dieja, dan deskriptif. Ini memang sulit dicapai, tetapi penting. Aturan penting: jangan gunakan spasi, gunakan garis bawah _ atau tanda hubung - sebagai gantinya. Hindari simbol, gunakan huruf dan angka saja.

**Tulis Tanggal dengan Format YYYY-MM-DD** – Untuk menghindari kebingungan, sangat disarankan menggunakan standar internasional ISO 8601.

**Jangan Ada Sel Kosong** – Isi semua sel dan gunakan kode umum untuk data yang hilang.

**Satu Hal per Sel** – Lebih baik menambahkan kolom baru untuk informasi tambahan daripada menaruh lebih dari satu informasi dalam satu sel.

**Buat Bentuk Persegi Panjang** – Spreadsheet sebaiknya berbentuk persegi panjang (jumlah baris dan kolom teratur).

**Buat Kamus Data (Data Dictionary)** – Jika perlu menjelaskan hal-hal seperti arti kolom atau label pada variabel kategorikal, tuliskan dalam file terpisah.

**Jangan Ada Perhitungan dalam File Data Mentah** – Excel memungkinkan melakukan perhitungan, tetapi jangan masukkan hal ini dalam spreadsheet. Simpan kode perhitungan dalam script terpisah.

**Jangan Gunakan Warna Font atau Highlight sebagai Data** – Sebagian besar fungsi impor tidak dapat membaca informasi ini. Sebaiknya informasi tersebut dikodekan sebagai variabel.

**Buat Cadangan (Backup)** – Lakukan pencadangan data secara rutin.

**Gunakan Validasi Data untuk Menghindari Kesalahan** – Manfaatkan fitur validasi data dalam perangkat lunak spreadsheet agar proses lebih bebas dari kesalahan dan aman dari repetitive stress injury.

**Simpan Data sebagai File Teks** – Untuk berbagi, simpan file dalam format teks dengan pemisah koma (CSV) atau tab (TSV).

# 3. Visualisasi Data

Dalam bab ini, kita akan mendemonstrasikan bagaimana kode **ggplot2** yang relatif sederhana dapat menghasilkan plot yang informatif sekaligus estetik. Sebagai motivasi, kita akan membuat plot yang membantu kita memahami tren kesehatan dan ekonomi dunia. Praktikum ini juga akan membahas prinsip umum visualisasi data.

## Studi Kasus 1: Wawasan Baru tentang Kemiskinan

Hans Rosling adalah salah satu pendiri **Gapminder Foundation**, sebuah organisasi yang didedikasikan untuk mendidik publik dengan menggunakan data untuk membongkar mitos tentang negara-negara berkembang. Organisasi ini menggunakan data untuk menunjukkan bahwa tren nyata dalam kesehatan dan ekonomi sering bertolak belakang dengan narasi media sensasional yang berfokus pada bencana, tragedi, dan kejadian negatif lainnya. Hans Rosling menyampaikan tren berbasis data secara dramatis dengan caranya sendiri, melalui visualisasi data yang efektif. Bagian ini didasarkan pada dua presentasi Rosling: **New Insights on Poverty** dan **The Best Stats You've Ever Seen**. Secara khusus, kita akan mencoba menjawab dua pertanyaan berikut menggunakan data:

1. Apakah adil menggambarkan dunia saat ini sebagai terbagi antara negara-negara kaya di Barat dan negara berkembang di Afrika, Asia, dan Amerika Latin?
2. Apakah ketimpangan pendapatan antarnegara memburuk selama 40 tahun terakhir?

Untuk menjawab pertanyaan ini, kita akan menggunakan dataset **gapminder** dari paket **dslabs**, yang disusun dari beberapa spreadsheet Gapminder Foundation. Contoh akses data:

```r
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v purrr     1.0.4
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.2     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::between()     masks data.table::between()
## x dplyr::filter()      masks stats::filter()
## x dplyr::first()       masks data.table::first()
## x lubridate::hour()    masks data.table::hour()
## x lubridate::isoweek() masks data.table::isoweek()
## x dplyr::lag()         masks stats::lag()
## x dplyr::last()        masks data.table::last()
## x lubridate::mday()    masks data.table::mday()
## x lubridate::minute()  masks data.table::minute()
## x lubridate::month()   masks data.table::month()
## x lubridate::quarter() masks data.table::quarter()
## x lubridate::second()  masks data.table::second()
## x purrr::transpose()   masks data.table::transpose()
## x lubridate::wday()    masks data.table::wday()
## x lubridate::week()    masks data.table::week()
## x lubridate::yday()    masks data.table::yday()
## x lubridate::year()    masks data.table::year()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
library(dslabs)
gapminder |> as_tibble()
```

```
## # A tibble: 10,545 x 9
##    country   year infant_mortality life_expectancy fertility population      gdp
##    <fct>    <int>            <dbl>           <dbl>     <dbl>      <dbl>    <dbl>
##  1 Albania   1960            115.             62.9      6.19    1636054 NA
##  2 Algeria   1960            148.             47.5      7.65   11124892  1.38e10
##  3 Angola    1960            208              36.0      7.32    5270844 NA
##  4 Antigua~  1960             NA              63.0      4.43      54681 NA
##  5 Argenti~  1960             59.9            65.4      3.11   20619075  1.08e11
##  6 Armenia   1960             NA              66.9      4.55    1867396 NA
##  7 Aruba     1960             NA              65.7      4.82      54208 NA
##  8 Austral~  1960             20.3            70.9      3.45   10292328  9.67e10
##  9 Austria   1960             37.3            68.8      2.7     7065525  5.24e10
## 10 Azerbai~  1960             NA              61.3      5.57    3897889 NA
## # i 10,535 more rows
## # i 2 more variables: continent <fct>, region <fct>
```

Dataset ini berisi data kesehatan, ekonomi, dan demografi dari berbagai negara sejak tahun 1960. Sebagai awal, kita akan menguji pengetahuan tentang perbedaan angka kematian anak di beberapa negara pada tahun 2015. Misalnya: Sri Lanka vs Turki, Polandia vs Korea Selatan, Malaysia vs Rusia, Pakistan vs Vietnam, dan Thailand vs Afrika Selatan. Tanpa data, orang cenderung menganggap negara non-Eropa memiliki angka kematian anak lebih tinggi. Namun, data menunjukkan hasil berbeda. Contoh:

```r
gapminder |>
filter(year == 2015 & country %in% c("Sri Lanka","Turkey")) |>
select(country, infant_mortality)
```

```
##     country infant_mortality
## 1 Sri Lanka              8.4
## 2    Turkey             11.6
```

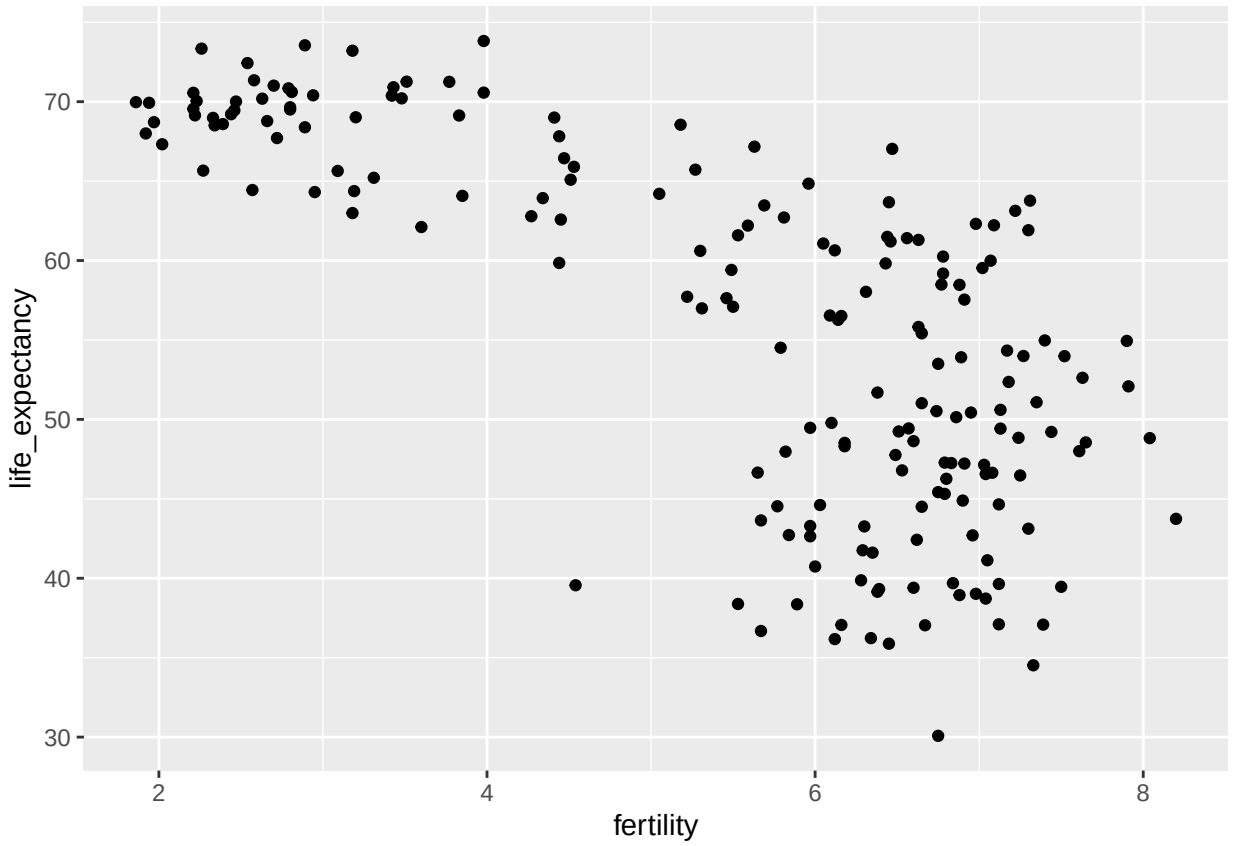
| country | infant_mortality | country | infant_mortality |
|---|---|---|---|
| Sri Lanka | 8.4 | Turkey | 11.6 |
| Poland | 4.5 | South Korea | 2.9 |
| Malaysia | 6.0 | Russia | 8.2 |
| Pakistan | 65.8 | Vietnam | 17.3 |
| Thailand | 10.5 | South Africa | 33.6 |

Kita melihat bahwa negara-negara Eropa dalam daftar ini memiliki angka kematian anak yang lebih tinggi: Polandia memiliki angka yang lebih tinggi dibanding Korea Selatan, dan Rusia memiliki angka yang lebih tinggi dibanding Malaysia. Kita juga melihat bahwa Pakistan memiliki angka yang jauh lebih tinggi dibanding Vietnam, dan Afrika Selatan jauh lebih tinggi dibanding Thailand. Ternyata, ketika Hans Rosling memberikan kuis ini kepada kelompok orang berpendidikan, skor rata-rata yang diperoleh kurang dari 2,5 dari 5, bahkan lebih buruk dibanding jika mereka menebak secara acak. Hal ini menunjukkan bahwa lebih dari sekadar tidak tahu, kita justru mendapat informasi yang keliru.

## Scatterplot

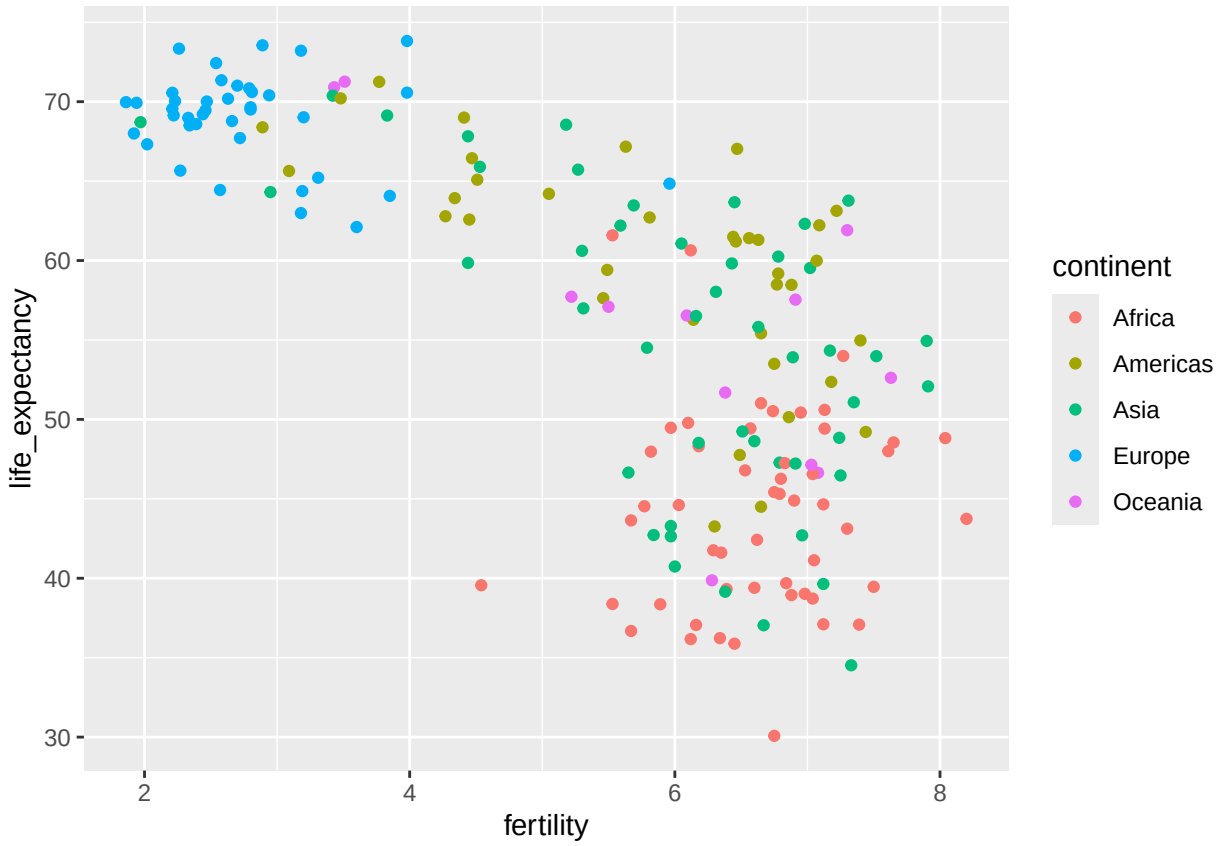
Kesalahpahaman muncul karena anggapan dunia terbagi dua: Barat dengan harapan hidup panjang dan keluarga kecil, serta negara berkembang dengan harapan hidup rendah dan keluarga besar. Apakah data mendukung pandangan dikotomis ini? Kita mulai dengan plot **fertility vs life expectancy** untuk tahun 1962:

```
filter(gapminder, year == 1962) |>
ggplot(aes(fertility, life_expectancy)) +
geom_point()
```




Hasilnya, sebagian besar negara memang terbagi ke dalam dua kelompok yang kontras. Namun, ketika diperiksa untuk tahun 2012 dengan pewarnaan per benua, pola ini tidak lagi terlihat jelas: banyak negara Asia, misalnya, telah mengalami kemajuan signifikan.

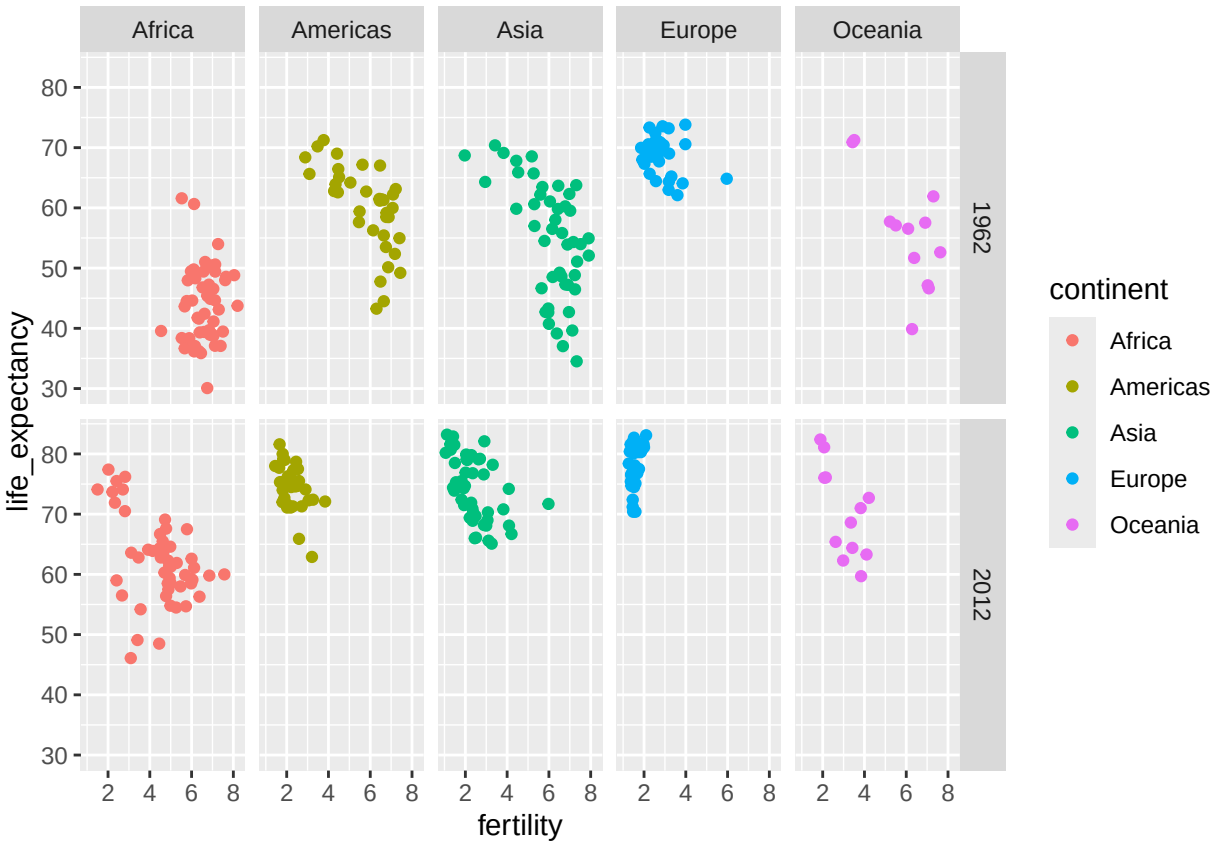
```
filter(gapminder, year == 1962) |>
ggplot( aes(fertility, life_expectancy, color = continent)) +
geom_point()
```




## Faceting

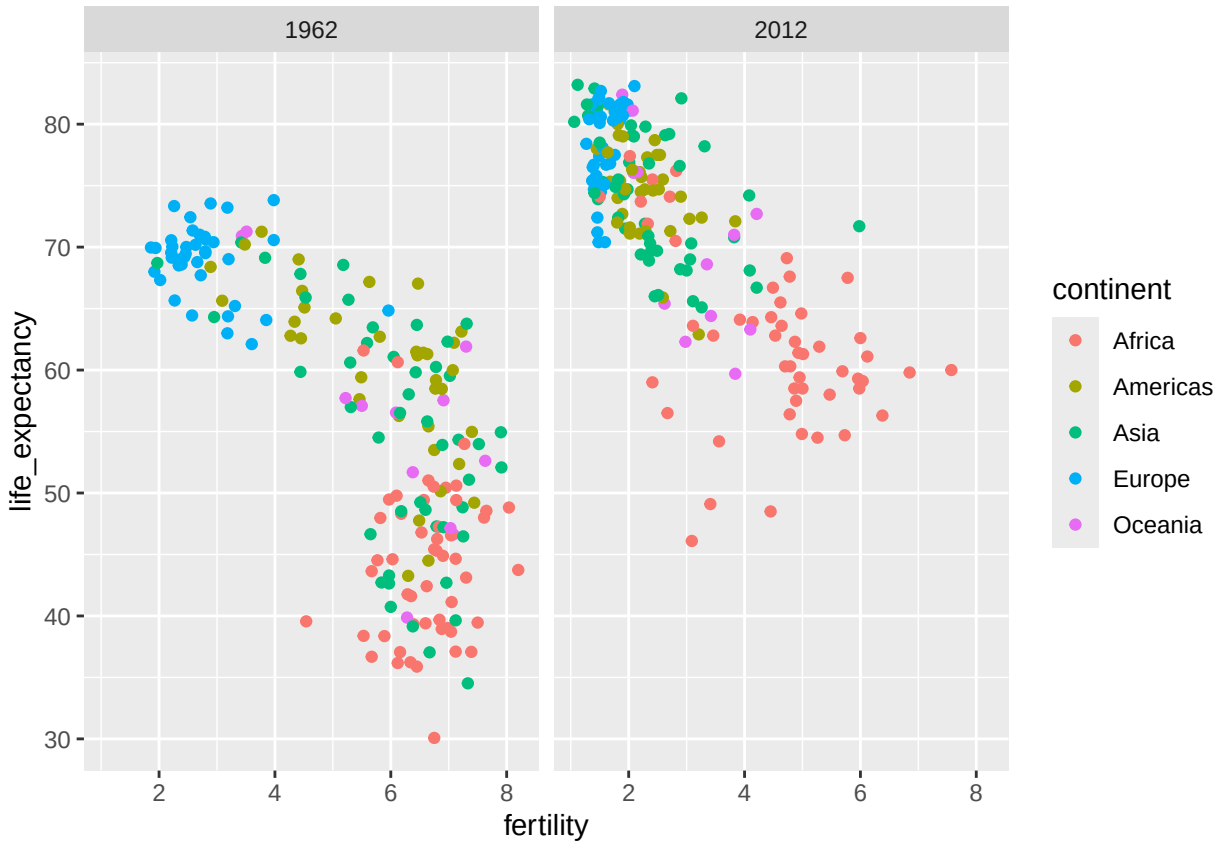
Kita dapat dengan mudah membuat plot data tahun 2012 dengan cara yang sama seperti yang kita lakukan untuk tahun 1962. Namun, untuk membuat perbandingan, plot berdampingan lebih disukai. Dalam **ggplot2**, kita dapat mencapainya dengan *faceting* variabel: kita membagi data berdasarkan suatu variabel dan membuat plot yang sama untuk setiap strata. Untuk melakukan *faceting*, kita menambahkan sebuah lapisan dengan fungsi **facet_grid**, yang secara otomatis memisahkan plot. Fungsi ini memungkinkan kita melakukan *faceting* hingga dua variabel, dengan kolom merepresentasikan satu variabel dan baris merepresentasikan variabel lainnya. Fungsi ini mengharuskan variabel baris dan kolom dipisahkan dengan tanda `~`. Berikut contoh *scatterplot* dengan **facet_grid** yang ditambahkan sebagai lapisan terakhir:

```
filter(gapminder, year %in% c(1962, 2012)) |>
ggplot(aes(fertility, life_expectancy, col = continent)) +
geom_point() +
facet_grid(year~continent)
```




Kita melihat sebuah plot untuk setiap pasangan benua/tahun. Namun, ini hanyalah sebuah contoh dan lebih dari yang kita butuhkan, karena yang sebenarnya kita inginkan hanyalah membandingkan tahun 1962 dan 2012. Dalam kasus ini, hanya ada satu variabel dan kita menggunakan tanda `.` untuk memberi tahu *facet* bahwa kita tidak menggunakan variabel kedua.

```
filter(gapminder, year %in% c(1962, 2012)) |>
ggplot(aes(fertility, life_expectancy, col = continent)) +
geom_point() +
facet_grid(. ~ year)
```




Plot ini dengan jelas menunjukkan bahwa sebagian besar negara telah berpindah dari kelompok negara berkembang ke kelompok dunia barat. Pada tahun 2012, pandangan mengenai dunia barat versus dunia berkembang tidak lagi relevan. Hal ini terlihat sangat jelas ketika membandingkan Eropa dengan Asia, di mana Asia mencakup beberapa negara yang telah mengalami peningkatan besar.

### facet_wrap

Untuk mengeksplorasi bagaimana transformasi ini terjadi sepanjang tahun, kita dapat membuat plot untuk beberapa tahun. Sebagai contoh, kita dapat menambahkan tahun 1970, 1980, 1990, dan 2000. Jika kita melakukan ini, kita tidak ingin semua plot berada pada satu baris, yang merupakan perilaku bawaan dari **facet_grid**, karena ukurannya akan menjadi terlalu tipis untuk menampilkan data. Sebagai gantinya, kita akan menggunakan beberapa baris dan kolom. Fungsi **facet_wrap** memungkinkan kita melakukan hal ini dengan secara otomatis membungkus serangkaian plot sehingga setiap tampilan memiliki dimensi yang mudah dibaca.

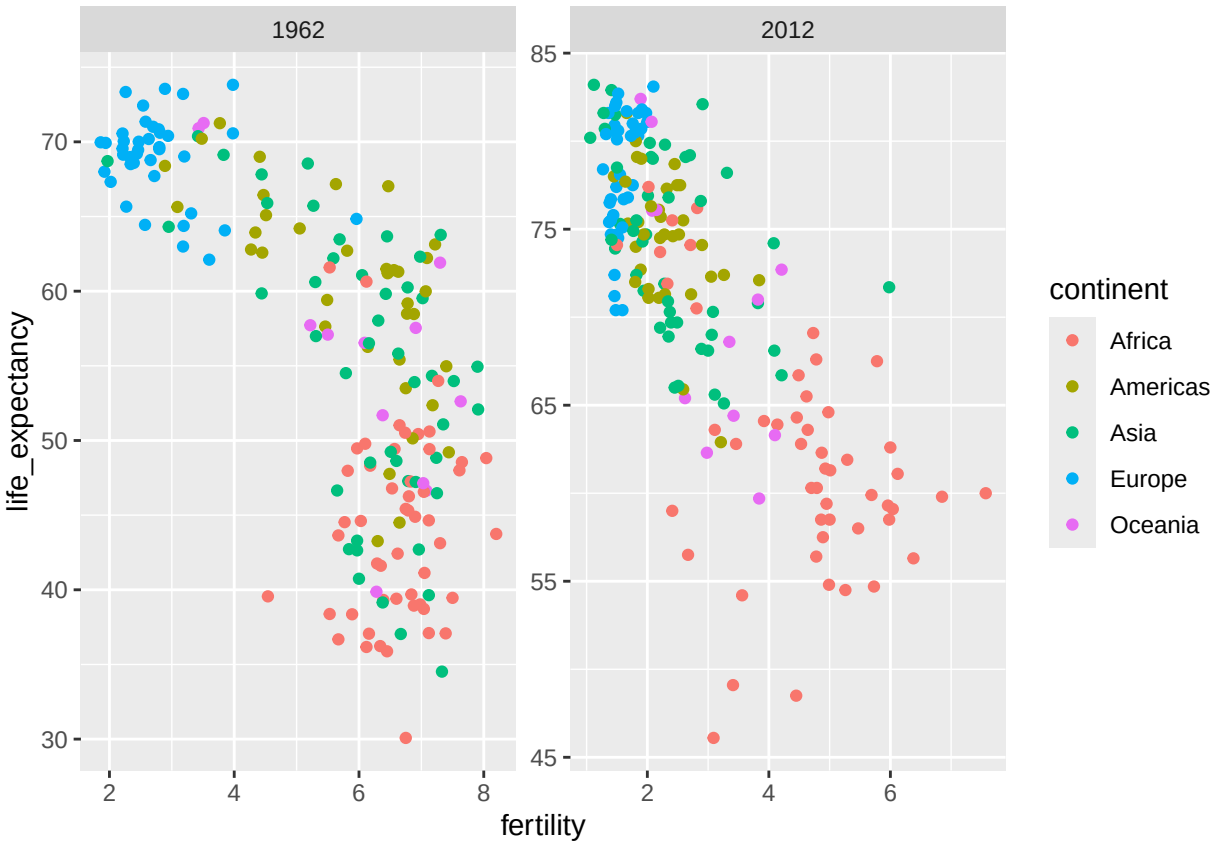
```
years <- c(1962, 1980, 1990, 2000, 2012)
continents <- c("Europe", "Asia")
gapminder |>
filter(year %in% years & continent %in% continents) |>
ggplot( aes(fertility, life_expectancy, col = continent)) +
geom_point() +
facet_wrap(~year)
```


Plot ini dengan jelas menunjukkan bagaimana sebagian besar negara Asia telah mengalami

**Skala tetap untuk perbandingan yang lebih baik**

Pilihan default untuk rentang sumbu sangatlah penting. Saat tidak menggunakan *facet*, rentang ini ditentukan oleh data yang ditampilkan dalam plot. Saat menggunakan *facet*, rentang ini ditentukan oleh data yang ditampilkan pada semua plot dan karenanya dijaga tetap sama di seluruh plot. Hal ini membuat perbandingan antar plot menjadi jauh lebih mudah. Sebagai contoh, pada plot di atas, kita dapat melihat bahwa harapan hidup meningkat dan tingkat fertilitas menurun di sebagian besar negara. Kita melihat hal ini karena sebaran titik berpindah. Hal ini tidak akan terjadi jika kita menyesuaikan skala:

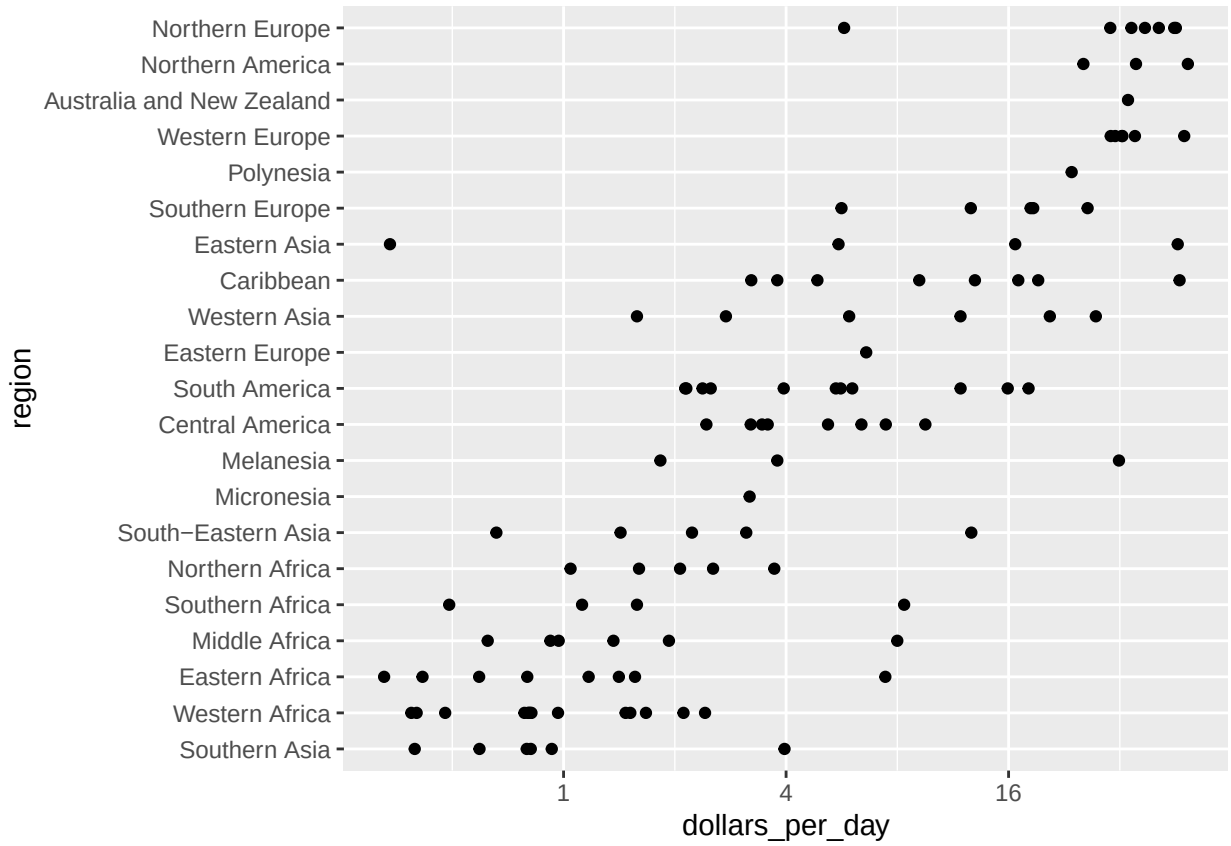
```
filter(gapminder, year %in% c(1962, 2012)) |>
ggplot(aes(fertility, life_expectancy, col = continent)) +
geom_point() +
facet_wrap(. ~ year, scales = "free")
```




## Membandingkan distribusi

Sebuah histogram menunjukkan kepada kita bahwa nilai distribusi pendapatan tahun 1970 memperlihatkan adanya dikotomi. Namun, histogram tidak menunjukkan apakah kedua kelompok negara tersebut adalah negara barat versus negara berkembang. Mari kita mulai dengan cepat memeriksa data berdasarkan wilayah. Kita menyusun ulang wilayah berdasarkan nilai median dan menggunakan skala log.

```
gapminder <- gapminder |>
mutate(dollars_per_day = gdp/population/365)
past_year <- 1970
gapminder |>
filter(year == past_year & !is.na(gdp)) |>
mutate(region = reorder(region, dollars_per_day, FUN = median)) |>
ggplot(aes(dollars_per_day, region)) +
geom_point() +
scale_x_continuous(trans = "log2")
```




Kita sudah dapat melihat bahwa memang ada dikotomi 'barat versus lainnya': kita melihat dua kelompok yang jelas, dengan kelompok kaya terdiri dari Amerika Utara, Eropa Utara dan Barat, serta Selandia Baru dan Australia. Kita mendefinisikan kelompok berdasarkan pengamatan ini :

```
gapminder <- gapminder |>
mutate(group = case_when(
region %in% c("Western Europe", "Northern Europe","Southern Europe",
"Northern America",
"Australia and New Zealand") ~ "West",
region %in% c("Eastern Asia", "South-Eastern Asia") ~ "East Asia",
region %in% c("Caribbean", "Central America",
"South America") ~ "Latin America",
continent == "Africa" &
region != "Northern Africa" ~ "Sub-Saharan",
TRUE ~ "Others"))
```

Kita mengubah variabel kelompok ini menjadi sebuah faktor untuk mengontrol urutan level-levelnya:

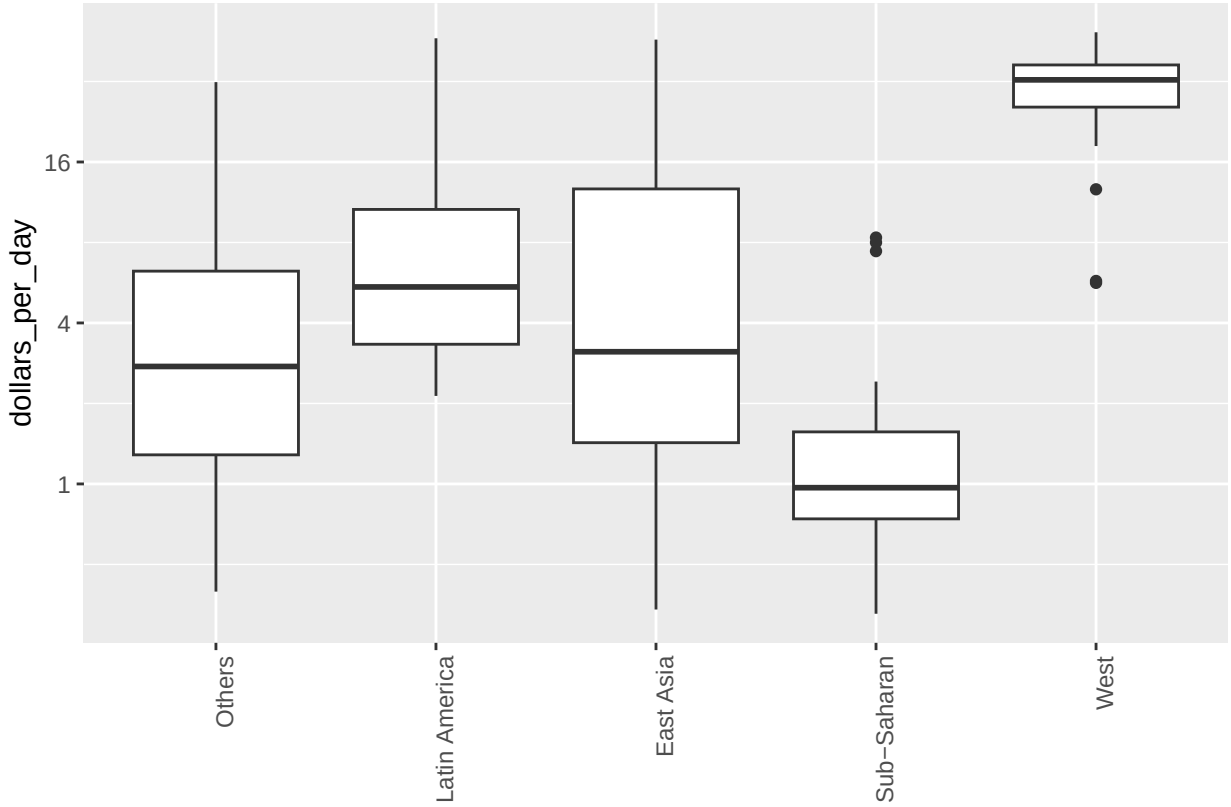
```
gapminder <- gapminder |>
mutate(group = factor(group, levels = c("Others", "Latin America",
"East Asia", "Sub-Saharan",
"West")))
```

Pada bagian berikutnya, kita akan mendemonstrasikan bagaimana memvisualisasikan dan membandingkan distribusi antar kelompok. ### Boxplot

Kita ingin membandingkan distribusi di antara lima kelompok ini untuk mengonfirmasi dikotomi 'barat versus sisanya'. Jumlah titik dalam setiap kategori cukup besar sehingga plot ringkasan mungkin berguna. Kita bisa saja membuat lima histogram atau lima plot densitas, tetapi akan lebih praktis jika semua ringkasan visual

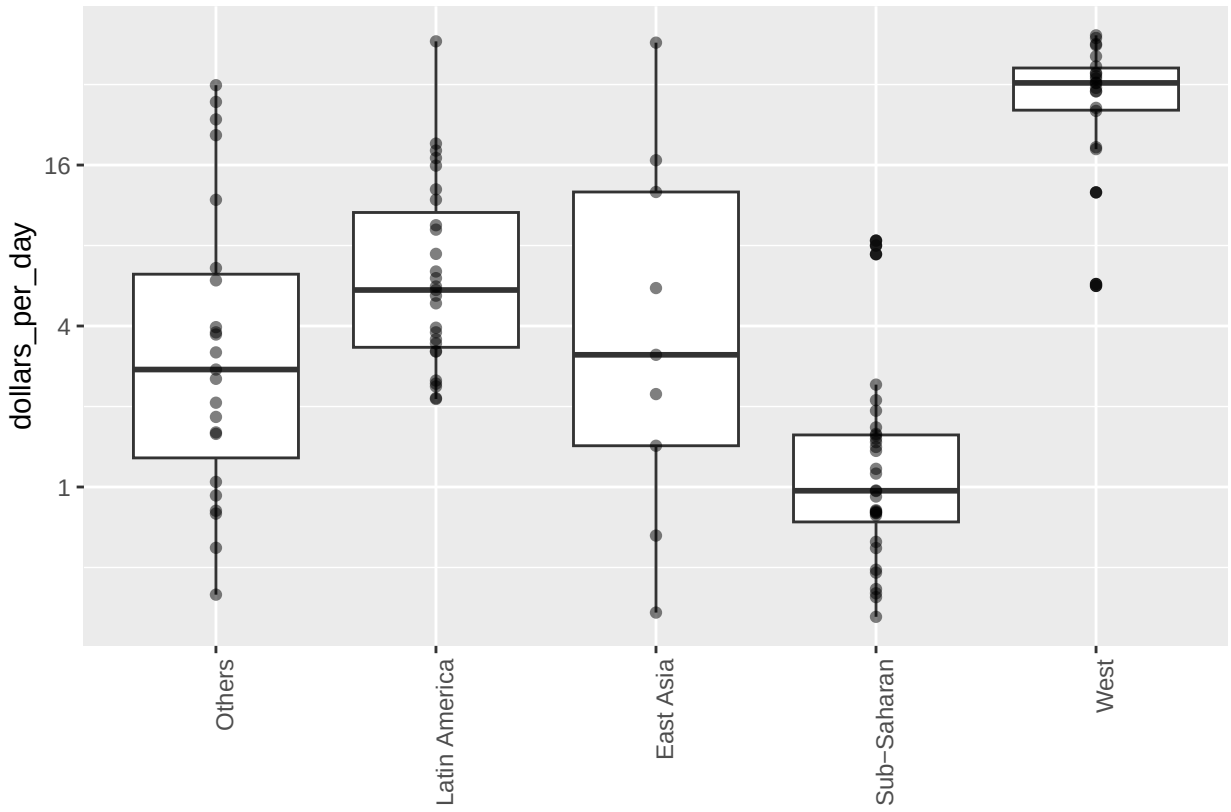
tersebut berada dalam satu plot. Oleh karena itu, kita mulai dengan menumpuk boxplot berdampingan. Perlu dicatat bahwa kita menambahkan lapisan theme(axis.text.x = element_text(angle = 90, hjust = 1)) untuk memutar label kelompok secara vertikal, karena tidak muat jika ditampilkan secara horizontal, dan kita menghapus label sumbu untuk memberi ruang.

```
p <- gapminder |>
filter(year == past_year & !is.na(gdp)) |>
ggplot(aes(group, dollars_per_day)) +
geom_boxplot() +
scale_y_continuous(trans = "log2") +
xlab("") +
theme(axis.text.x = element_text(angle = 90, hjust = 1))
p
```




Boxplot memiliki keterbatasan karena dengan meringkas data menjadi lima angka, kita mungkin melewatkan karakteristik penting dari data. Salah satu cara untuk menghindari hal ini adalah dengan menampilkan datanya.

```
p + geom_point(alpha = 0.5)
```

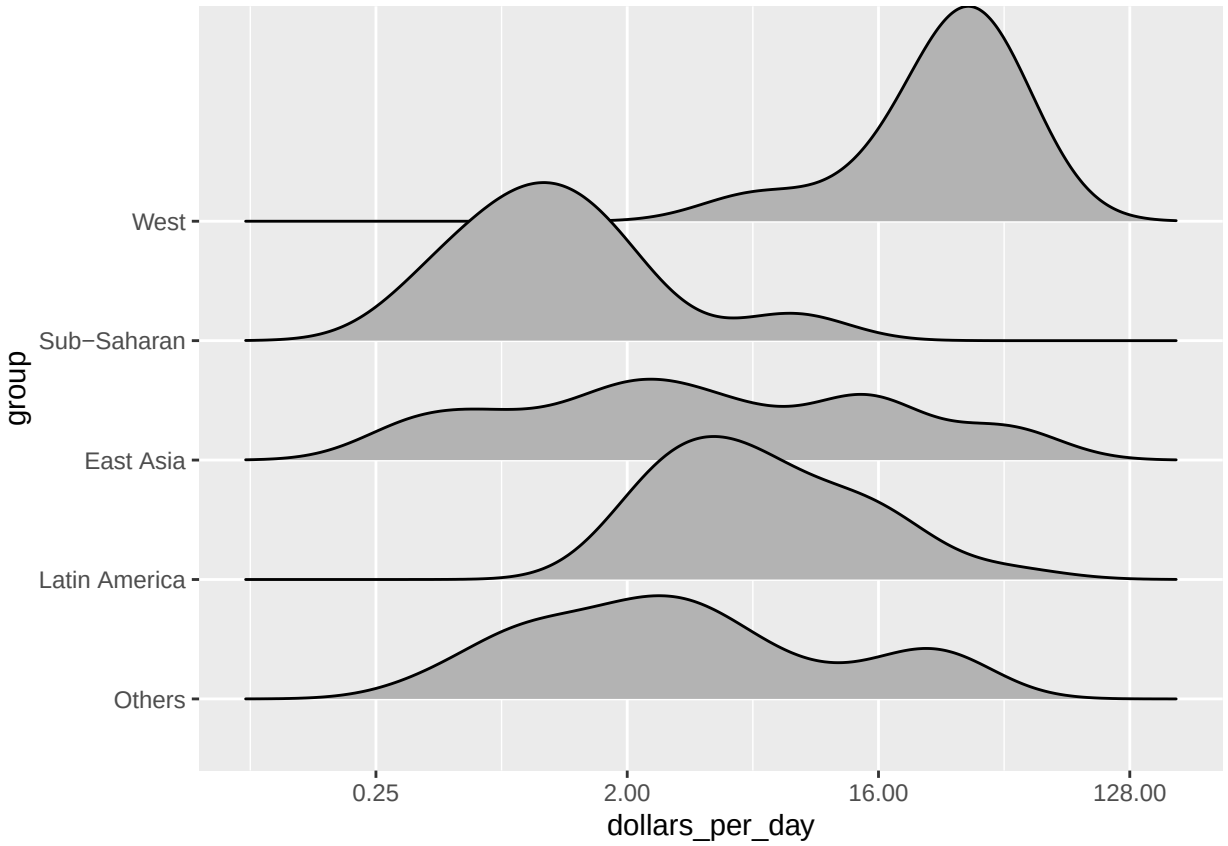



### Ridge plots

Menampilkan setiap titik individu tidak selalu mengungkapkan karakteristik penting dari distribusi. Meskipun tidak terjadi pada kasus ini, ketika jumlah titik data sangat besar sehingga terjadi over-plotting, menampilkan data dapat menjadi kontra-produktif. Boxplot membantu dalam hal ini dengan memberikan ringkasan lima angka, tetapi ini juga memiliki keterbatasan. Misalnya, boxplot tidak memungkinkan kita menemukan distribusi bimodal. Dalam kasus ketika kita khawatir ringkasan boxplot terlalu sederhana, kita dapat menampilkan stacked smooth densities atau histogram. Kita menyebut ini sebagai ridge plots. Karena kita terbiasa memvisualisasikan kepadatan dengan nilai pada sumbu-x, kita menumpuknya secara vertikal. Selain itu, karena pendekatan ini membutuhkan lebih banyak ruang, lebih praktis jika ditumpuk saling menimpa (overlay). Paket ggridges menyediakan fungsi yang mudah untuk melakukan ini. Berikut adalah data pendapatan yang ditampilkan di atas dengan boxplot tetapi divisualisasikan menggunakan ridge plot.

```
library(ggridges)
p <- gapminder |>
filter(year == past_year & !is.na(dollars_per_day)) |>
ggplot(aes(dollars_per_day, group)) +
scale_x_continuous(trans = "log2")
p + geom_density_ridges()
```

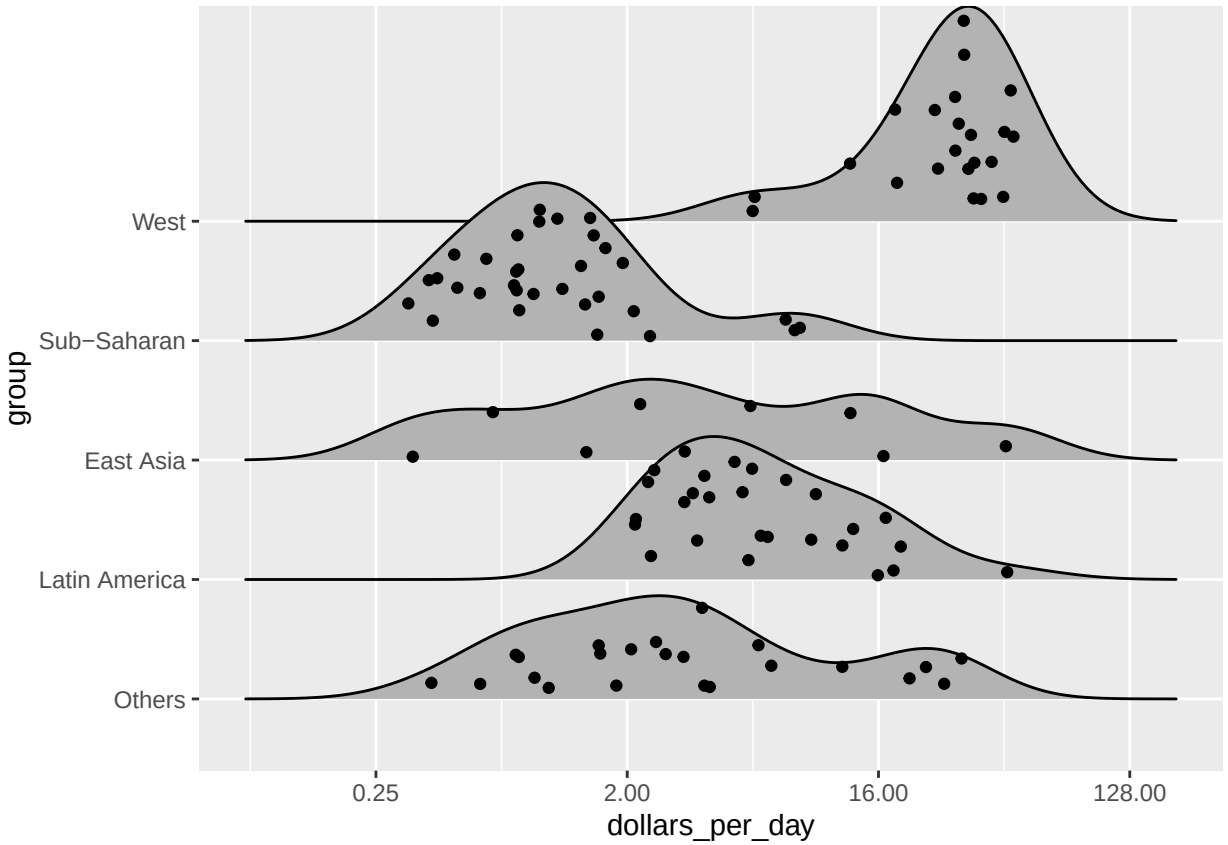
```
## Picking joint bandwidth of 0.648
```




Perhatikan bahwa kita harus membalik sumbu x dan y yang digunakan untuk boxplot. Parameter yang berguna pada geom_density_ridges adalah scale, yang memungkinkan kita menentukan jumlah tumpang tindih, dengan scale = 1 berarti tidak ada tumpang tindih dan nilai yang lebih besar menghasilkan lebih banyak tumpang tindih. Jika jumlah titik data cukup kecil, kita dapat menambahkannya ke dalam ridge plot menggunakan kode berikut:

```
p + geom_density_ridges(jittered_points = TRUE)
```

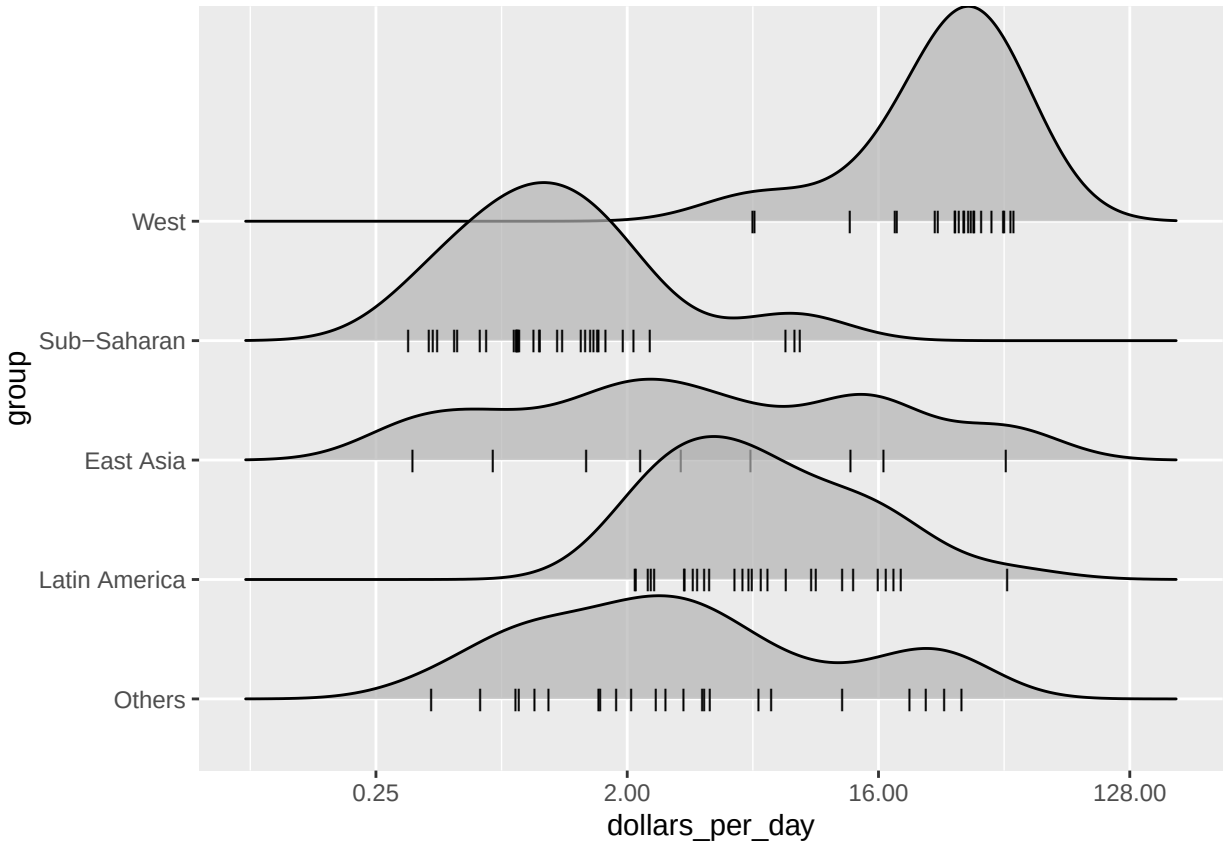
```
## Picking joint bandwidth of 0.648
```




Secara bawaan, tinggi titik data diberi jitter dan tidak boleh diinterpretasikan dengan cara apa pun. Untuk menampilkan titik data tetapi tanpa menggunakan jitter, kita dapat menggunakan kode berikut untuk menambahkan apa yang disebut sebagai representasi rug dari data.

```
p + geom_density_ridges(jittered_points = TRUE,
position = position_points_jitter(height = 0),
point_shape ='|'
, point_size = 3,
point_alpha = 1, alpha = 0.7)
```

```
## Picking joint bandwidth of 0.648
```




### 1970 versus 2010 income distributions

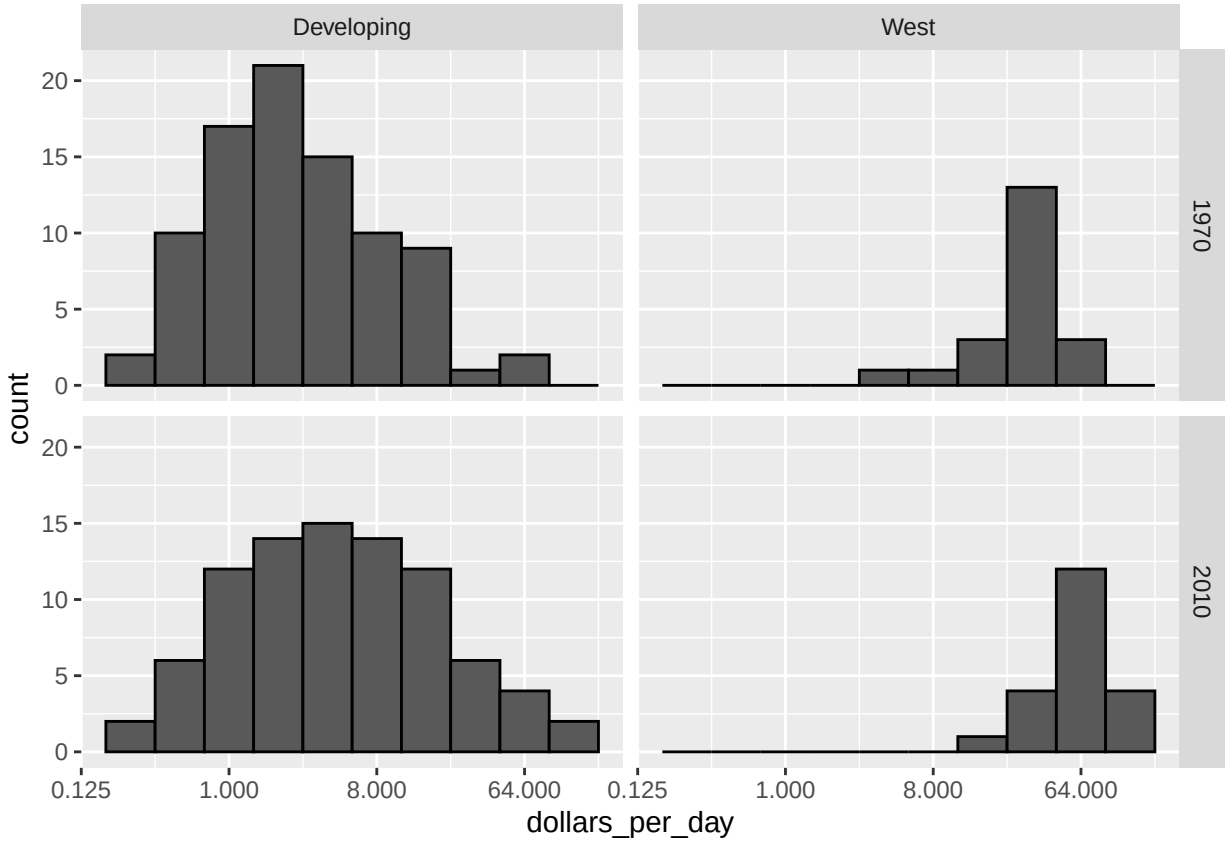
Eksplorasi data dengan jelas menunjukkan bahwa pada tahun 1970 terdapat dikotomi 'barat versus sisanya'. Namun, apakah dikotomi ini masih bertahan? Mari kita gunakan facet_grid dan melihat bagaimana distribusi tersebut telah berubah. Untuk memulai, kita akan fokus pada dua kelompok: barat dan sisanya. Kita membuat empat histogram. Plot ini hanya dibuat untuk negara-negara yang memiliki data pada tahun 1970 dan 2010. Perlu dicatat bahwa beberapa negara baru berdiri setelah tahun 1970; misalnya, Uni Soviet terpecah menjadi beberapa negara selama tahun 1990-an. Kita juga mencatat bahwa data tersedia untuk lebih banyak negara pada tahun 2010. Oleh karena itu, kita membuat plot hanya untuk negara-negara yang memiliki data pada kedua tahun tersebut.

```
past_year <- 1970
present_year <- 2010
years <- c(past_year, present_year)
country_list <- gapminder |>
filter(year %in% c(present_year, past_year)) |>
group_by(country) |>
summarize(n = sum(!is.na(dollars_per_day)), .groups = "drop") |>
filter(n == 2) |>
pull(country)
```

Ke-108 negara ini mencakup 86% dari populasi dunia, sehingga subset ini seharusnya representatif. Kita dapat membandingkan distribusi menggunakan kode berikut:

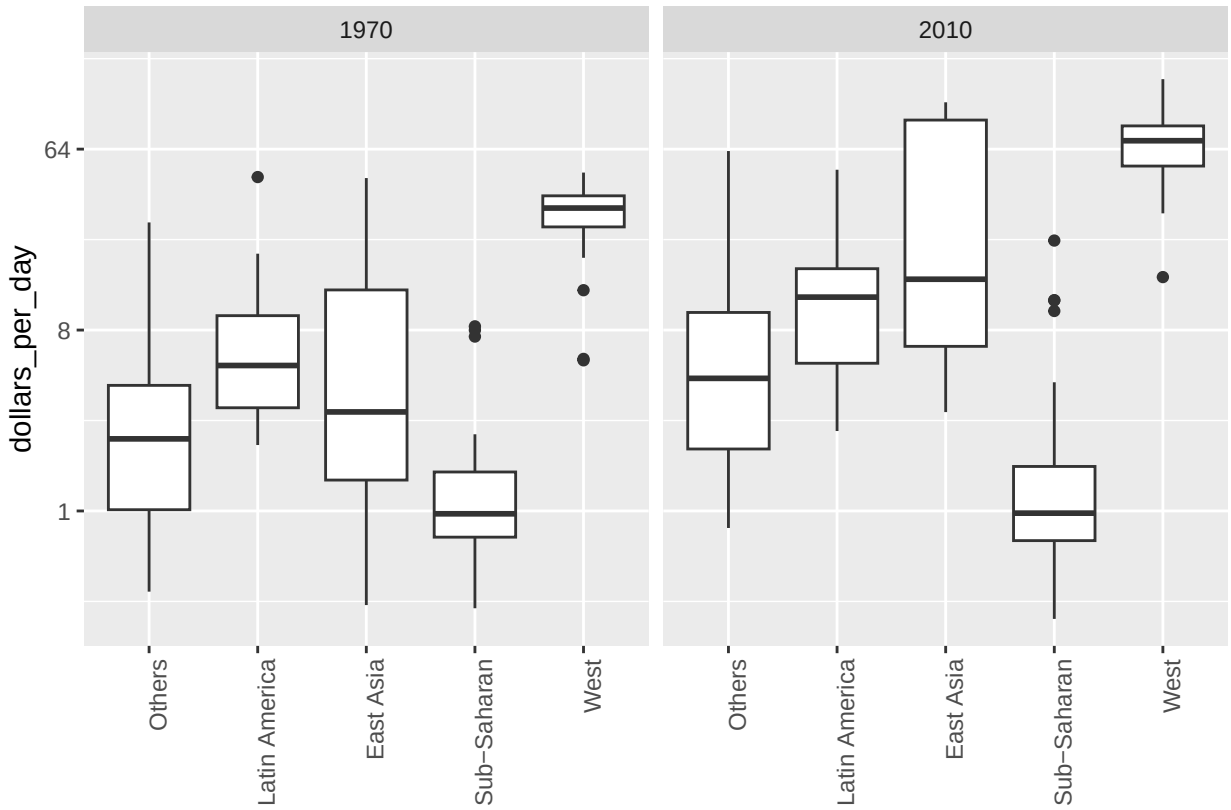
```
gapminder |>
filter(year %in% years & country %in% country_list) |>
mutate(west = ifelse(group == "West", "West", "Developing")) |>
ggplot(aes(dollars_per_day)) +
```

```
geom_histogram(binwidth = 1, color = "black") +
scale_x_continuous(trans = "log2") +
facet_grid(year ~ west)
```



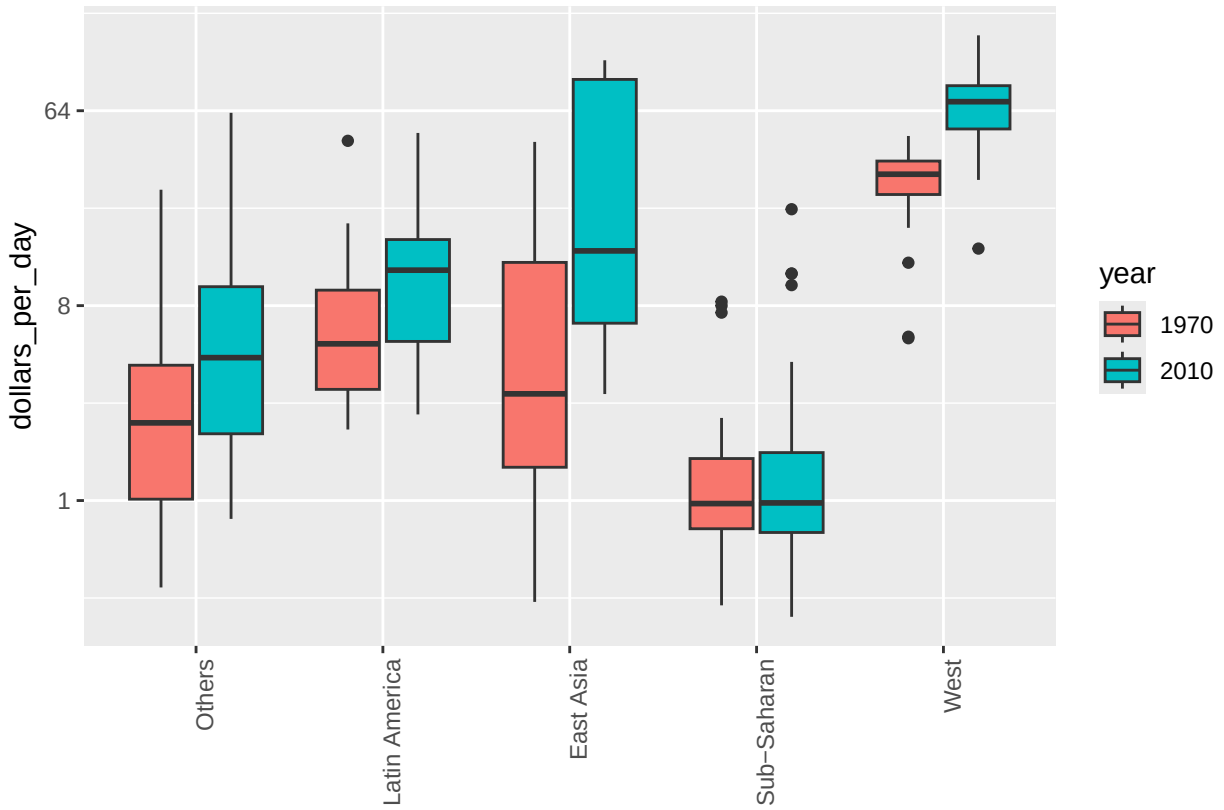

Kita sekarang melihat bahwa negara-negara kaya menjadi sedikit lebih kaya, tetapi secara persentase, negara-negara miskin tampaknya lebih banyak mengalami peningkatan. Secara khusus, kita melihat bahwa proporsi negara berkembang yang berpenghasilan lebih dari $16 per hari meningkat secara signifikan. Untuk melihat wilayah spesifik mana yang mengalami peningkatan terbesar, kita dapat membuat ulang boxplot yang telah dibuat sebelumnya, tetapi sekarang menambahkan tahun 2010 dan kemudian menggunakan *facet* untuk membandingkan kedua tahun tersebut.

```
gapminder |>
filter(year %in% years & country %in% country_list) |>
ggplot(aes(group, dollars_per_day)) +
geom_boxplot() +
theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
scale_y_continuous(trans = "log2") +
xlab("") +
facet_grid(. ~ year)
```



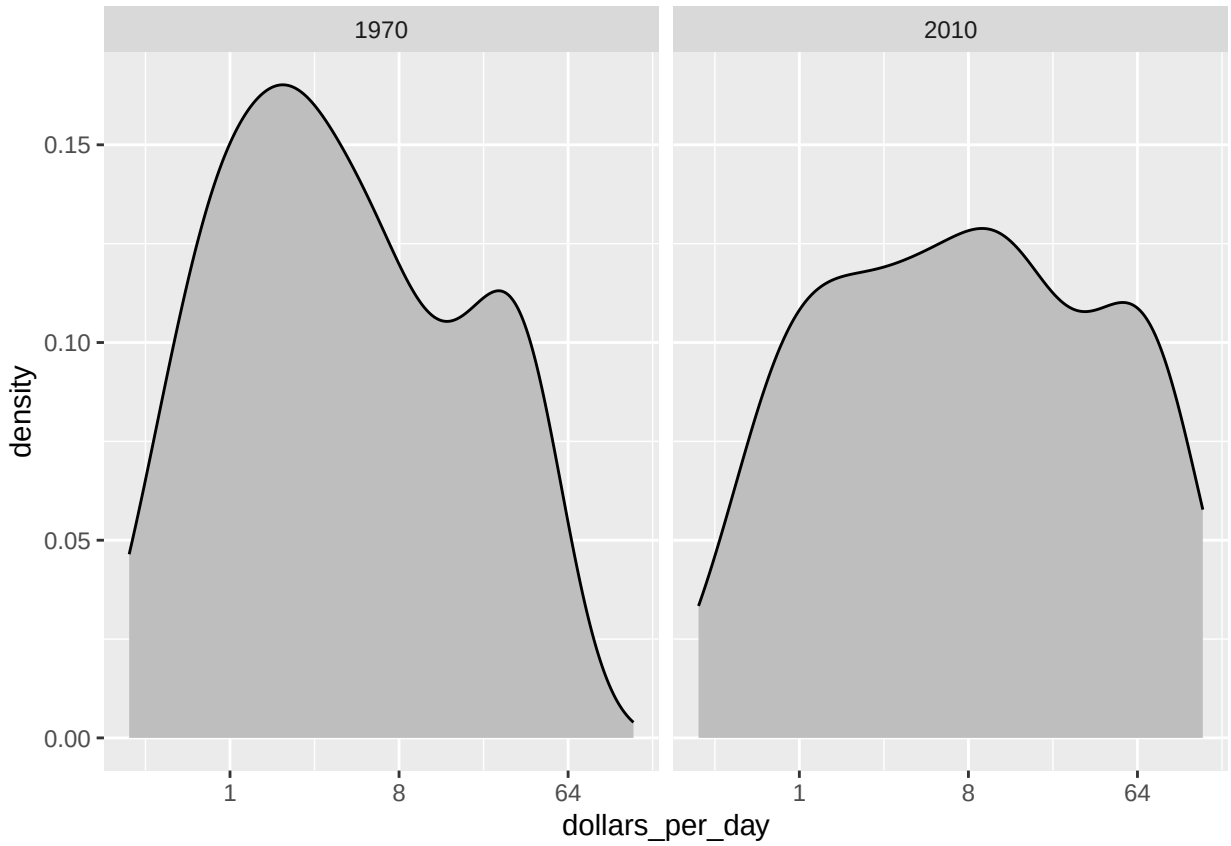

Karena kita ingin membandingkan setiap wilayah sebelum dan sesudah, akan lebih praktis jika boxplot tahun 1970 ditempatkan di sebelah boxplot tahun 2010 untuk setiap wilayah. Secara umum, perbandingan akan lebih mudah ketika data diplot bersebelahan. Jadi, alih-alih menggunakan *faceting*, kita menyatukan data dari setiap tahun dan meminta agar data diberi warna (atau *fill*) sesuai dengan tahun. Perlu dicatat bahwa kelompok secara otomatis dipisahkan berdasarkan tahun dan setiap pasangan boxplot digambar bersebelahan. Karena kolom tahun berupa angka, kita harus mengonversinya menjadi *factor* karena ggplot2 secara otomatis menetapkan warna pada setiap kategori dari sebuah *factor*.

```
gapminder |>
filter(year %in% years & country %in% country_list) |>
mutate(year = factor(year)) |>
ggplot(aes(group, dollars_per_day, fill = year)) +
geom_boxplot() +
theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
scale_y_continuous(trans = "log2") +
xlab("")
```




Eksplorasi data sebelumnya menunjukkan bahwa kesenjangan pendapatan antara negara kaya dan miskin telah menyempit secara signifikan selama 40 tahun terakhir. Kita menggunakan serangkaian histogram dan boxplot untuk melihat hal ini. Kami menyarankan cara yang lebih ringkas untuk menyampaikan pesan ini hanya dengan satu plot. Mari mulai dengan mencatat bahwa plot densitas untuk distribusi pendapatan pada tahun 1970 dan 2010 menyampaikan pesan bahwa kesenjangan tersebut semakin mengecil:

```
gapminder |>
filter(year %in% years & country %in% country_list) |>
ggplot(aes(dollars_per_day)) +
geom_density(fill = "grey") +
scale_x_continuous(trans = "log2") +
facet_grid(. ~ year)
```




Pada plot tahun 1970, kita melihat dua mode yang jelas: negara miskin dan negara kaya. Pada tahun 2010, tampak bahwa beberapa negara miskin telah bergeser ke kanan, mempersempit kesenjangan. Pesan berikutnya yang perlu kita sampaikan adalah bahwa alasan perubahan distribusi ini adalah karena beberapa negara miskin menjadi lebih kaya, bukan karena beberapa negara kaya menjadi lebih miskin. Untuk melakukan ini, kita dapat memberikan warna pada kelompok yang telah kita identifikasi selama eksplorasi data. Namun, karena ketika kita menumpangkan dua distribusi densitas, secara bawaan luas di bawah kurva distribusi dijumlahkan menjadi 1 untuk setiap kelompok, tanpa memperhatikan ukuran masing-masing kelompok, maka kita terlebih dahulu perlu mempelajari cara membuat distribusi densitas halus dengan cara yang tetap mempertahankan informasi tentang jumlah negara di setiap kelompok. Untuk itu, kita perlu mempelajari cara mengakses variabel yang dihitung dengan fungsi `geom_density`.

### Mengakses variabel yang dihitung

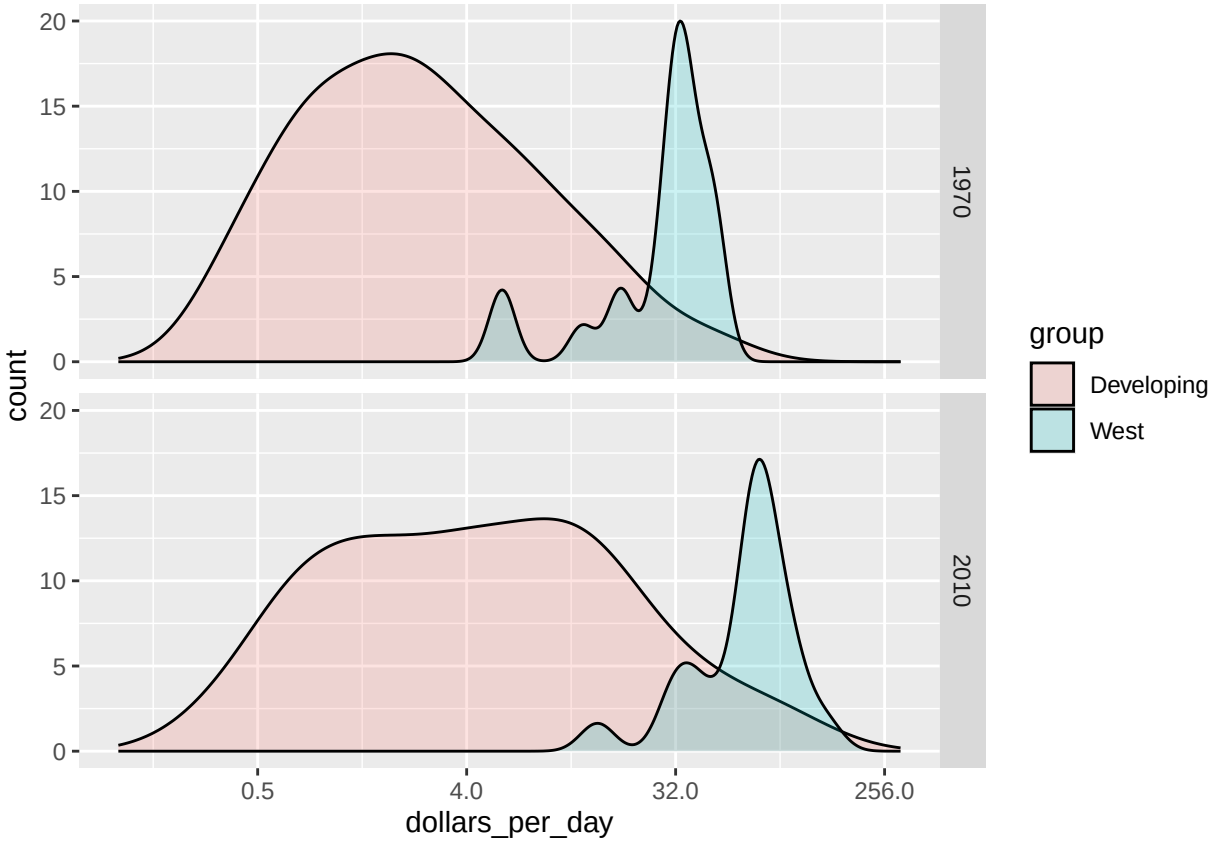
Agar luas dari distribusi densitas ini proporsional dengan ukuran kelompok, kita dapat dengan mudah mengalikan nilai pada sumbu-y dengan ukuran kelompok. Dari *help file* `geom_density`, kita melihat bahwa fungsi tersebut menghitung sebuah variabel bernama `count` yang melakukan hal ini secara langsung. Kita ingin variabel ini berada pada sumbu-y, bukan densitas. Dalam ggplot2, kita mengakses variabel-variabel ini menggunakan fungsi `after_stat`. Oleh karena itu, kita akan menggunakan pemetaan berikut:

```
aes(x = dollars_per_day, y = after_stat(count))
```

```
## Aesthetic mapping:
## * `x` -> `dollars_per_day`
## * `y` -> `after_stat(count)`
```

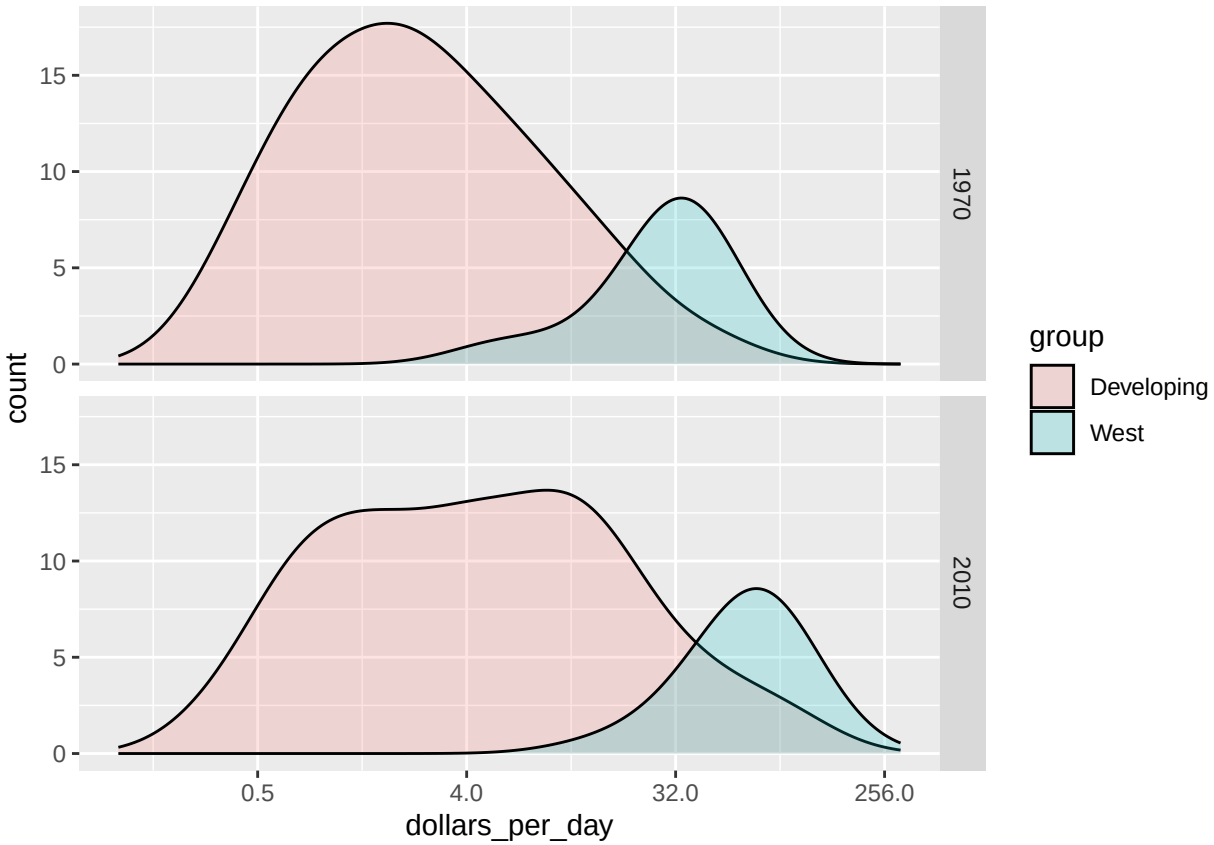
Kita sekarang dapat membuat plot yang diinginkan hanya dengan mengubah pemetaan pada potongan kode sebelumnya. Kita juga akan memperluas batas pada sumbu-x.

```
p <- gapminder |>
filter(year %in% years & country %in% country_list) |>
mutate(group = ifelse(group == "West", "West", "Developing")) |>
ggplot(aes(dollars_per_day, y = after_stat(count), fill = group)) +
scale_x_continuous(trans = "log2", limits = c(0.125, 300))
p + geom_density(alpha = 0.2) + facet_grid(year ~ .)
```



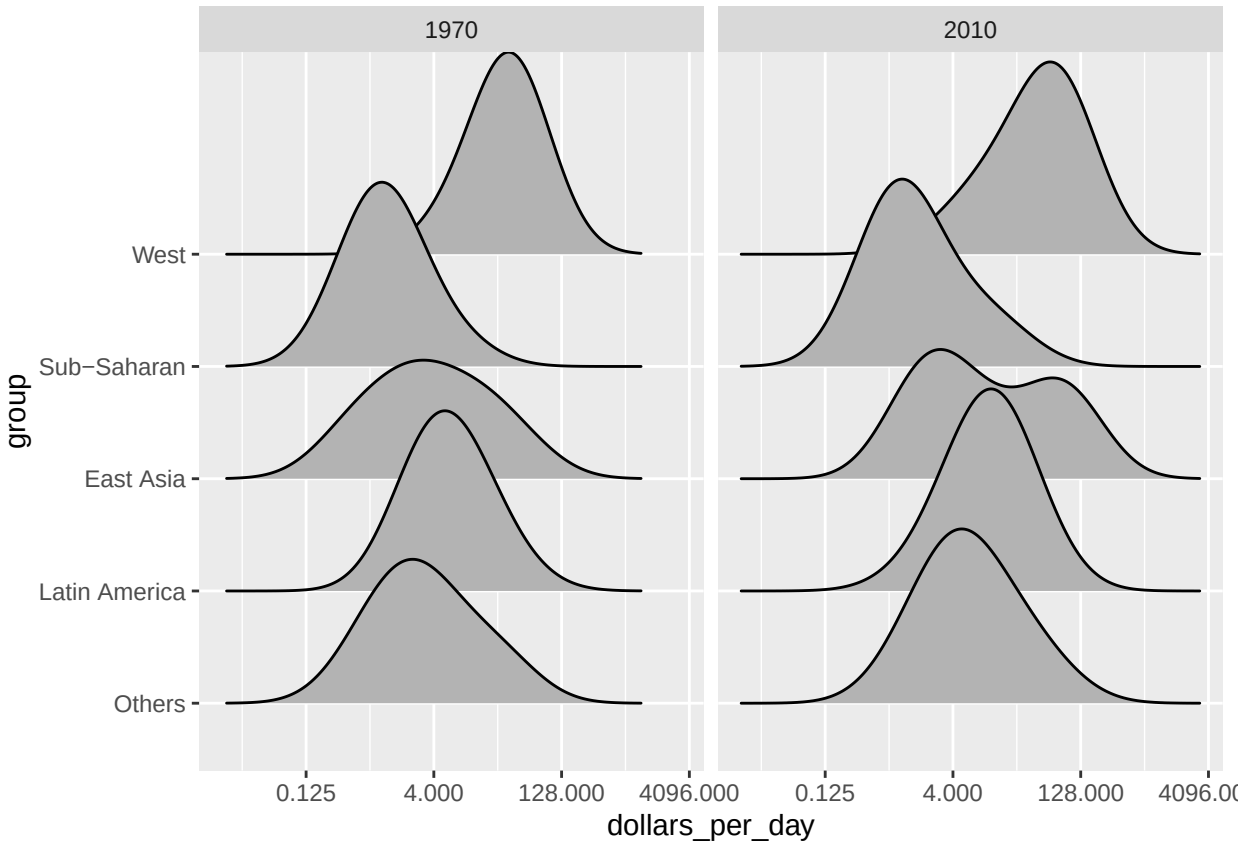

Jika kita ingin distribusi densitas menjadi lebih halus, kita menggunakan argumen `bw` sehingga bandwidth yang sama digunakan pada setiap distribusi densitas. Kami memilih 0.75 setelah mencoba beberapa nilai.

```
p + geom_density(alpha = 0.2, bw = 0.75) + facet_grid(year ~ .)
```



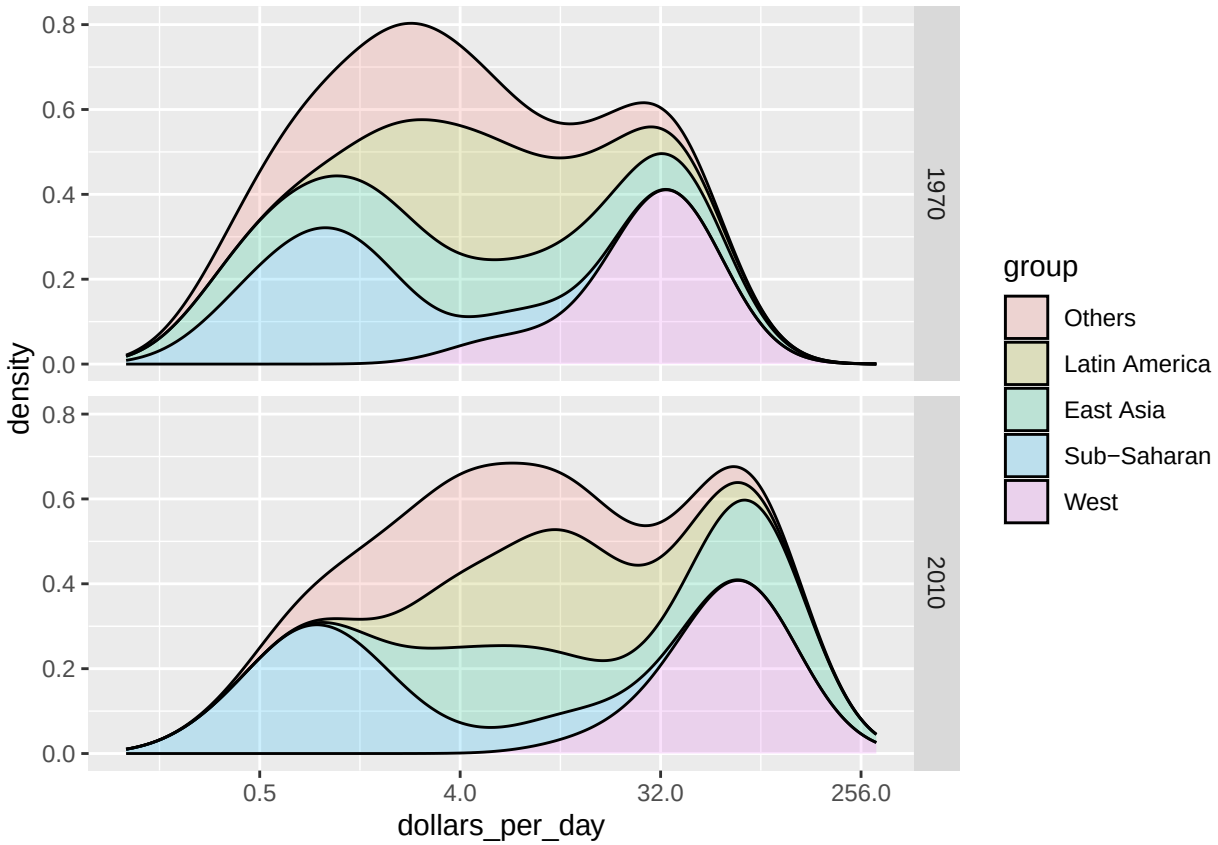

Plot ini sekarang menunjukkan dengan sangat jelas apa yang sedang terjadi. Distribusi dunia berkembang sedang berubah. Muncul mode ketiga yang terdiri dari negara-negara yang paling mempersempit kesenjangan. Untuk memvisualisasikan apakah ada kelompok yang didefinisikan di atas yang mendorong hal ini, kita dapat dengan cepat membuat ridge plot:

```
gapminder |>
filter(year %in% years & !is.na(dollars_per_day)) |>
ggplot(aes(dollars_per_day, group)) +
scale_x_continuous(trans = "log2") +
geom_density_ridges(bandwidth = 1.5) +
facet_grid(. ~ year)
```




Cara lain untuk mencapai hal ini adalah dengan menumpuk distribusi densitas satu di atas yang lain:

```
gapminder |>
filter(year %in% years & country %in% country_list) |>
group_by(year) |>
mutate(weight = population/sum(population)*2) |>
ungroup() |>
ggplot(aes(dollars_per_day, fill = group)) +
scale_x_continuous(trans = "log2", limits = c(0.125, 300)) +
geom_density(alpha = 0.2, bw = 0.75, position = "stack") +
facet_grid(year ~ .)
```




Di sini kita dapat melihat dengan jelas bagaimana distribusi untuk Asia Timur, Amerika Latin, dan wilayah lainnya bergeser secara signifikan ke kanan sementara Sub-Sahara Afrika tetap stagnan. Perhatikan bahwa kita mengurutkan level kelompok sehingga distribusi Barat dipetakan terlebih dahulu, kemudian Sub-Sahara Afrika. Dengan memetakan kedua ekstrem tersebut terlebih dahulu, kita dapat melihat bimodalitas yang tersisa dengan lebih baik.

## vaccines and infectious diseases

Vaksin telah membantu menyelamatkan jutaan nyawa. Pada abad ke-19, sebelum tercapainya kekebalan kelompok melalui program vaksinasi, kematian akibat penyakit menular seperti cacar dan polio sangat umum terjadi. Namun saat ini, program vaksinasi menjadi agak kontroversial meskipun ada banyak bukti ilmiah yang menunjukkan pentingnya. Kontroversi ini dimulai dengan sebuah makalah yang diterbitkan pada tahun 1988 dan dipimpin oleh Andrew Wakefield yang mengklaim adanya hubungan antara pemberian vaksin campak, gondok, dan rubella (MMR) dengan munculnya autisme dan penyakit usus. Meskipun terdapat banyak bukti ilmiah yang membantah temuan ini, laporan media yang sensasional dan penyebaran ketakutan oleh para penganut teori konspirasi membuat sebagian masyarakat percaya bahwa vaksin berbahaya. Akibatnya, banyak orang tua berhenti memvaksinasi anak-anak mereka. Praktik berbahaya ini berpotensi menimbulkan bencana, mengingat Pusat Pengendalian dan Pencegahan Penyakit (CDC) memperkirakan bahwa vaksinasi dapat mencegah lebih dari 21 juta rawat inap dan 732.000 kematian pada anak-anak yang lahir dalam 20 tahun terakhir (lihat *Benefits from Immunization during the Vaccines for Children Program Era — United States, 1994-2013, MMWR*). Makalah tahun 1988 tersebut sejak itu telah ditarik kembali dan Andrew Wakefield akhirnya 'dikeluarkan dari daftar medis Inggris, dengan pernyataan yang mengidentifikasi adanya pemalsuan yang disengaja dalam penelitian yang diterbitkan di *The Lancet*, dan dengan demikian dilarang untuk berpraktik kedokteran di Inggris.' (sumber: Wikipedia). Namun, kesalahpahaman masih tetap ada, sebagian karena aktivis yang mengaku dirinya ahli terus menyebarkan informasi yang salah tentang vaksin. Komunikasi data yang efektif adalah penangkal yang kuat terhadap misinformasi dan penyebaran ketakutan. Dalam pengantar bagian buku ini, kami menunjukkan sebuah contoh yang disajikan oleh artikel *Wall Street Journal*, yang menampilkan data terkait dampak vaksin terhadap penanggulangan penyakit menular. Di sini,

kami merekonstruksi contoh tersebut.

**Data vaksin** Data yang digunakan untuk plot ini dikumpulkan, diorganisir, dan didistribusikan oleh Tycho Project. Data tersebut mencakup jumlah laporan mingguan untuk tujuh penyakit dari tahun 1928 hingga 2011, dari seluruh lima puluh negara bagian. Kami menyertakan total tahunan dalam paket dslabs:

```
library(tidyverse)
library(RColorBrewer)
library(dslabs)
names(us_contagious_diseases)
```

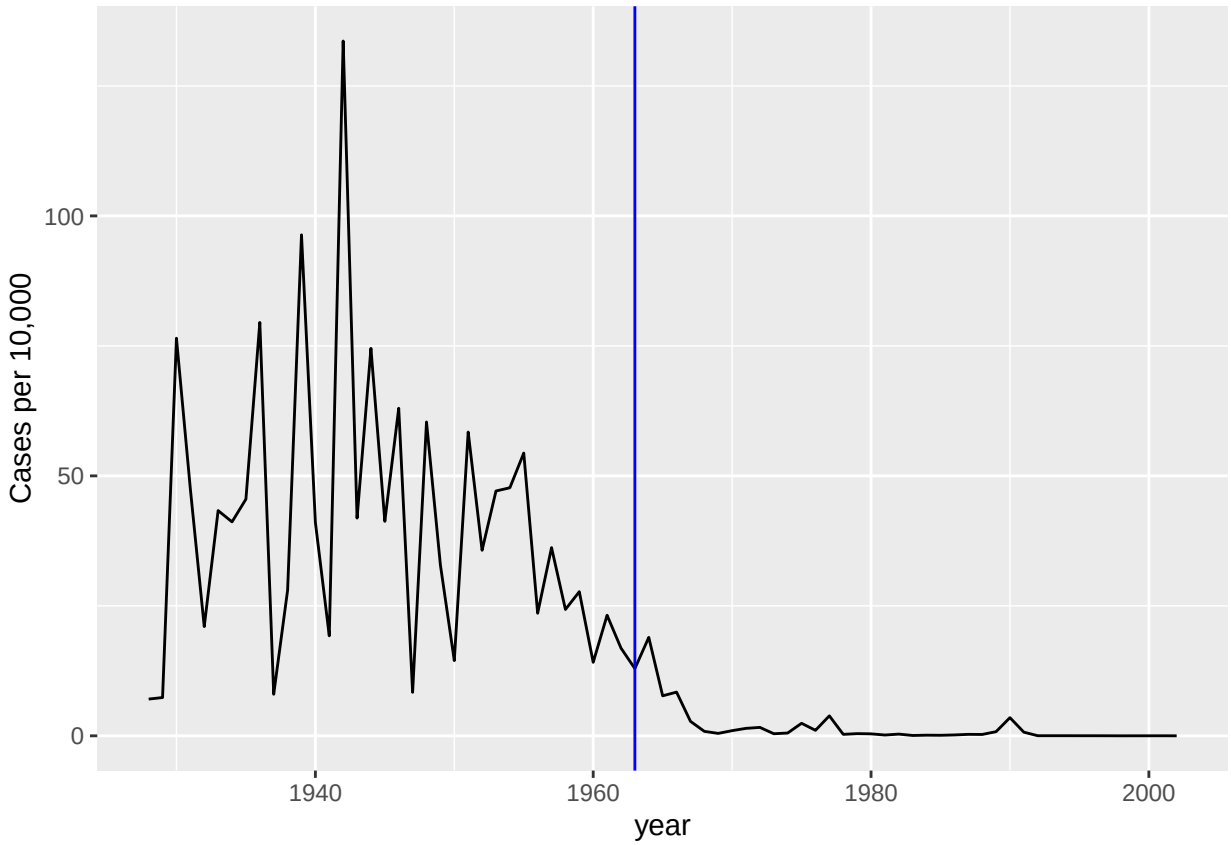
```
## [1] "disease"         "state"           "year"            "weeks_reporting"
## [5] "count"           "population"
```

Kami membuat sebuah objek sementara bernama dat yang hanya menyimpan data campak, menyertakan laju per 100.000, mengurutkan negara bagian berdasarkan nilai rata-rata penyakit, dan menghapus Alaska serta Hawaii karena keduanya baru menjadi negara bagian pada akhir 1950-an. Perlu dicatat bahwa terdapat kolom weeks_reporting yang memberi tahu berapa minggu dalam setahun data dilaporkan. Kita harus menyesuaikan nilai tersebut saat menghitung laju.

```
the_disease <- "Measles"
dat <- us_contagious_diseases |>
filter(!state %in% c("Hawaii","Alaska") & disease == the_disease) |>
mutate(rate = count / population * 10000 * 52 / weeks_reporting) |>
mutate(state = reorder(state, ifelse(year <= 1963, rate, NA),
median, na.rm = TRUE))
```

**Trend plot** Sekarang kita dapat dengan mudah memplot laju penyakit per tahun. Berikut adalah data campak dari California:

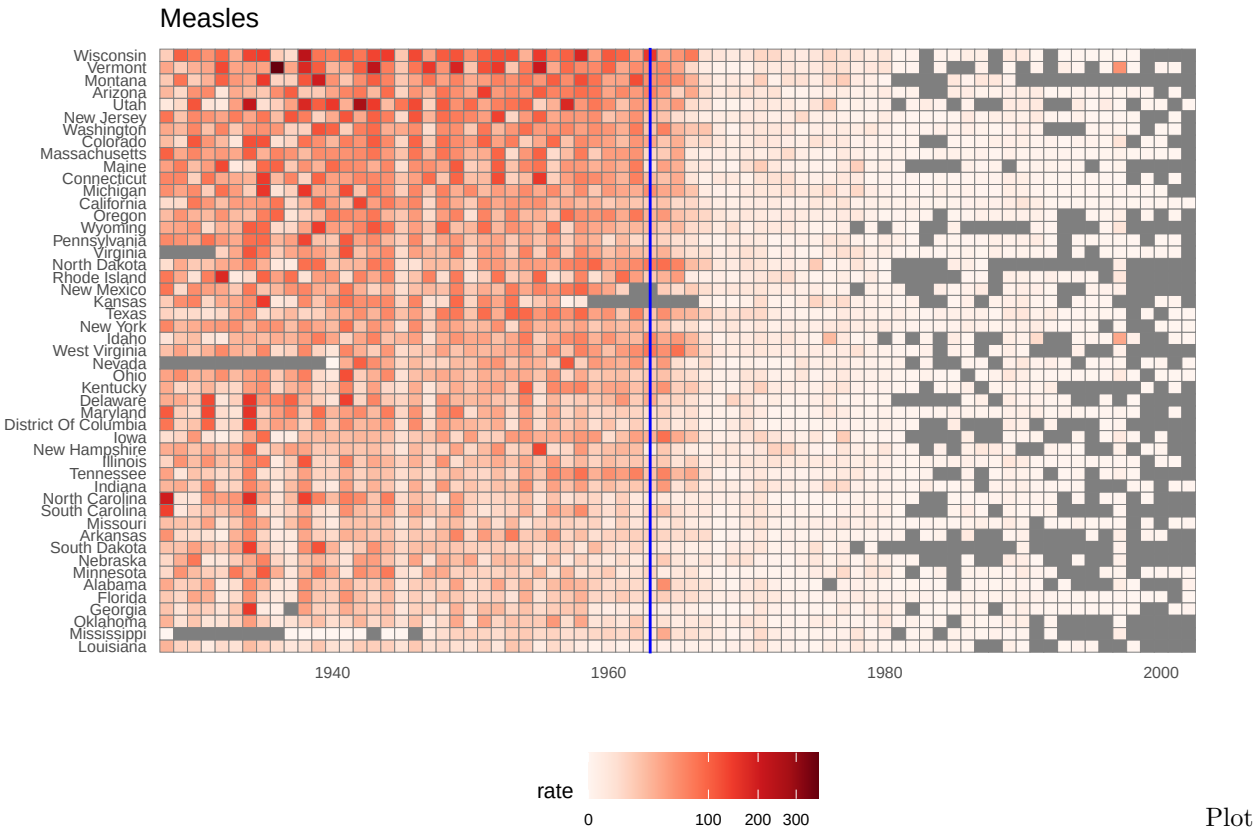
```
dat |> filter(state == "California" & !is.na(rate)) |>
ggplot(aes(year, rate)) +
geom_line() +
ylab("Cases per 10,000") +
geom_vline(xintercept = 1963, col = "blue")
```




Kita tambahkan sebuah garis vertikal pada tahun 1963 karena pada tahun inilah vaksin diperkenalkan.

**Heatmaps** Sekarang, bisakah kita menampilkan data untuk semua negara bagian dalam satu plot? Kita memiliki tiga variabel yang akan ditampilkan: tahun, negara bagian, dan tingkat (rate). Pada gambar dari WSJ, mereka menggunakan sumbu-x untuk tahun, sumbu-y untuk negara bagian, dan gradasi warna untuk merepresentasikan tingkat penyakit. Namun, skala warna yang mereka gunakan — dari kuning ke biru ke hijau ke oranye ke merah — masih bisa ditingkatkan. Dalam contoh kita, kita ingin menggunakan palet sekuensial karena tidak ada pusat yang bermakna, hanya tingkat rendah dan tinggi. Kita menggunakan geometri geom_tile untuk membuat ubin (tile) dengan warna yang merepresentasikan tingkat penyakit. Kita juga menggunakan transformasi akar kuadrat (square root) untuk menghindari dominasi dari jumlah yang sangat tinggi. Perhatikan bahwa nilai yang hilang ditampilkan dalam warna abu-abu. Catat juga bahwa setelah suatu penyakit hampir sepenuhnya diberantas, beberapa negara bagian berhenti melaporkan kasus sama sekali. Inilah sebabnya kita melihat begitu banyak warna abu-abu setelah tahun 1980.

```
dat |> ggplot(aes(year, state, fill = rate)) +
geom_tile(color = "grey50") +
scale_x_continuous(expand = c(0,0)) +
scale_fill_gradientn(colors = brewer.pal(9, "Reds"), trans = "sqrt") +
geom_vline(xintercept = 1963, col = "blue") +
theme_minimal() +
theme(panel.grid = element_blank(),
legend.position = "bottom",
text = element_text(size = 8)) +
labs(title = the_disease, x = ""
, y = "")
```

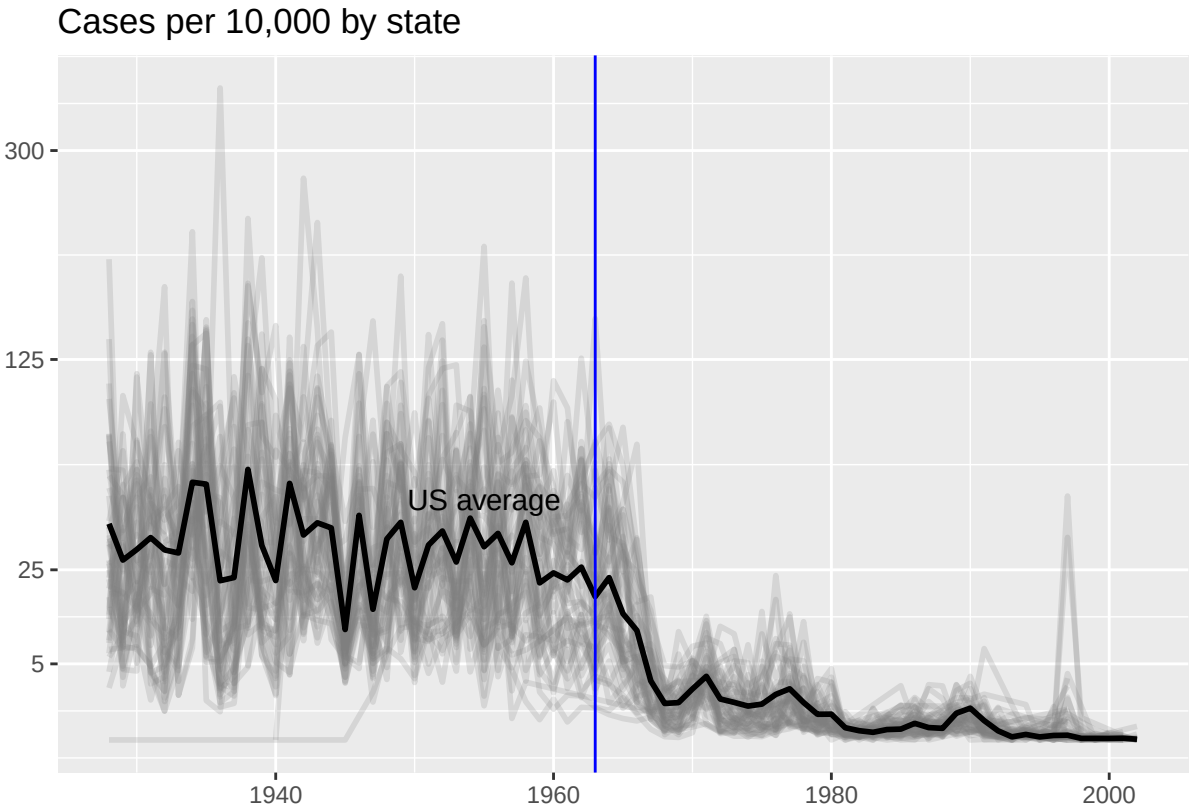



Plot ini memberikan argumen yang sangat kuat mengenai kontribusi vaksin. Namun, ada satu keterbatasan dari plot ini, yaitu penggunaan warna untuk merepresentasikan kuantitas. Seperti yang sudah kita jelaskan sebelumnya, hal ini membuat kita lebih sulit mengetahui dengan tepat seberapa tinggi nilai tersebut. Posisi dan panjang merupakan petunjuk yang lebih baik. Jika kita bersedia kehilangan informasi terkait negara bagian, kita bisa membuat versi plot yang menampilkan nilai menggunakan posisi. Kita juga dapat menampilkan rata-rata untuk AS, yang kita hitung seperti ini:

```
avg <- us_contagious_diseases |>
filter(disease == the_disease) |> group_by(year) |>
summarize(us_rate = sum(count, na.rm = TRUE) /
sum(population, na.rm = TRUE) * 10000)
```

Sekarang, untuk membuat plotnya kita cukup menggunakan geometri geom_line:

```
dat |>
filter(!is.na(rate)) |>
ggplot() +
geom_line(aes(year, rate, group = state), color = "grey50",
show.legend = FALSE, alpha = 0.2, linewidth = 1) +
geom_line(mapping = aes(year, us_rate), data = avg, linewidth = 1) +
scale_y_continuous(trans = "sqrt", breaks = c(5, 25, 125, 300)) +
ggtitle("Cases per 10,000 by state") +
xlab("") + ylab("") +
geom_text(data = data.frame(x = 1955, y = 50),
mapping = aes(x, y, label = "US average"),
color = "black") +
geom_vline(xintercept = 1963, col = "blue")
```




Secara teori, kita bisa menggunakan warna untuk merepresentasikan nilai kategorikal state, tetapi sulit untuk memilih 50 warna yang berbeda.