

MODUL 2

GIT, GITHUB, R PROJECT, DAN R MARKDOWN

A. Tujuan Praktikum

- Mengimplementasikan penggunaan Git dan GitHub pada RStudio
- Mengimplementasikan langkah-langkah membuat *project* baru dengan R Project
- Mengimplementasikan penggunaan R Markdown

B. Alokasi Waktu

1 x pertemuan = 120 menit

C. Dasar Teori

Proyek analisis data biasanya akan melibatkan beberapa *dataset* dan *script*. Oleh karena itu, pengorganisasian *dataset-dataset* dan *script-script* yang digunakan dalam suatu proyek analisis data merupakan tantangan tersendiri. Proses analisis data juga dilakukan secara berulang dan adaptif. Akibatnya, dimungkinkan akan terjadi beberapa kali proses *editing script* atau pembaruan versi *script*. Dalam modul ini, kita akan mencoba menggunakan sistem kontrol versi (*versioning control*) dengan menggunakan Git, yang merupakan *tools* bantuan untuk melacak perubahan-perubahan atau pembaruan-pembaruan *script* yang dilakukan khususnya dalam proyek analisis data. Selain Git, kita juga akan menggunakan GitHub, yang merupakan layanan yang memungkinkan kita untuk berbagi-pakai kode dengan pengguna GitHub lainnya. Salah satu tujuan penggunaan Git dan GitHub adalah untuk memfasilitasi kolaborasi dengan antar pengguna.

Selanjutnya kita akan mempelajari langkah-langkah membuat *project* baru pada RStudio, dan belajar menulis laporan dengan menggunakan R Markdown. Dengan menggunakan R Markdown, kita dapat membuat laporan yang berisi teks dan kode sekaligus dalam satu dokumen.

Mengapa menggunakan Git dan GitHub?

Ada tiga alasan dasar utama yang mendorong kita untuk menggunakan Git dan GitHub.

1. **Berbagi:** Meskipun kita tidak memanfaatkan fungsionalitas kontrol versi pada *script* atau *project* yang kita miliki, kita masih bisa menggunakan Git dan GitHub untuk membagikan *script* yang kita buat. Pada modul ini kita akan mencoba menggunakan Git dan GitHub pada RStudio.
2. **Berkolaborasi:** Jika telah memiliki dan mengatur *repository* GitHub yang digunakan, beberapa *user* dapat membuat perubahan pada *script* yang terdapat pada *repository* tersebut dengan tetap memperhatikan *versioning control*. GitHub menyediakan layanan gratis untuk *repository* terpusat. GitHub juga memiliki utilitas khusus, seperti *fork* dan *clone*. Mem-*fork* berarti menyalin sebuah *project*, mengubah namanya, serta membuat sebuah *project* dan komunitas baru dengan salinan yang telah dibuat.
3. **Kontrol versi:** Kemampuan *versioning control* pada Git memungkinkan kita untuk melacak perubahan yang kita lakukan pada *script*. Selain itu, kita juga dapat mengembalikan *script* ke versi file sebelumnya. Git juga memiliki fitur *branches* untuk menguji terlebih dahulu ide perubahan *script* yang telah kita lakukan sebelum memutuskan apakah perubahan *script* akan di *merge* pada *script* yang asli.

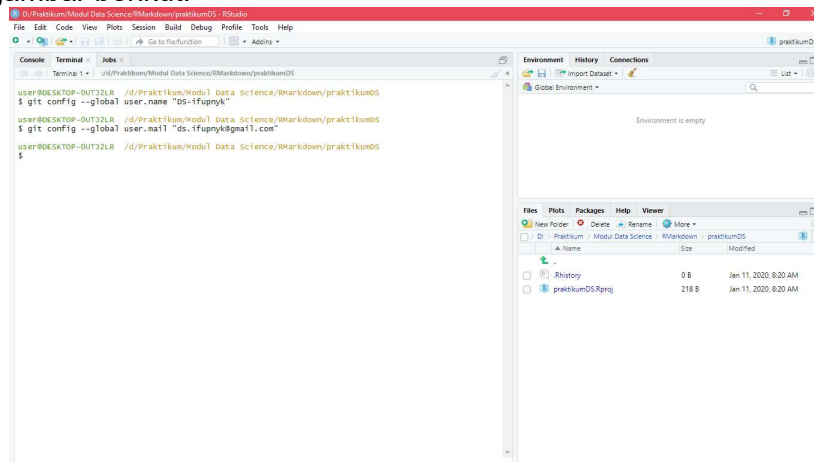
Git, dan GitHub pada RStudio

Setelah menginstal Git, langkah pertama adalah membuat akun GitHub. Jika telah memiliki akun GitHub, Anda dapat langsung menuju ke langkah selanjutnya, yaitu menghubungkan Git dan RStudio ke akun GitHub yang telah dimiliki.

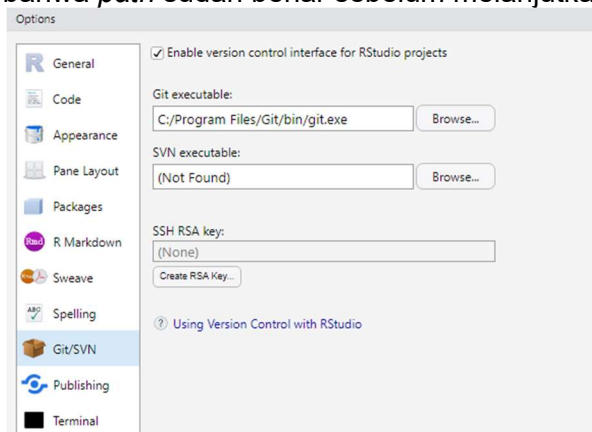
Konfigurasi Git pada RStudio dapat dilakukan pada RStudio *terminal* dengan menggunakan dua perintah berikut:

```
git config --global user.name "Your Name"
git config --global user.mail "your@email.com"
```

Pastikan bahwa email yang dimasukkan pada perintah diatas sama dengan akun email yang digunakan pada GitHub. Setelah memasukkan perintah diatas, tampilan sesi RStudio akan terlihat seperti gambar berikut:

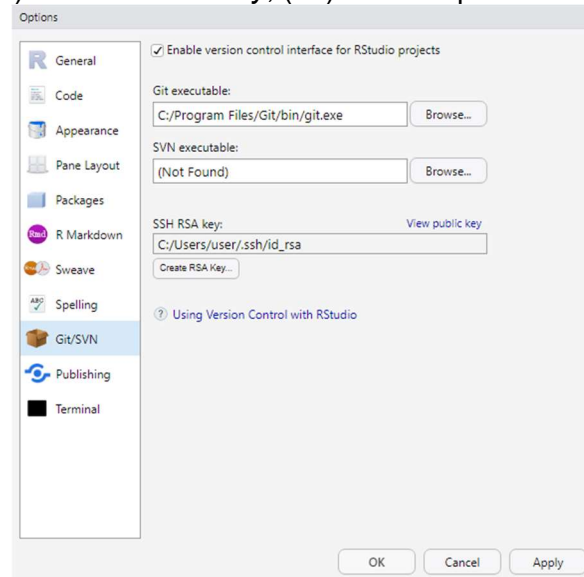


Selanjutnya, pilih Tools – Global Option - Git / SVN, aktifkan opsi *version control interface for RStudio projects*. Kemudian memasukkan *path* yang digunakan untuk eksekusi Git yang telah kita instal. Pada instalasi *default* di *Windows*, *path* nya adalah: *C:/Program File/Git/bin/git.exe*. Pastikan terlebih dahulu bahwa *path* sudah benar sebelum melanjutkan ke langkah berikutnya.



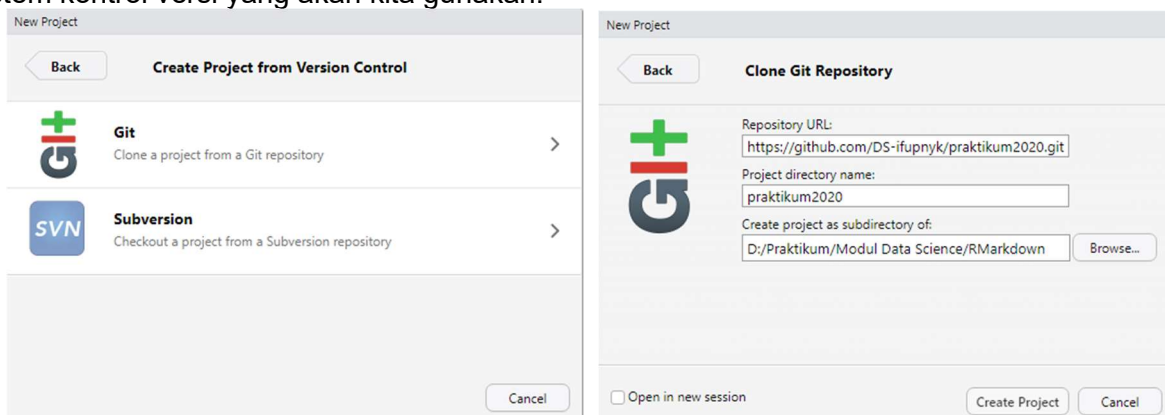
Agar GitHub tidak meminta kata sandi setiap kali kita melakukan akses terhadap repositori, Kita dapat memilih opsi *Create RSA Key* pada *window GIT/SVN*. Pengaturan Key SSH RSA dapat dilakukan dengan langkah-langkah berikut : (1) Klik pada tombol *Create RSA Key*, (2) Pilih *Create* tanpa memasukkan frasa sandi apa pun, (3) Klik *Close* pada *window* baru yang muncul, (4) Klik pada *View Public Key*, (5) *Copy* key yang ditampilkan pada *window*, (6) *Login* ke Akun Github Anda dan Buka *Settings*, (7) Pilih *SSH and GPG keys* pada menu *Personal Settings* di sebelah

kiri, (8) Klik pada *button: new SSH keys* di bagian kanan atas, (9) Berikan Judul dan *Paste SSH key* pada bagian *Key*, (10) Klik *Add SSH Key*, (11) Kembali pada RStudio, klik *Apply* - OK:



Setelah melakukan langkah-langkah diatas, Git, RStudio, dan GitHub telah terhubung. Selanjutnya kita dapat membuat repositori GitHub baru, melakukan *fork* atau *clone* pada repositori yang telah ada.

Pada bagian ini, kita akan memulai proyek RStudio dengan menggunakan *version control* dan menyimpan kode/*script* yang telah tersedia pada *repository* GitHub. Untuk mulai membuat R *project* baru, File – New Project, kemudian pilih Version Control dan selanjutnya, pilih Git sebagai sistem kontrol versi yang akan kita gunakan:

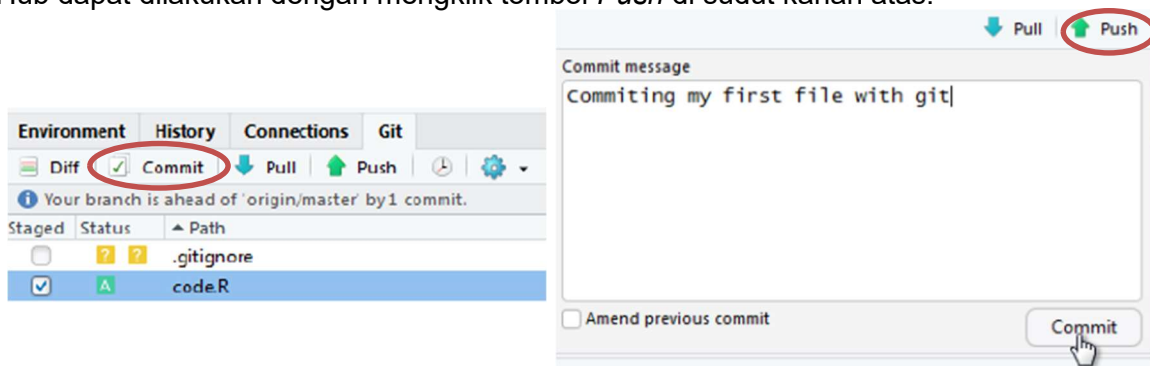


Repository URL adalah alamat tautan yang akan dikloning. Sebelum memasukkan *Repository* URL, silahkan buka *repository* berikut <https://github.com/DS-ifupnyk/praktikum2020.git> dan klik *fork* untuk menambahkan *repository* tersebut pada akun GitHub yang anda miliki. Kemudian buka akun GitHub anda, dan salin *repository* URL yang berisi hasil *fork* pada langkah sebelumnya. Dalam *Project Directory Name*, kita dapat memilih nama direktori proyek yang kita inginkan, namun secara *default* *Project Directory Name* akan muncul secara otomatis setelah *Repository* URL dimasukkan. Setelah menentukan *Project Directory Name*, folder baru dengan nama *praktikum2020* akan ditambahkan di direktori kerja sistem lokal kita yang telah didefinisikan pada bagian *Create project as subdirectory of*. Jika koneksi ke *repository* GitHub berhasil dibuat, tampilan dari sesi R akan berubah seperti contoh gambar selanjutnya. Pada sudut kanan atas,

akan muncul *Project Directory Name* sesuai dengan nama direktori pada *repository* GitHub yang di *clone*, serta muncul *tab* baru di panel kanan atas , yaitu: Git.



Selanjutnya, kita akan membahas mengenai panel Git. File lokal dan *repository* GitHub tidak melakukan sinkronisasi secara otomatis. Untuk menyimpan semua perubahan yang telah dilakukan pada *project* di file lokal, sinkronisasi dapat dilakukan dengan menggunakan perintah `git push` pada terminal RStudio. Untuk melakukan sinkronisasi dengan *repository* GitHub, pilih *stage box* sehingga warna ikon status berubah menjadi hijau, kemudian klik tombol *commit*. Dengan Git, setiap kali melakukan *commit*, kita akan diminta untuk memasukkan komentar yang menjelaskan mengenai perubahan yang dilakukan. Setelah melakukan *commit*, kita akan melihat pesan dari Git dengan ringkasan perubahan yang dilakukan. Menyimpan perubahan di *repository* GitHub dapat dilakukan dengan mengklik tombol *Push* di sudut kanan atas:

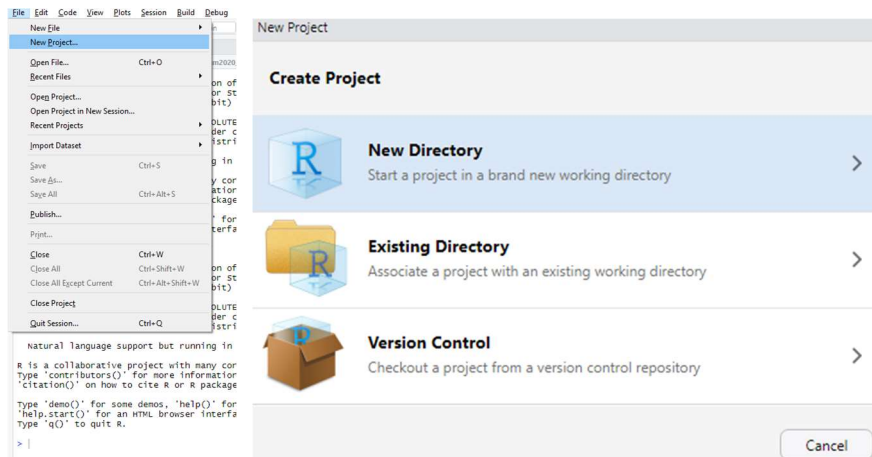


Untuk memastikan perubahan yang dilakukan telah berhasil di-*push* pada *repository* GitHub, silahkan kunjungi *repository* GitHub Anda.

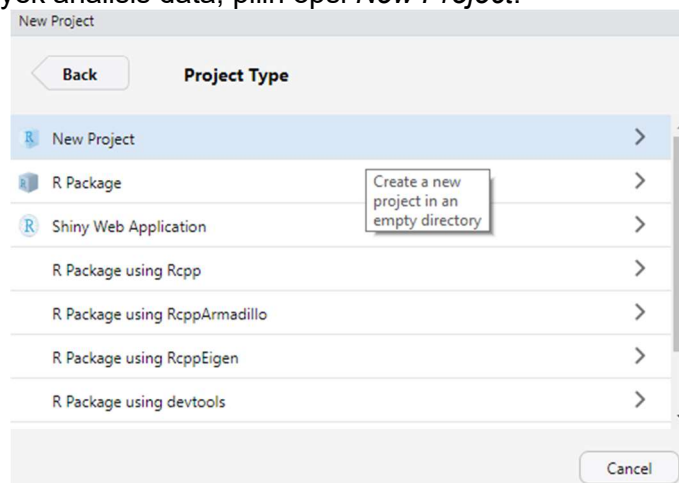
RStudio projects

RStudio memfasilitasi penyimpanan semua komponen proyek analisis data yang diatur dalam satu folder dengan ekstensi file `.Rproj`. Dalam bagian sebelumnya, kita telah mencoba fasilitas Git dan GitHub pada RStudio yang dapat digunakan untuk sinkronisasi dan kolaborasi proyek-proyek RStudio melalui *repository* GitHub. Di bagian ini, kita akan mencoba membuat/memulai proyek baru pada RStudio. Dengan menggunakan RStudio *projects* kita dapat memiliki beberapa sesi RStudio dan berpindah antar proyek dengan tetap terorganisir.

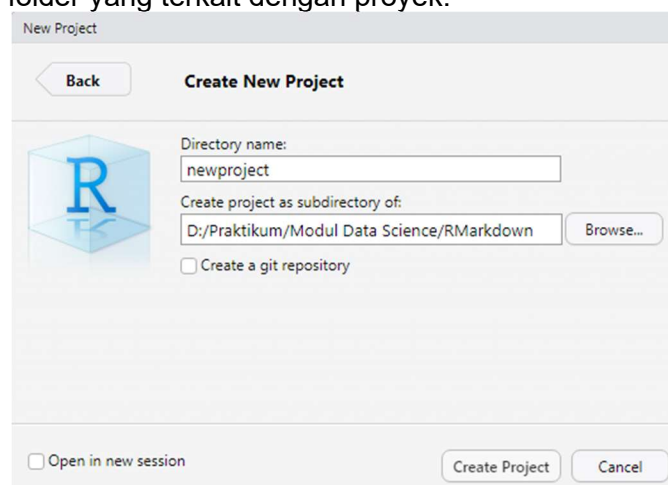
Untuk memulai proyek, klik File dan kemudian New Project. Di sini kita akan membuat folder baru dengan memilih opsi New Directory.



Kemudian, untuk proyek analisis data, pilih opsi *New Project*:



Selanjutnya, tentukan *Directory name* dan lokasi penyimpanan file .Rproj yang diinginkan pada *Create project as subdirectory of*. Saat memilih nama folder, sama halnya dengan nama file, pastikan bahwa nama yang digunakan adalah nama yang bermakna yang akan membantu Anda mengingat apa proyek tersebut. Langkah ini kemudian akan menghasilkan file .Rproj bernama newdirectory.Rproj di folder yang terkait dengan proyek.



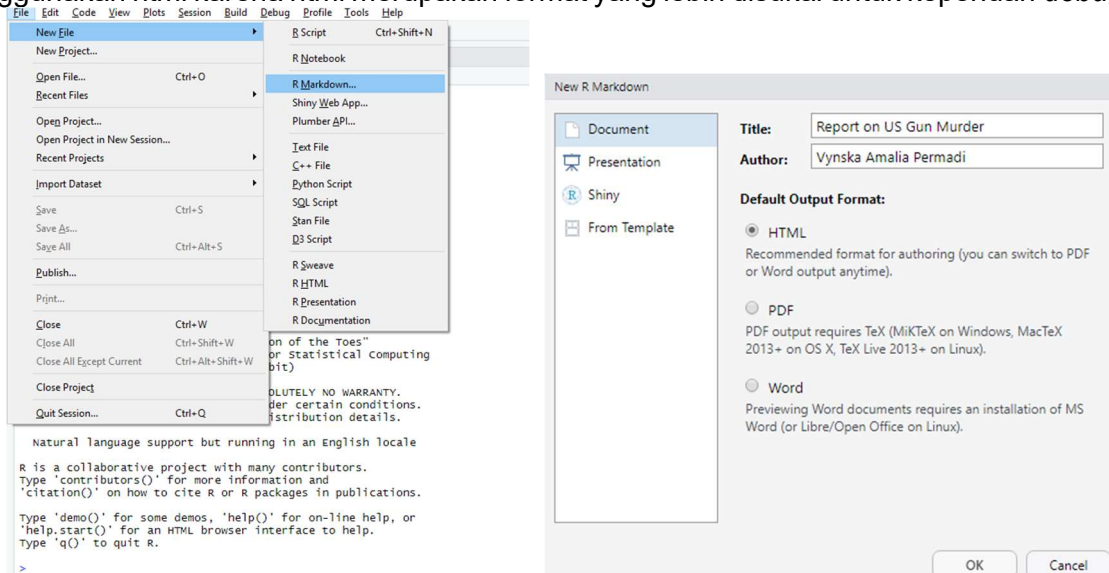
Ketika kita mulai menggunakan RStudio dengan proyek, nama proyek pada sesi yang ditampilkan dapat dilihat di sudut kiri atas. Ketika kita membuka sesi RStudio tanpa proyek, maka pada sudut kiri atas akan terlihat keterangan *Project: (None)*. Saat bekerja pada sesi suatu proyek, semua *file* atau *script* yang dibuat akan disimpan pada direktori folder yang ditentukan sebelumnya pada saat proses pembuatan *New Project*.

Salah satu keuntungan utama dalam penggunaan R *Projects* adalah setelah menutup RStudio, jika kita ingin melanjutkan proyek, buka direktori penyimpanan proyek yang telah ditentukan saat pertama kali membuat proyek RStudio, lalu klik dua kali pada nama *project* yang akan dilanjutkan. Keuntungan lain adalah, jika kita memilih untuk membuka dua atau lebih file *.Rproj* yang berbeda, maka secara otomatis RStudio akan memulai sesi R baru untuk masing-masing *project* yang kita pilih.

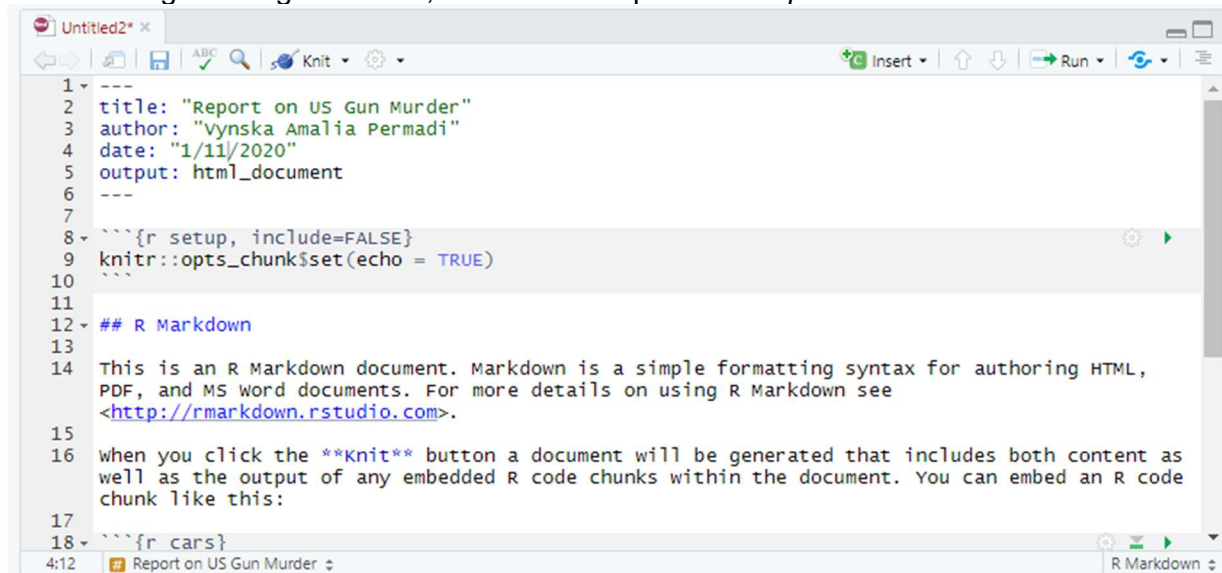
R Markdown

R Markdown adalah format untuk dokumen *literate programming* (paradigma *programming* yang memfasilitasi sebuah program komputer untuk memberikan penjelasan tentang logikanya dalam bahasa alami, seperti bahasa Inggris, diselingi dengan potongan makro dan *script*. Dimana *script* yang dituliskan dalam dokumen yang sama dapat *compile* dan menampilkan *output* yang dihasilkan). R Markdown merupakan turunan dari *markup language*, yaitu bahasa pemrograman yang banyak digunakan untuk menghasilkan halaman html. Tidak seperti menggunakan aplikasi pengolah kata seperti Microsoft Word, dengan R Markdown kita harus meng-*compile* dokumen untuk dapat menampilkan laporan akhir. Dokumen R Markdown yang kita buat akan terlihat berbeda dengan tampilan dokumen laporan yang dihasilkan. Misalnya, kita dapat memasukkan plot secara otomatis dengan mendefinisikan potongan *script* tertentu, dan tidak perlu memasukkannya satu per satu ke dalam dokumen pengolah kata.

Di RStudio, kita dapat memulai dokumen R Markdown dengan mengklik File-New File-R Markdown. Kemudian, kita akan diminta untuk memasukkan judul dan penulis untuk dokumen *Markdown* yang akan kita buat. Sebagai contoh, kita akan membuat dokumen R Markdown yang berisi laporan tentang kejadian pembunuhan senjata di US. Selanjutnya, kita juga dapat memutuskan format laporan akhir yang kita inginkan: HTML, PDF, atau Microsoft Word. Format laporan akhir dapat diubah-ubah sesuai dengan kebutuhan. Tetapi, di sini kita akan menggunakan html karena html merupakan format yang lebih disukai untuk keperluan *debugging*:



Setelah mengikuti langkah diatas, kita akan memperoleh *template* dokumen R Markdown:

A screenshot of a text editor window titled 'Untitled2*'. The editor shows an R Markdown template. The header section (lines 1-6) is enclosed in three dashes and contains: title: "Report on US Gun Murder", author: "vynska Amalia Permadi", date: "1/11/2020", and output: html_document. Line 7 is another set of three dashes. Lines 8-10 are an R code chunk starting with {r setup, include=FALSE}, followed by knitr::opts_chunk\$set(echo = TRUE), and ending with }. Line 11 is a blank line. Line 12 is a section header ## R Markdown. Line 13 is a blank line. Line 14 contains a paragraph of text explaining R Markdown. Line 15 is a blank line. Line 16 contains another paragraph of text. Line 17 is a blank line. Line 18 is an R code chunk starting with {r cars}, followed by summary(cars), and ending with }. The status bar at the bottom shows '4:12' and 'Report on US Gun Murder'.

Sebelum melanjutkan, silahkan simpan dahulu dokumen yang telah dibuat. Dokumen R Markdown akan memiliki ekstensi .Rmd. *Template* yang dihasilkan telah memberikan kita kemudahan dalam memulai dokumen *Markdown* baru. Dari *template* tersebut, kita akan membahas lebih rinci mengenai beberapa hal yang perlu diperhatikan dalam membuat dokumen R Markdown.

1. Header

Di bagian atas kita akan menemukan *header*:

```
---
title: "Report on US Gun Murder"
author: "vynska Amalia Permadi"
date: "1/11/2020"
output: html_document
---
```

Semua *input* yang dituliskan di antara tanda (---) adalah *header*. Penulis *header* tidak diwajibkan pada dokumen R Markdown. Secara *default*, setelah kita mendefinisikan *tittle* dan *author* pada saat membuat dokumen baru, *header* akan berisi *tittle*, *author* dan *output format* yang telah kita masukkan sebelumnya. Kita juga dapat menambahkan beberapa informasi tambahan lain di area *header*. Salah satu parameter penting pada *header* adalah *output*. Pada bagian ini, kita dapat mengontrol tipe *output* yang ingin kita hasilkan setelah proses *compile* dilakukan. Tidak hanya dokumen, *markdown* juga dapat menghasilkan output lain seperti *Presentation*, dan *Shiny*.

2. R code chunks

Di beberapa bagian dokumen, kita akan melihat area seperti ini:

```
{r}
summary(cars)
}
```

Area tersebut selanjutnya akan kita sebut *chunk*. Ketika kita meng*compile* dokumen, kode R di dalam *chunk*, yang dalam *template* berisi perintah `summary(cars)`, akan dievaluasi dan hasil evaluasi perintah tersebut akan ditampilkan dalam posisi yang sama dalam dokumen akhir. Untuk menambahkan potongan *script* atau perintah R baru, kita dapat mengetikkan karakter *three backticks* seperti contoh diatas, dilanjutkan dengan mendefinisikan bahasa pemrograman apa

yang akan kita tuliskan pada perintah/*script*, dan diakhiri dengan *three backticks*. Sebagai alternatif, dapat pula digunakan *key binding*: Ctrl-Alt-I di Windows.

Secara *default*, perintah atau *script* yang kita tuliskan pada *chunk* juga akan ditampilkan pada dokumen akhir. Jika kita ingin menyembunyikan apa yang kita tuliskan pada *chunk*, maka kita akan menambahkan argumen `echo = FALSE`. Sebagai contoh:

```
```${r echo=FALSE}
summary(pressure)
```
```

Sebaiknya biasakan pula untuk menambahkan label ke potongan kode R. Hal ini akan sangat berguna saat men-*debug*. Contohnya, kita dapat menambahkan kata deskriptif seperti berikut:

```
```${r pressure-summary}
summary(pressure)
```
```

3. Global options

Dibawah *header*, kita akan menemui salah satu potongan *script* R berisi perintah yang tampak rumit:

```
```${r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```
```

Kita tidak akan membahas detail mengenai *global option* pada modul ini, tetapi jika Anda telah memiliki pengalaman dengan R Markdown, Anda akan dapat memahami keuntungan dari menetapkan opsi global untuk proses *compile*.

4. knitr

Kita akan menggunakan paket knitr untuk meng-*compile* dokumen R Markdown. Fungsi spesifik yang digunakan untuk proses *compile* adalah fungsi `knit`, yang mengambil nama file sebagai input. RStudio telah menyediakan tombol spesifik yang memudahkan untuk mengeksekusi fungsi `knit`. Pada contoh di bawah ini, dokumen *template* telah *diedit* sehingga kita dapat membuat laporan tentang kejadian pembunuhan bersenjata di US. Anda dapat melihat file di direktori yang telah anda *clone* dengan nama file: `report.Rmd`. Selanjutnya, klik tombol *Knit*:



Pertama kali Anda mengklik tombol *Knit*, sebuah kotak dialog mungkin muncul meminta untuk menginstal paket yang dibutuhkan. Setelah menginstal paket, Knit akan meng*compile* file R Markdown dan dokumen laporan yang dihasilkan akan muncul.

Kita juga dapat menghasilkan dokumen PDF atau Microsoft dengan mengubah:

`output: html_document` menjadi `output: pdf_document` atau `output: word_document`.

