R Packages: Write R Code

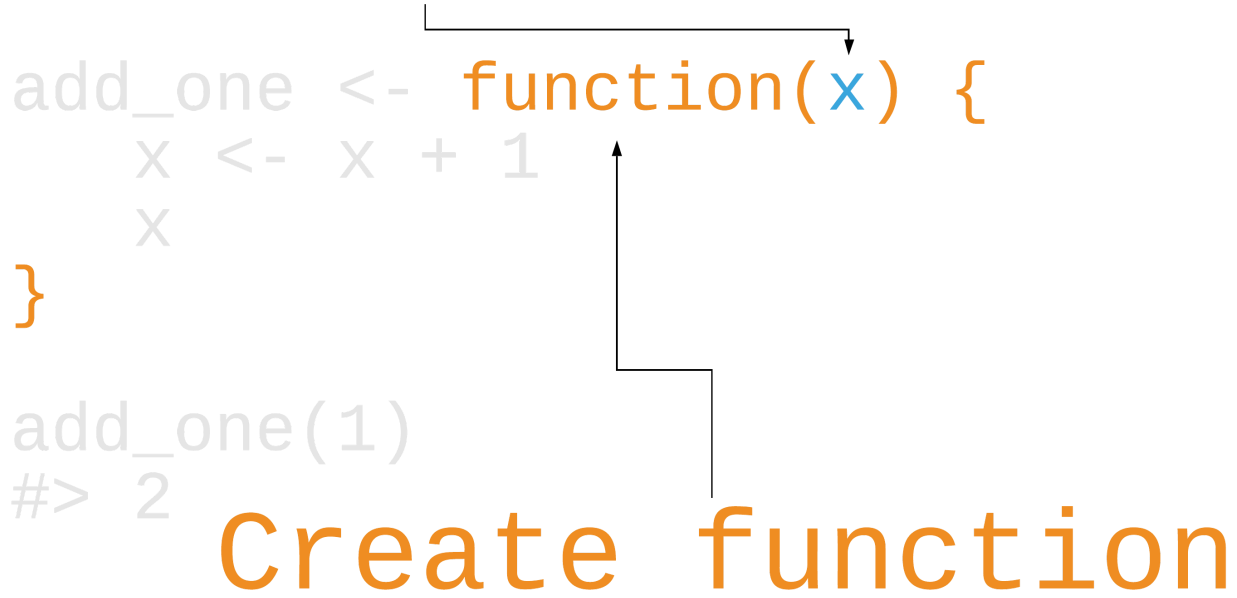
Writing Functions: Review

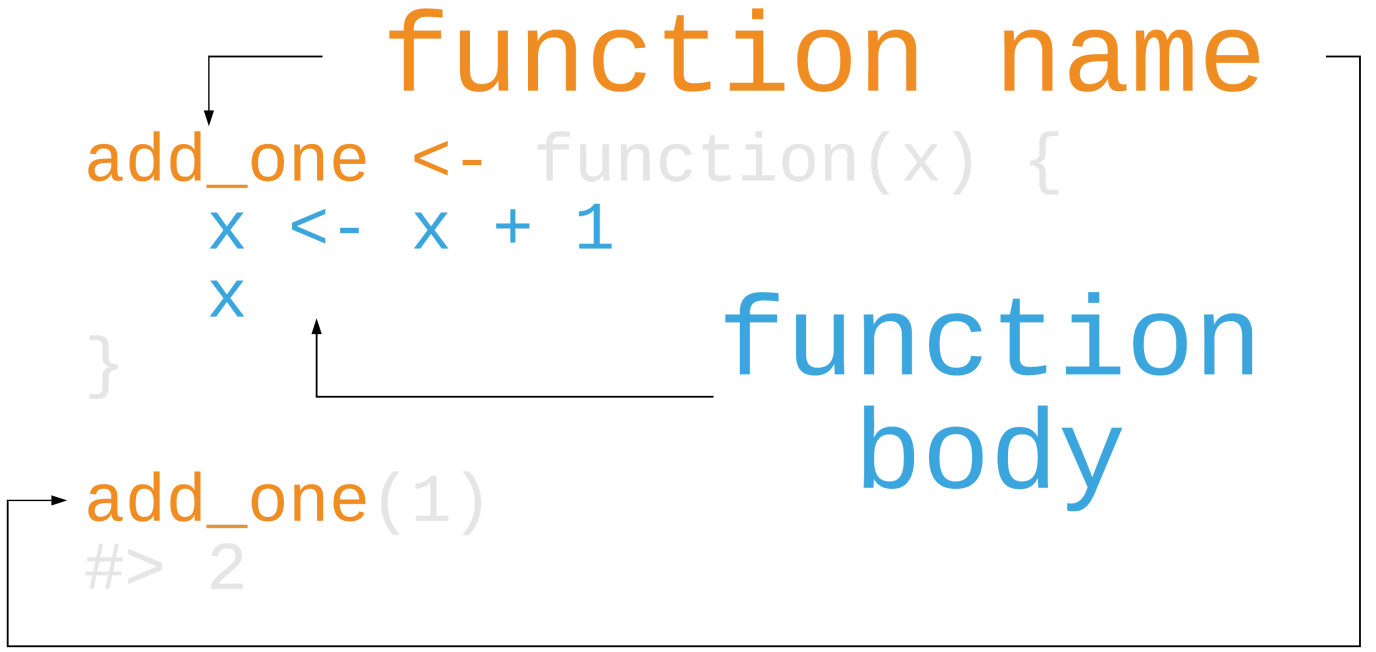
Writing functions

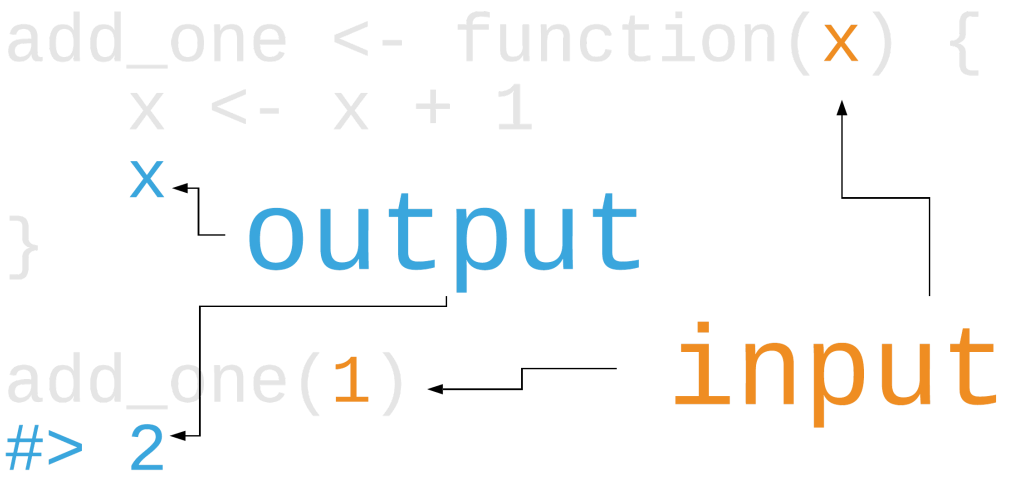
```
add_one <- function(x) {
    x <- x + 1
    x
}

add_one(1)
#> 2
```

Function arguments




function name
add_one <- function(x) {
x <- x + 1
x
}
function body
add_one(1)
#> 2


add_one <- function(x) {
x <- x + 1
x
}
output
input
add_one(1)
#> 2

Your Turn 1

Re-write this ggplot2 theme as a function. Call it `theme_avalanche_h()`.

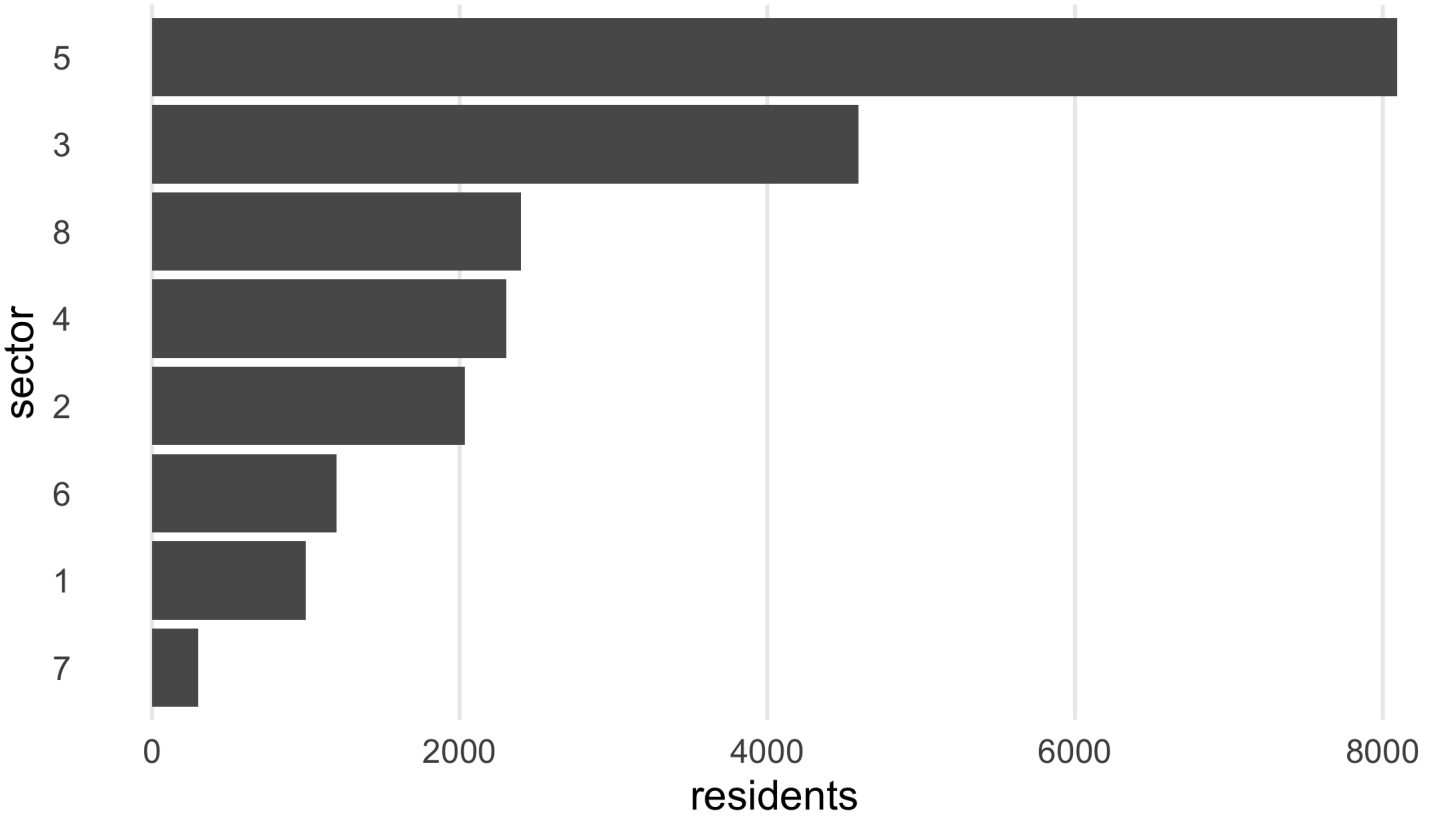
Run this code to test that your function works

Your Turn 1

```
theme_avalanche_h <- function() {
  theme_minimal(base_size = 14) +
    theme(panel.grid.minor = element_blank(), panel.grid.major.y = element_blank())
}

residents_per_sector <-
  data.frame(
    sector = as.factor(1:8),
    residents = c(1000, 2034, 4594, 2304, 8093, 1200, 300, 2398)
  )

ggplot(residents_per_sector, aes(forcats::fct_reorder(sector, residents), residents)) +
  geom_col() +
  coord_flip() +
  xlab("sector") +
  theme_avalanche_h()
```


sector
5
3
8
4
2
6
1
7
0
2000
4000
6000
8000
residents

We're going to start writing functions to our package, not the global environment. This will require a change in workflow!

```
use_r("file_name")
```

Write a new file to `R/`

```
use_r("themes")
```

```
shinRa
├── .Rbuildignore
├── .gitignore
├── DESCRIPTION
├── NAMESPACE
├── R/
│   └── themes.R
└── shinRa.Rproj
```

workflow alert

source()

load_all()
build()

devtools: loading vs. building

`load_all()`: fast, all functions available

`build()`: builds and installs the package

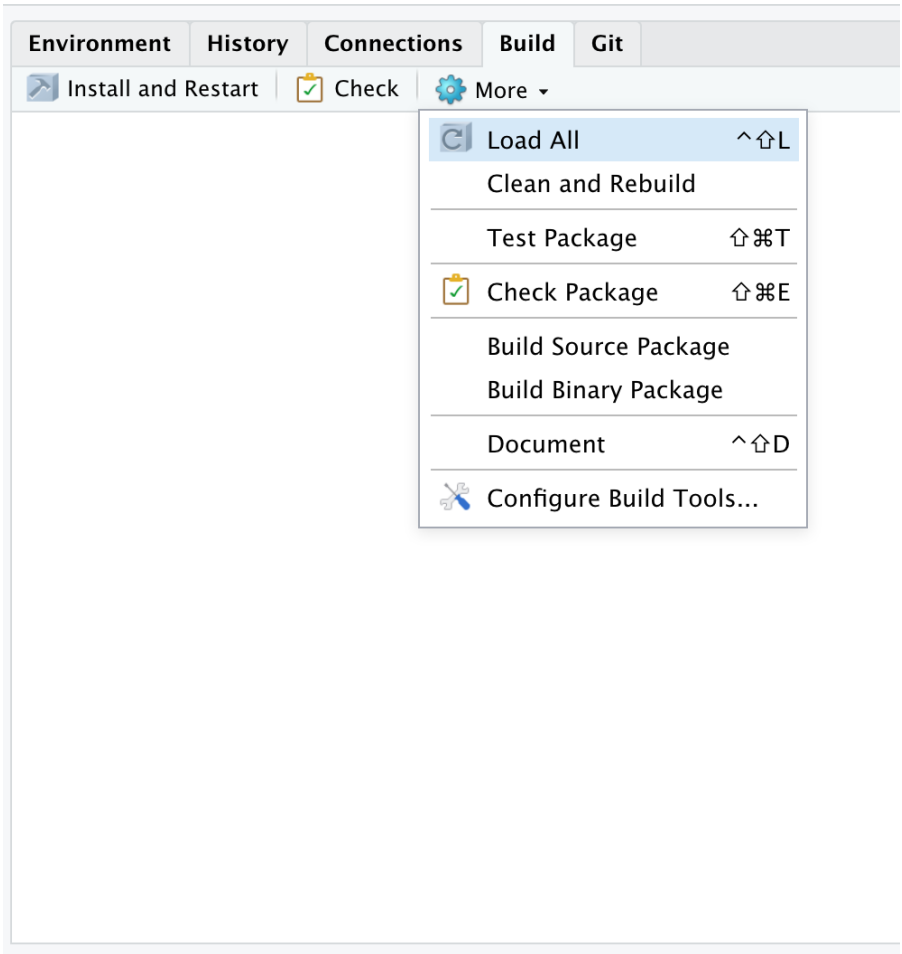
Keyboard shortcuts

`load_all()`:

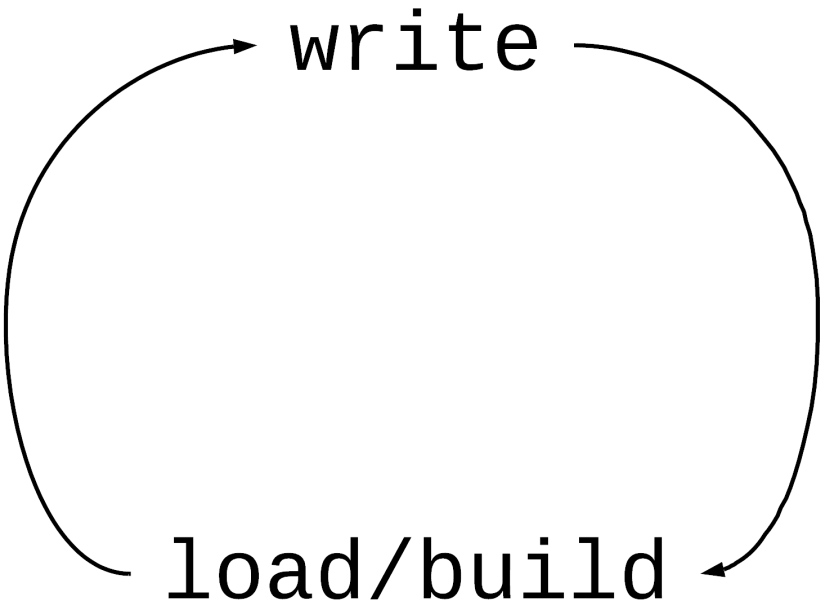
CMD/CTRL + Shift + L

`build()`:

CMD/CTRL + Shift + B


Environment
History
Connections
Build
Git
Install and Restart
Check
More
Load All ^⇧L
Clean and Rebuild
Test Package ⇧⌘T
Check Package ⇧⌘E
Build Source Package
Build Binary Package
Document ^⇧D
Configure Build Tools...

Our new workflow



Your Turn 2

Create a new file with `use_r()` called "db_con"

Put this function in the file and save it

Use `load_all()` to load the package function.

Run this code to make sure it works:

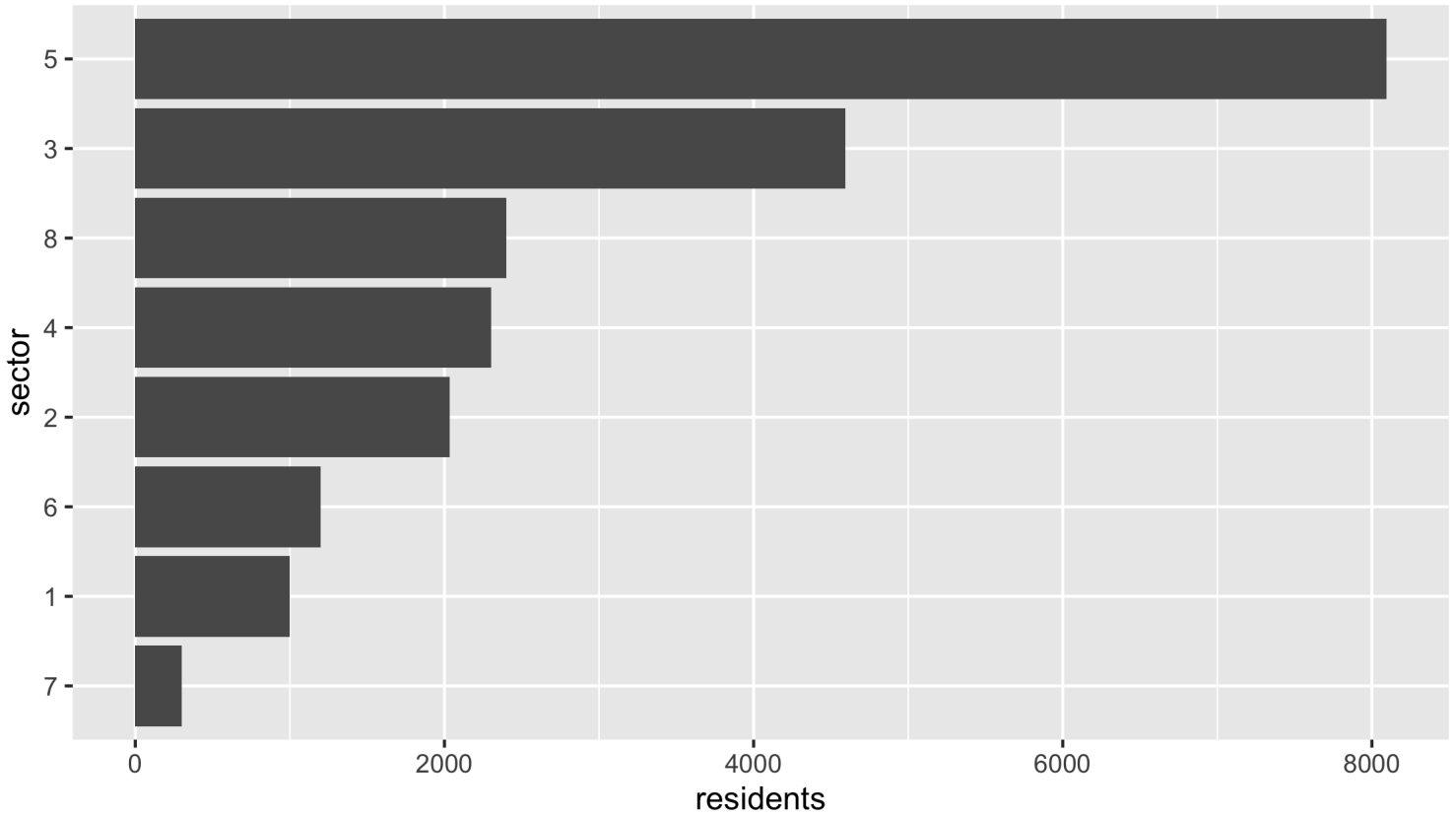
Your Turn 2

```
use_r("db_con")
```

in `R/db_con.R`:

```
db_con <- function(dbname = "residents_per_sector") {
  dbname <- match.arg(dbname)
  # We'll pretend we've connected to a database
  # and just return some hard-coded data instead.
  data.frame(
    sector = as.factor(1:8),
    residents = c(1000, 2034, 4594, 2304, 8093, 1200, 300, 2398)
  )
}
```

Your Turn 2



Using other packages

1. Import a package with `use_package()`
2. Use the package with `pkg::fun()`
3. **DO NOT** use `library()`. Avoid it completely while developing.

workflow alert

```
library()
```

```
use_package()
pkg::fun()
```

```
use_package("ggplot2")
```

```
shinRa
├── .Rbuildignore
├── .gitignore
├── DESCRIPTION
├── NAMESPACE
├── R/
│   ├── themes.R
└── shinRa.Rproj
```

```
use_package("ggplot2")
```

```
Package: shinRa
Title: What the Package Does (One Line, Title
  Case)
Version: 0.0.0.9000
Authors@R:
    person(given = "Malcolm",
           family = "Barrett",
           role = c("aut", "cre"),
           email = "malcolmbarrett@gmail.com")
Description: What the package does (one
  paragraph).
License: MIT + file LICENSE
Encoding: UTF-8
LazyData: true
Roxygen: list(markdown = TRUE)
Imports:
    ggplot2
```

```
use_package("ggplot2", min_version = TRUE)
Package: shinRa
Title: What the Package Does (One Line, Title
  Case)
Version: 0.0.0.9000
Authors@R:
    person(given = "Malcolm",
           family = "Barrett",
           role = c("aut", "cre"),
           email = "malcolmbarrett@gmail.com")
Description: What the package does (one
  paragraph).
License: MIT + file LICENSE
Encoding: UTF-8
LazyData: true
Roxygen: list(markdown = TRUE)
Imports:
    ggplot2 (>= 3.3.0)
```

```
use_dev_package("ggplot2")
```

```
Package: shinRa
Title: What the Package Does (One Line, Title
  Case)
Version: 0.0.0.9000
Authors@R:
    person(given = "Malcolm",
           family = "Barrett",
           role = c("aut", "cre"),
           email = "malcolmbarrett@gmail.com")
Description: What the package does (one
  paragraph).
License: MIT + file LICENSE
Encoding: UTF-8
LazyData: true
Roxygen: list(markdown = TRUE)
Imports:
    ggplot2 (>= 3.3.0.9000)
Remotes:
    tidyverse/ggplot2
```

```
use_package("shiny", type = "Suggests")
```

```
Package: shinRa
Title: What the Package Does (One Line, Title
  Case)
Version: 0.0.0.9000
Authors@R:
    person(given = "Malcolm",
           family = "Barrett",
           role = c("aut", "cre"),
           email = "malcolmbarrett@gmail.com")
Description: What the package does (one
  paragraph).
License: MIT + file LICENSE
Encoding: UTF-8
LazyData: true
Roxygen: list(markdown = TRUE)
Imports:
    ggplot2
Suggests:
    shiny
```

Testing for suggested packages

```r
if (requireNamespace("shiny", quietly = TRUE)) {
  # ... shiny code
}
```

or `use_package("rlang")`

```r
# offers to install if not already installed
rlang::check_installed("shiny")
# ... shiny code
```

```
use_package("R", "Depends", min_version = "3.2.0")
Package: shinRa
Title: What the Package Does (One Line, Title
  Case)
Version: 0.0.0.9000
Authors@R:
    person(given = "Malcolm",
           family = "Barrett",
           role = c("aut", "cre"),
           email = "malcolmbarrett@gmail.com")
Description: What the package does (one
  paragraph).
License: MIT + file LICENSE
Encoding: UTF-8
LazyData: true
Roxygen: list(markdown = TRUE)
Imports:
    ggplot2
Suggests:
    shiny
Depends:
    R (>= 3.2.0)
```

```
use_tidy_description()
Package: shinRa
Title: What the Package Does (One Line, Title
  Case)
Version: 0.0.0.9000
Authors@R:
    person(given = "Malcolm",
           family = "Barrett",
           role = c("aut", "cre"),
           email = "malcolmbarrett@gmail.com")
Description: What the package does (one
  paragraph).
License: MIT + file LICENSE
Depends:
    R (>= 3.2.0)
Imports:
    ggplot2
Suggests:
    shiny
Encoding: UTF-8
LazyData: true
Roxygen: list(markdown = TRUE)
```

Your Turn 3

Fix the code in `R/themes.R` to use `ggplot2::` instead of `library(ggplot2)`

Run `use_package("ggplot2")` to add ggplot2 to `Imports`

Re-load the package (`Cmd/Ctrl+Shift+L`) and run this code to make sure it works

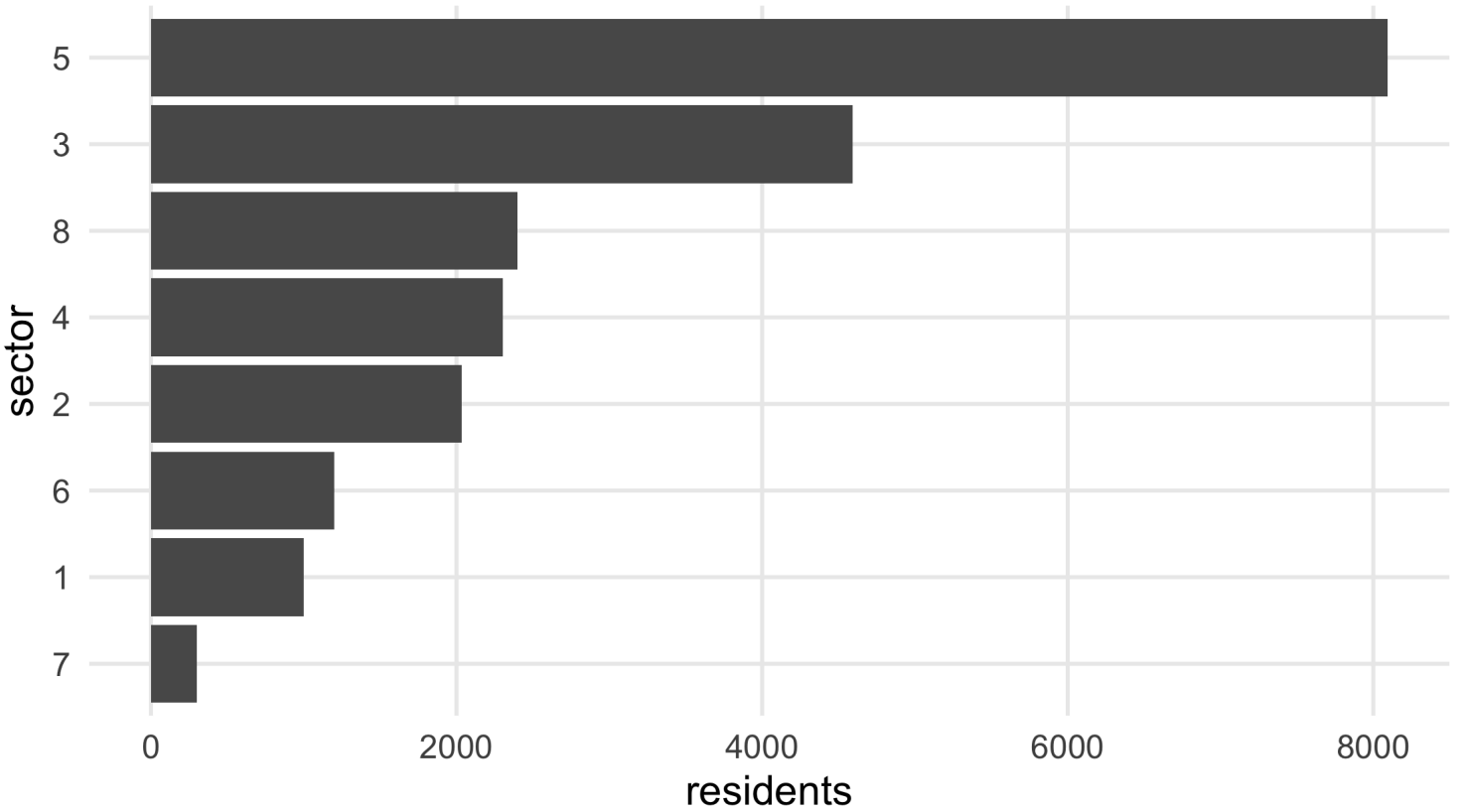
Your Turn 3

```
use_package("ggplot2")
```

in `R/themes.R`:

```
theme_avalanche <- function() {
  ggplot2::theme_minimal(base_size = 14) +
    ggplot2::theme(panel.grid.minor = ggplot2::element_blank())
}
```

Your Turn 3



Importing functions directly

```
use_import_from("pkg.name", "fun")
```

Adds `@importFrom pkg.name fun`, letting you write `fun()` instead of `pkg.name::fun()`

Import the whole package with `@import pkg.name`. *Use this with caution!*

Commonly imported components

```
use_tibble()

use_data_table()

use_pipe()

use_rcpp*()
```

Your Turn 4

We need roxygen2 for this exercise. We'll learn more about it in the next module. For now, just run `use_roxygen_md()`

Run `use_tibble()` and `use_data_table()`

In `R/get_data.R`, edit the function to be able to return a data table: Add the argument `data_table = FALSE`. If `data_table` is `TRUE`, convert the data frame with `data.table::as.data.table()`

Load the package and run this code to make sure it works

Your Turn 4

```
use_roxygen_md()
use_tibble()
use_data_table()
```

```
get_resident_data <- function(data_table = FALSE) {
  residents_per_sector <- db_con("residents_per_sector")

  if (data_table)
    return(data.table::as.data.table(residents_per_sector))

  tibble::as_tibble(residents_per_sector)
}
```

Your Turn 4

```
##    sector residents
## 1:      1      1000
## 2:      2      2034
## 3:      3      4594
## 4:      4      2304
## 5:      5      8093
## 6:      6      1200
## 7:      7       300
## 8:      8      2398
```

```
data.table::is.data.table(res_data)
```

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

Your Turn 4: Stretch goal

Run `use_pipe()` to add the magrittr pipe to your package. What changed?

Organizing `.R` files

Organizing `.R` files

Less than 1:1, more than all:1

Organizing `.R` files

Less than 1:1, more than all:1

`utils.R`

Organizing `.R` files

Less than 1:1, more than all:1

`utils.R`

`zzz.R`, `.onLoad`

See R Packages, ed. 2 for more.