script

Turn on keyboard shortcut broadcasting

Package Structure and State

```
.libPaths()
```

back to slides

Get Ready

```
install.packages(
  c("devtools", "roxygen2", "testthat", "knitr", "pkgdown")
)
```

```
all(c("devtools", "roxygen2", "testthat", "knitr", "pkgdown") %in% installed.packages())
```

```
library(devtools)

has_devel()
dev_sitrep()
git_sitrep()
```

go through each line

back to slides

BREAK

Create a package

Discussion how that path expansion won't work on windows

```
library(devtools)
create_package("~/Desktop/libminer")
```

Explore package structure in RStudio with learners

back to slides

```
# gert::git_config()
# gert::git_config_global()
library(devtools)
use_git_config(
  user.name = "Sam Albers",
  user.email = "sam.albers@gmail.com"
)
```

Git global: Name, email GitHub user: PAT discovered, User, email(s)

```
use_git()
```

**Explain that `use_git` automatically commits everything in the repo.

Ok but loading devtools all the time is getting annoying.

back to slides

```
usethis::use_devtools()
```

Restart R - check that devtools is loaded

back to slides

```
use_r("lib-summary")
```

```r
lib_summary <- function() {
  pkgs <- utils::installed.packages()
  pkg_tbl <- table(pkgs[, "LibPath"])
  pkg_df <- as.data.frame(pkg_tbl, stringsAsFactors = FALSE)
  names(pkg_df) <- c("Library", "n_packages")
  pkg_df
}
```

back to slides

```r
load_all()

lib_summary()
```

Commit

back to slides

```r
check()
```

back to slides

Open DESCRIPTION file

```r
use_mit_license()
```

Commit

back to slides

DESCRIPTION file:

```
Package: libminer
Title: Explore Your R Libraries
Version: 0.0.0.9000
Authors@R:
    person("Sam", "Albers", , "sam.albers@gmail.com", role = c("aut", "cre"),
           comment = c(ORCID = "0000-0002-9270-7884"))
Description: Provides functions for learning about your R libraries, and the
    packages you have installed.
```

back to slides

```
check()
```

Commit

back to slides

Push all your committed code to that repository

```
use_github()
```

At this step you may be required to enter in your GitHub username and password.

```
git push --set-upstream origin main
```

Go to GitHub page, explore show clone of local

back to slides

```
edit_r_profile()
```

put this in the chat

```
# Set usethis options
#
options(
  usethis.description = list(
    "Authors@R" = utils::person(
      "Jane", "Doe",
      email = "jane@example.com",
      role = c("aut", "cre"),
      comment = c(ORCID = "0000-1111-2222-3333")
    )
  )
)

options(
  usethis.description = list(
    "Authors@R" = utils::person(
      "Sam", "Albers",
      email = "sam.albers@gmail.com",
      role = c("aut", "cre"),
```

```
      comment = c(ORCID = "0000-0002-9270-7884")
    )
  )
)

options(
  warnPartialMatchArgs = TRUE,
  warnPartialMatchDollar = TRUE,
  warnPartialMatchAttr = TRUE
)
```

back to slides

Documentation

`Ctrl + .`

`Ctrl+Alt+Shift+R`

```
#' R Library Summary
#'
#' Provides a brief summary of the package libraries on your machine
#'
#' @return A data.frame containing the count of packages in each of the user's
#'   libraries
#' @export
#'
#' @examples
#' lib_summary()
```

```
document()
```

Go to `man/lib_summary.Rd`

```
load_all()

?lib_summary
```

```
check()
```

Look at NAMESPACE

commit

back to slides

Package-level documentation

```
use_package_doc()

document()
```

Go to `man/libminer2-package.Rd`

Preview and check again

```
load_all()

?libminer

check()
```

back to slides

```
install()
library(libminer)

lib_summary() # note one more package than before - that's yours!
```

commit and push

README

```
use_readme_rmd()
```

````
---
output: github_document
---

<!-- README.md is generated from README.Rmd. Please edit that file -->



# libminer

<!-- badges: start -->

<!-- badges: end -->

The goal of libminer is to provide an overview of your R library setup. It is a toy
package created as a part of a workshop and not meant for serious use.

## Installation

You can install the development version of libminer from [GitHub](https://GitHub.com/) with:

```r
# install.packages("devtools")
devtools::install_GitHub("ateucher/libminer")
```

## Example usage

To get a count of installed packages in each of your library locations,
optionally with the total sizes, use the `lib_summary()` function:

```{r example}
library(libminer)
lib_summary()
```
````

```
build_readme()
```

```
check()

install()
```

commit + push

return to slides

Testing

restart R

```
use_testthat()
```

Have `R/lib-summary.R` open

```
use_test()
```

```
test_that("lib_summary returns expected results", {
  res <- lib_summary()
  expect_s3_class(res, "data.frame")
  expect_equal(ncol(res), 2)
  expect_equal(names(res), c("Library", "n_packages"))
  expect_type(res$Library, "character")
  expect_type(res$n_packages, "integer")
})

test_that("lib_summary fails appropriately", {
  expect_error(lib_summary("foo"), "unused argument")
})
```

```
test()

check()
```

reinforce file structure

commit

back to slides

Workflow iteration

Set keyboard shortcuts

Micro:

```
load_all() # shift cmd L
```

- run expectations interactively

Mezzo:

- run test file `test_active_file()`: cmd T
- report coverage of active file: cmd R

Macro: (we already did) - `test_package()`: shift cmd T - report package coverage: shift cmd R

Back to Slides

Talk about what we would NOT want to snapshot

Error is most obvious:

```
test_that("lib_summary fails appropriately", {
  expect_snapshot(lib_summary("foo"), error = TRUE)
})
```

Maybe names?

```
test_that("lib_summary returns expected results with defaults", {
  res <- lib_summary()
  expect_snapshot(names(res))
})
```

Back to slides

Dependencies

```
use_package("fs")
```

Look at DESCRIPTION, NAMESPACE (no change)

commit

```r
lib_summary <- function(sizes = FALSE) {
  if (!is.logical(sizes)) {
    stop("'sizes' must be logical (TRUE/FALSE).")
  }

  pkgs <- utils::installed.packages()
  pkg_tbl <- table(pkgs[, "LibPath"])
  pkg_df <- as.data.frame(pkg_tbl, stringsAsFactors = FALSE)
  names(pkg_df) <- c("Library", "n_packages")

  if (sizes) {
    pkg_df$lib_size <- vapply(
      pkg_df$Library,
      function(x) {
        sum(fs::file_size(fs::dir_ls(x, recurse = TRUE)))
      },
      FUN.VALUE = numeric(1)
    )
  }
  pkg_df
}
```

```r
test() # failure for unused argument
```

```r
test_that("lib_summary fails appropriately", {
  expect_error(lib_summary(sizes = "foo"), "'sizes' must be logical")
})

test_that("sizes argument works", {
  res <- lib_summary(sizes = TRUE)
  expect_equal(names(res), c("Library", "n_packages", "lib_size"))
  expect_type(res$lib_size, "double")
})
```

commit

```r
check() # will warn about undocumented parameter
```

Ctrl+Alt+Shift+R will insert the spot for the sizes param

```
#' Provides a brief summary of the package libraries on your machine
#'
#' @param sizes Should the sizes of the libraries be calculated?
#'   Logical; default `FALSE`.
#'
#' @return A data.frame containing the count of packages in each of the user's
#'   libraries. A `lib_size` column is included if `sizes = TRUE`.
#' @export
#'
#' @examples
#' lib_summary()
#' lib_summary(sizes = TRUE)
```

```
document()

check()
```

Test it out

```
load_all()

?lib_summary

lib_summary(sizes = TRUE)
```

commit

back to slides

```
use_import_from("purrr", "map_dbl")
```

Look at: DESCRIPTION, R/libminer-package.R, NAMESPACE and see what that did.

```
lib_summary <- function(sizes = FALSE) {
  pkgs <- utils::installed.packages()
  pkg_tbl <- table(pkgs[, "LibPath"])
  pkg_df <- as.data.frame(pkg_tbl, stringsAsFactors = FALSE)
  names(pkg_df) <- c("Library", "n_packages")

  if (sizes) {
    pkg_df$lib_size <- map_dbl(
```

```
      pkg_df$Library,
      \(x) sum(fs::file_size(fs::dir_ls(x, recurse = TRUE)))
    )
  }

  pkg_df
}
```

```
test()
```

Note importance of tests here

```
check()
```

commit and push

Continuous Integration

```
use_github_action()
```

commit and push

return to slides

Design Principles

Refactor this function to add 2 helper functions

```
lib_summary <- function(sizes = FALSE) {
  pkgs <- utils::installed.packages()
  pkg_tbl <- table(pkgs[, "LibPath"])
  pkg_df <- as.data.frame(pkg_tbl, stringsAsFactors = FALSE)
  names(pkg_df) <- c("Library", "n_packages")

  if (sizes) {
    pkg_df$lib_size <- map_dbl(
      pkg_df$Library,
      \(x) sum(fs::file_size(fs::dir_ls(x, recurse = TRUE)))
```

```
    )
  }
  pkg_df
}
```

into:

```
#' Generate a dataframe of installed packages
#'
#' @return dataframe of all packages installed on a system
#' @export
lib <- function() {
  pkgs <- utils::installed.packages()
  as.data.frame(pkgs, stringsAsFactors = FALSE)
}

#' calculate sizes
#'
#' @param df a data.frame
#'
#' @return df with a lib_size column
#' @noRd
calculate_sizes <- function(df) {
  df$lib_size <- map_dbl(
    df$library,
    \(x) sum(file_size(dir_ls(x, recurse = TRUE)))
  )
  df
}

#' Provides a brief summary of the package libraries on your machine
#'
#' @param sizes Should the sizes of the libraries be calculated?
#'   Logical; default `FALSE`.
#'
#' @return A data.frame containing the count of packages in each of the user's
#'   libraries. A `lib_size` column is included if `sizes = TRUE`.
#' @export
#'
#' @examples
#' lib_summary()
#' lib_summary(sizes = TRUE)
```

```
lib_summary <- function(sizes = FALSE) {
  pkg_df <- lib()
  pkg_df <- table(pkg_df$LibPath)
  pkg_df <- as.data.frame(pkg_df, stringsAsFactors = FALSE)
  names(pkg_df) <- c("Library", "n_packages")

  if (sizes) {
    pkg_df <- calculate_sizes(pkg_df)
  }

  pkg_df
}
```

```
document()
```

check, commit and push

No `fs::`

```
#' Generate a dataframe of installed packages
#'
#' @return dataframe of all packages installed on a system
#' @export
lib <- function() {
  pkgs <- utils::installed.packages()
  as.data.frame(pkgs, stringsAsFactors = FALSE)
}

#' calculate sizes
#'
#' @param df a data.frame
#'
#' @return df with a lib_size column
#' @noRd
calculate_sizes <- function(df) {
  df$lib_size <- map_dbl(
    df$Library,
    \(x) sum(fs::file_size(fs::dir_ls(x, recurse = TRUE)))
  )
  df
}
```

```r
lib_summary <- function(sizes = FALSE) {
  pkg_df <- lib()
  pkg_df <- table(pkg_df$LibPath)
  pkg_df <- as.data.frame(pkg_df, stringsAsFactors = FALSE)
  names(pkg_df) <- c("Library", "n_packages")

  if (sizes) {
    pkg_df <- calculate_sizes(pkg_df)
  }
  pkg_df
}
```

update README

- include `lib()` in example usage

```
To get a nicely formatted tibble of your installed packages, use the `lib()`
function:

::: {.cell}

```{.r .cell-code}
library(libminer)
lib()
```
:::
```

** back slides **

Input checking

```r
lib_summary <- function(sizes = FALSE) {
  pkg_df <- lib()
  pkg_df <- table(pkg_df$LibPath)
  pkg_df <- as.data.frame(pkg_df, stringsAsFactors = FALSE)
  names(pkg_df) <- c("Library", "n_packages")

  if (sizes) {
    pkg_df <- calculate_sizes(pkg_df)
```

```
  }
  pkg_df
}
```

to:

```
lib_summary <- function(sizes = FALSE) {
  if (!is.logical(sizes)) {
    stop("sizes should be a logical (TRUE/FALSE) value", call. = FALSE)
  }
  pkg_df <- lib()
  pkg_df <- table(pkg_df$LibPath)
  pkg_df <- as.data.frame(pkg_df, stringsAsFactors = FALSE)
  names(pkg_df) <- c("Library", "n_packages")

  if (sizes) {
    pkg_df <- calculate_sizes(pkg_df)
  }
  pkg_df
}
```

Modify a test

explain escape parentheses

```
expect_error(lib_summary(sizes = "foo"), regexp = "sizes should be a logical \\(TRUE/FALSE\\)
```

back to slides

Code readability

- 80 character lines
- spaces around operators
- proper indentation
- newlines where appropriate
- don't be afraid of vertical spaces

demo styler

Creating a website

** go over what files were created **

```
use_pkgdown_github_pages()
```

Vignettes

** go over what files were created **

```
use_vignette("lib-sitrep", "Package Library Situation Report")
```

** add an article with ggplot2 **

```
use_article("musings-lib", title = "Musings on the state of libminer")
```

add ggplot2 to action

Tidyverse

Demo - not part of package

Do in a new file eg scratch/tidyverse-testing.R and add scratch to Rbuildignore

data masking

Error: object mpg not found

```
var_summary <- function(data, var) {
  data |>
    summarise(
      min = min(var),
      max = max(var)
    )
}

mtcars |>
  group_by(cyl) |>
  var_summary(mpg)
```

```r
var_summary <- function(data, var) {
  data |>
    summarise(
      min = min({{ var }}),
      max = max({{ var }})
    )
}

mtcars |>
  group_by(cyl) |>
  var_summary(mpg)
```

```r
big_cars_summary <- function(var) {
  mtcars |>
    filter(.data$cyl >= 6) |>
    group_by(.data$cyl) |>
    summarise(
      n = n(),
      mean = mean({{ var }}),
    )
}

big_cars_summary(disp)
```

Your turn solution

```r
height_sum <- function(data, group_var) {
  data |>
    dplyr::group_by({{ group_var }}) |>
    dplyr::summarise(
      n = dplyr::n(),
      mean_height = mean(.data$height)
    )
}

height_sum(starwars, hair_color)
```

Your turn solution

```
height_sum <- function(data, ...) {
  data |>
    dplyr::group_by(...) |>
    dplyr::summarise(
      n = dplyr::n(),
      mean_height = mean(.data$height)
    )
}

height_sum(starwars, hair_color, eye_color)
```

Dynamic dots

```
var_summary <- function(data, var) {
  data |>
    summarise(
      "{{var}}_min" := min({{ var }})
    )
}

mtcars |>
  group_by(cyl) |>
  var_summary(mpg)
```

Your turn solution

```
dynamic_sum <- function(data, group_var, sum_var) {
  data |>
    dplyr::group_by({{ group_var }}) |>
    dplyr::summarise(
      n = dplyr::n(),
      "mean_{{sum_var}}" := mean({{ sum_var }})
    )
}

dynamic_sum(starwars, hair_color, mass)
```

tidyselect

```
mtcars |>
  dplyr::group_by(cyl) |>
  dplyr::summarise(
    dplyr::across(c("mpg", "disp"), mean)
  )

cols <- c("mpg", "disp", "xyz")
mtcars |>
  dplyr::group_by(cyl) |>
  dplyr::summarise(
    dplyr::across(dplyr::all_of(cols), mean)
  )

cols <- c("mpg", "disp", "xyz")
mtcars |>
  dplyr::group_by(cyl) |>
  dplyr::summarise(
    dplyr::across(dplyr::any_of(cols), mean)
  )
```

Using a character vector:

```
summy <- function(df, cols) {
  df |>
    group_by(.data$cyl) |>
    summarise(
      across(all_of(cols), .fns = mean)
    )
}
```

```
summy(mtcars, c("mpg", "disp"))
```

Using bare names or tidy-select helpers:

```
summy <- function(df, cols) {
  df |>
    group_by(.data$cyl) |>
    summarise(
```

```
      across({{ cols }}, .fns = mean)
    )
}
```

```
summy(mtcars, c(mpg, disp))
summy(mtcars, starts_with("mp"))
summy(mtcars, where(is.numeric))
```

Setup

```
use_package("dplyr")
```

We are using the new base pipe, need R >= 4.1

```
lib_summary <- function(by) {
  pkgs <- lib()

  dplyr::group_by(pkgs, by) |>
    dplyr::count()
}
```

```
load_all()
lib_summary(LibPath)
```

Errors - no column called 'by'

2a - use curly-curly with bare names

```
lib_summary <- function(by) {
  pkgs <- lib()

  dplyr::group_by(pkgs, {{ by }}) |>
    dplyr::count()
}
```

```
load_all()
lib_summary(LibPath)
```

2b - use .data[[x]] with character

```
use_import_from("rlang", ".data")
```

```
lib_summary <- function(by) {
  pkgs <- lib()

  dplyr::group_by(pkgs, .data[[by]]) |>
    dplyr::count()
}
```

```
load_all()
lib_summary("License")
```

3 - add sizes

```
lib_summary <- function(by) {
  pkgs <- lib()

  pkgs <- calculate_sizes(pkgs)

  dplyr::group_by(pkgs, {{by}}) |>
    dplyr::summarise(
      n = dplyr::n(),
      size = sum(size)
    )
}

calculate_sizes <- function(df) {
  df |>
    dplyr::mutate(
      size = purrr::map_dbl(
        fs::path(LibPath, Package),
        \(x) sum(fs::file_size(fs::dir_ls(x, recurse = TRUE)))
```

```
      )
    )
}
```

```
load_all()
lib_summary(LibPath)
check() # should throw notes about undefined globals
```

4 - add .data

```
lib_summary <- function(by) {
  pkgs <- lib()

  pkgs <- calculate_sizes(pkgs)

  dplyr::group_by(pkgs, {{by}}) |>
    dplyr::summarise(
      n = dplyr::n(),
      size = sum(.data$size)
    )
}

calculate_sizes <- function(df) {
  df |>
    dplyr::mutate(
      size = purrr::map_dbl(
        fs::path(.data$LibPath, .data$Package),
        \(x) sum(fs::file_size(fs::dir_ls(x, recurse = TRUE)))
      )
    )
}
```

5 use ... for multiple by's

```
lib_summary <- function(...) {
  lib() |>
    calculate_sizes() |>
    dplyr::group_by(...) |>
```

```
    dplyr::summarise(
      n = dplyr::n(),
      size = sum(.data$size)
    )
}
```

Final - make sizes conditional again

Also, drop groups so we don't get the message any more

```
lib_summary <- function(..., sizes = FALSE) {
  if (!is.logical(sizes)) {
    stop("'sizes' must be logical (TRUE/FALSE).", call. = FALSE)
  }

  lib() |>
    calculate_sizes(do_calc = sizes) |>
    dplyr::group_by(...) |>
    dplyr::summarise(
      n = dplyr::n(),
      dplyr::across(dplyr::any_of("size"), sum),
      .groups = "drop"
    )
}

calculate_sizes <- function(df, do_calc) {
  if (!isTRUE(do_calc)) return(df)

  df |>
    dplyr::mutate(
      size = purrr::map_dbl(
        fs::path(.data$LibPath, .data$Package),
        \(x) sum(fs::file_size(fs::dir_ls(x, recurse = TRUE)))
      )
    )
}

lib <- function() {
  utils::installed.packages() |>
    dplyr::as_tibble()
}
```

```
lib_summary(LibPath, License)
lib_summary(LibPath, License, sizes = TRUE)
lib_summary(LibPath, License, sizes = 10)
```

```
test()
# update tests
check()
# update documentation
```

CLI

Final

```
lib_summary <- function(..., sizes = FALSE) {
  if (!is.logical(sizes)) {
    cli::cli_abort("You supplied {.val {sizes}} to {.var sizes}. It should be a {.cls logical
  }

  lib() |>
    calculate_sizes(do_calc = sizes) |>
    dplyr::group_by(...) |>
    dplyr::summarise(
      n = dplyr::n(),
      dplyr::across(dplyr::any_of("size"), sum),
      .groups = "drop"
    )
}

calculate_sizes <- function(df, do_calc) {
  if (!isTRUE(do_calc)) return(df)

  cli::cli_inform(c("i" = "Calculating sizes..."))

  df |>
    dplyr::mutate(
      size = purrr::map_dbl(
        fs::path(.data$LibPath, .data$Package),
        \(x) sum(fs::file_size(fs::dir_ls(x, recurse = TRUE)))
      )
```

```
    )
}
```

```
lib_summary(LibPath, License)
lib_summary(LibPath, License, sizes = TRUE)
lib_summary(LibPath, License, sizes = 10)
lib_summary(LibPath, License, sizes = "hello")
lib_summary(LibPath, License, sizes = NULL)
```

update tests

- snapshot for error
- groups

Releasing

```
use_news_md()
```