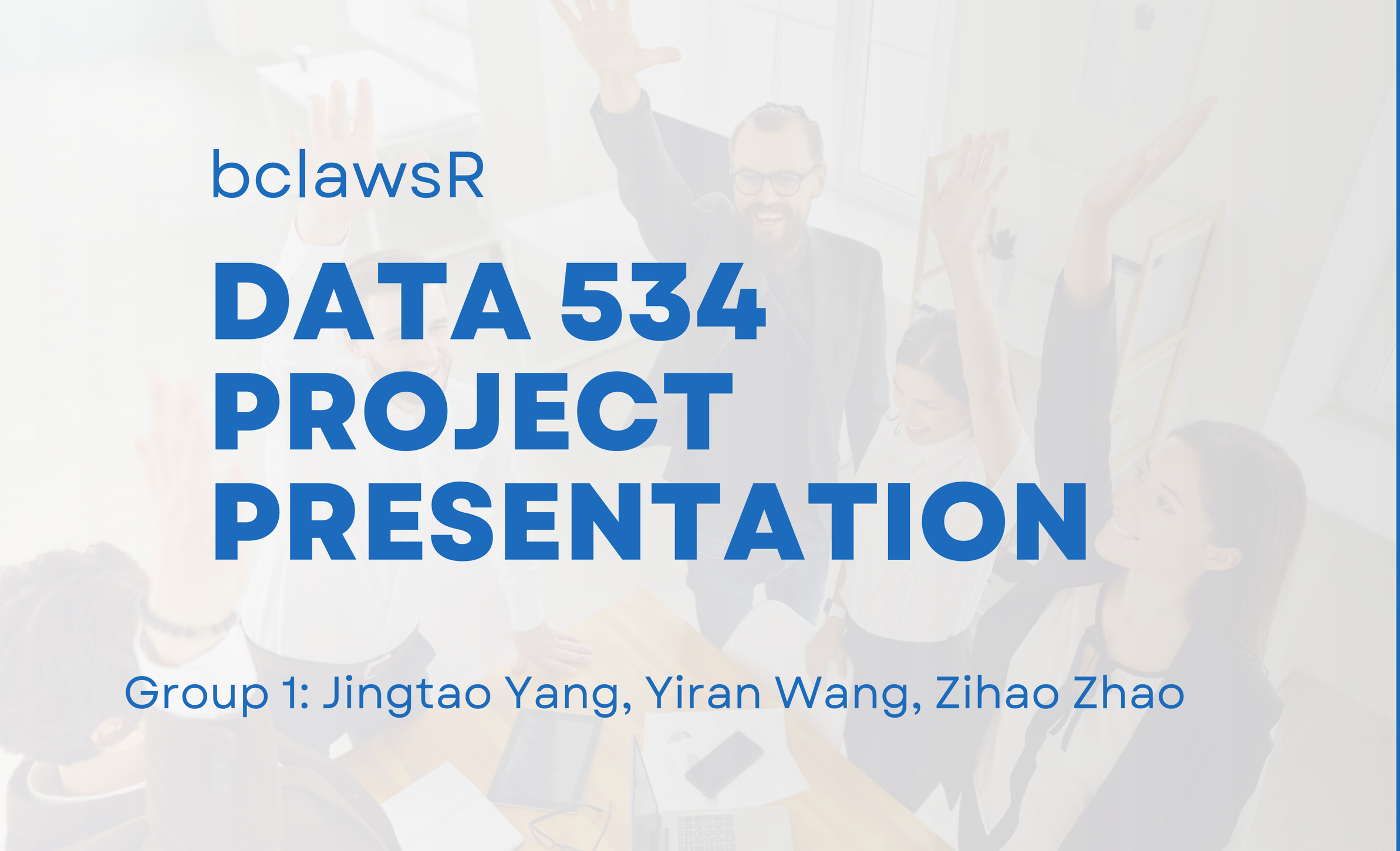
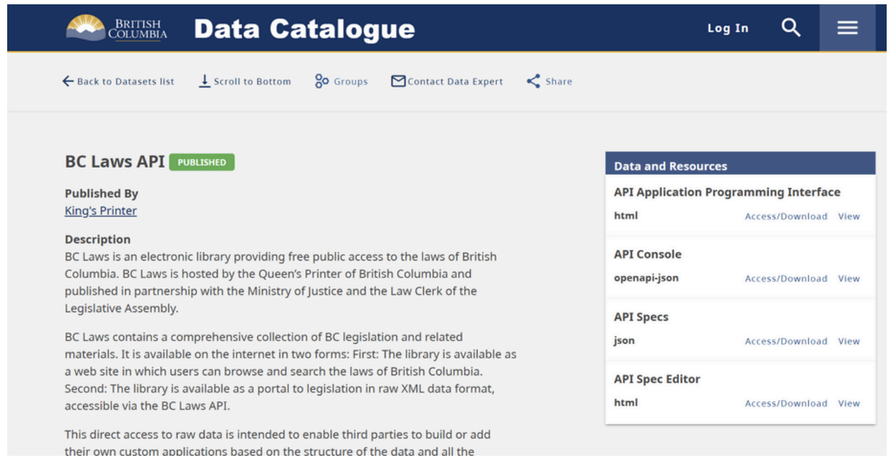


bclawsR

DATA 534 PROJECT PRESENTATION

Group 1: Jingtao Yang, Yiran Wang, Zihao Zhao

Motivation



- British Columbia laws and regulations are available online through BC Laws and its public API.
- These public information are useful for policy and government studies, academic projects, and public understanding of the laws.
- Currently, there is no simple R package that:
- 1. searches BC laws by keyword and year
- 2. retrieves law text in a easy-to read format
- 3. provides a ready-to-use workflow from API to plot

make legal search & basic analysis 1–2 lines of R

Our Project Is ...

bclaws is an R package that wraps the BC Laws REST API, providing a complete workflow from search, to trend analysis and visualization,

Data source

- BC Laws Civix API

Inputs: keyword, year range, law type
Outputs: tidy tables + plot + plain-text law content

Core User Scenarios

1. I want to search a keyword (e.g., “tax”) across BC Laws and see which documents match.
2. I want to view how the keyword changes over time in a graph
3. I want to fetch the plain-text content of a specific law for reading or downstream analysis.
4. I want to have someone analyze the trends for me, and provide me some ideas for furture analysis

Package Architecture

Low-level

bc_get()

One helper that sends a Civix request and returns the raw response (XML/HTML)

Process: build URL → attach query params → perform request → check status

Output: raw response used by higher-level functions (search_laws(), get_law_text())

Package Architecture

Core API capabilities

search_laws(keyword, start, end, type , nFrag, lFrag)

Calls the API search endpoint to retrieve search results for a keyword.
Output: an xml_document

parse_search_results(xml_doc)

Converts the raw XML search results into a tidy data.frame, contains title, law_type, law_id, loc, hits

Package Architecture

Analysis

keyword_trend_paged(keyword, from, to, type, pages, page_size)

- measure how often a keyword appears across years
- repeat search across year windows → parse → aggregate counts by year,
- loops pages (pages, page_size) and aggregates across pages

get_law_text(law_type, law_id)

- law_type: document type (from search results)
- law_id: document identifier (from search results)

Output: a long string containing the law's text.

Package Architecture

Visulization

plot_keyword_trend()

Returns a simple time-series plot, keyword hits vs year to show the trend of a certain keyword over time

Package Architecture

AI Summarization

call_openai_responses()

Send a prompt to OpenAI and return generated text

- **Inputs: prompt and API key**
- **Output: a short summary string**

summarize_keyword_trend()

Send the output from keywork_trend_paged(), and the prepared prompt, and get back the summarization text

Live Demo !

Other Components

Installable on Windows / Mac / Linux

- Standard R package structure (DESCRIPTION , NAMESPACE, R, man)
- Install locally - devtools::install(".")
- GitHub Actions runs R CMD check on: ubuntu-latest, windows-latest, macos-latest

Testing

- XML parsing: parse_search_results()
- validation: request helpers and input checks

Vignette

The vignette shows the full workflow: search, parse, trend, plot, fetch text, so a new user can reproduce results in one place.

Limitations / Future Works

- ***We only fetch a limited number of pages, we may miss some documents.***
- ***Fetching speed depends on the BC Laws server and network conditions***
- ***Limited visulization selections (only trend plot)***

- ***We are looking for extending our project by supporting more custimized law categories for fetching, fine-tuning our prompt to generate better ai-summary, and add more visulization function for more diversed visulization***

Thank You for Watching !