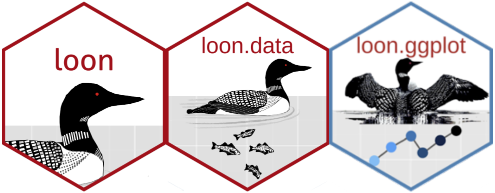
Interactive Exploratory Data Visualization in R

A workshop introduction to loon and related packages



Wayne Oldford

UNIVERSITY OF
WATERLOO

June 6, 2020

Making this work

1. You should already have your machine set up as described in the preparations
   - Most recent R and RStudio and all necessary packages installed and tested
   - A mouse/trackpad with two buttons and scrolling enabled
   - ideally two screens:
     - one for my presentation
     - one for your work in RStudio (the larger of your two screens)
   - a microphone and headphones/earbuds
   - if your machine is not working, there is not much we can do about it today
2. Communication
   - I (the host) will be speaking and sharing my screen
   - audio/video communication is mostly one way:
     - this may be the biggest challenge ($>$ 250 registrations)
     - only the host and designated panellists can be heard
     - I can switch people to be a panelist when they want to speak/share their screen ... I think :-)
     - everyone (I think) joins muted so unmute yourself if called on
   - there is a text chat facility (public and private)
     - you can “raise your hand”, make it urgent (“!”), and give some canned comments from a menu (“Yes”, “No”, “Too fast”, “Applause”, ...)
     - I will try to pay attention to this but forgive me if I overlook a comment/question. Shout it out with “!”.
3. This is all new to me ... so likely I will mess up (more than once 🙂)

What is going to happen?

Mainly I will be leading you through several R scripts

- You should have these downloaded already
- I will be sharing my screen throughout
    - sharing the code onscreen and explaining it as we go
    - encouraging you to try stuff and waiting for you to explore
    - showing the results when I run the code
    - showing my interactions with `loon` and with `RStudio`
- So mainly, I will be following you
    - which makes pacing a challenge
    - PLEASE stop me if/when you have a question. Guaranteed you will not be the only person with that question.

I will also occasionally switch to slides like these.

When will it happen?

Part 1 (approx 2 hours)

- `covidNZ.R` and `plots3D.R` will be the scripts covered
- the first of these is where we will really work `loon` in the case of a topical dataset.
- the second involves two different datasets

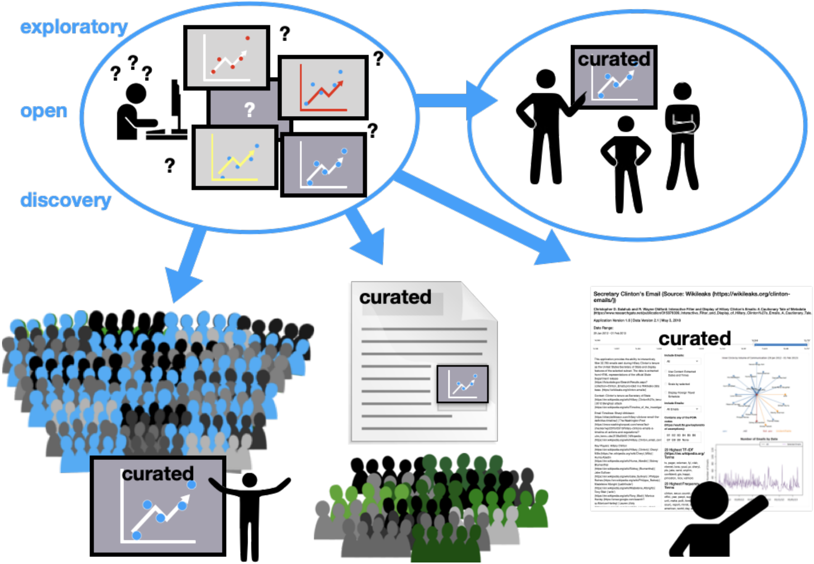
Break: We will break for an hour.

Part 2: (approx 2 hours)

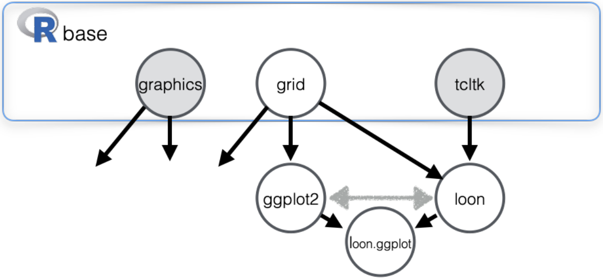
- `higher_dimensional.R` for visualizing more than 3 dimensions
- `pipes.R` using `loon` with `pipes` (`magrittr`) and an interesting Canadian data set on minority distributions in cities across Canada. This also illustrates some high-dimensional methods.
- `loon.ggplot.R` shows how `loon` and `ggplot2` can work together
- `teachingDemo.R` illustrates some visualizations that can be created for teaching.

Questions?

Exploratory visualization


exploratory
open
discovery
?
curated
curated
Secretary Clinton's Email (Source: Wikileaks (https://wikileaks.org/clinton-emails/))
Application Version 1.5 | Data Version 2.1 | May 3, 2019
Date Range:
curated
curated

Graphics systems in R



R ships with three fundamental graphics packages on which most graphics packages are built:

- `graphics` ... simple but powerful traditional base graphics
  - e.g. `hist()`, `plot()`, `pairs()`
- `grid` ... powerful classically designed computer graphics system
  - coordinate systems, viewports, ...
- `tcltk` ... a classic widget packing system
  - `widgets`, `events`, `sliders`, ... interactive

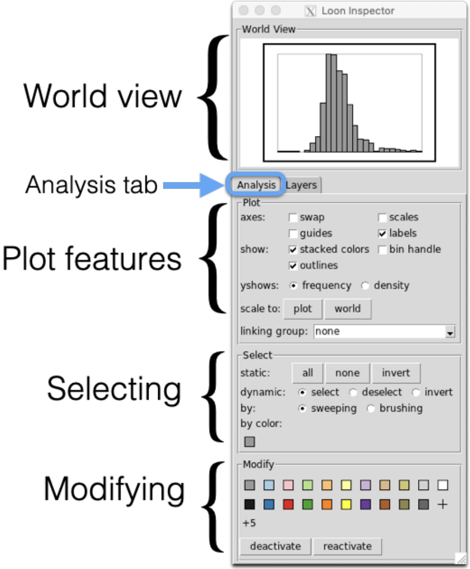
E.g. `ggplot2` is built on `grid`, `loon` on `tcltk`.

Covid-19 cases in New Zealand

Let’s get started!

Open up the file ‘covidNZ.R‘

The loon inspector



World view

Analysis tab

Plot features

Selecting

Modifying

Loon Inspector

World View

Analysis | Layers

Plot

axes: □ swap ☑ labels □ guides □ scales

show: ☑ stacked colors □ bin handle ☑ outlines

yshows: ● frequency ○ density

scale to: plot | world

linking group: none

Select

static: all | none | invert

dynamic: ● select ○ deselect ○ invert

by: ● sweeping ○ brushing

by color: ▨

Modify

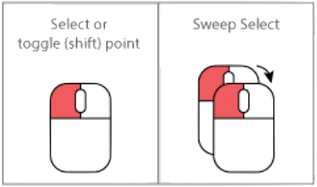
+5

deactivate | reactivate

Mouse movements - selection

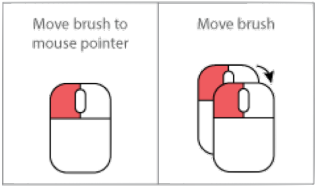
Selecting

Shift will not reset previous selection



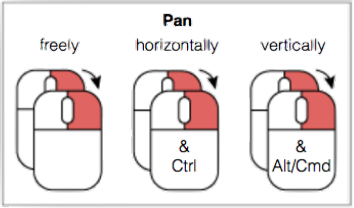
Brushing

Shift will make the selection permanent

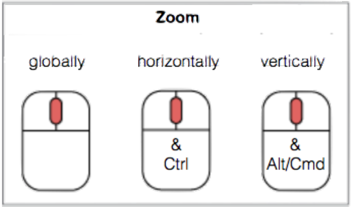


Mouse movements - panning and zooming

Panning

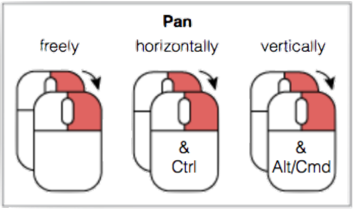


Zooming

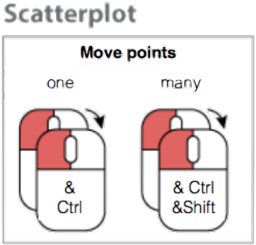


Mouse movements - moving

Panning is like moving the background



Moving points (scatterplots)



Back to ‘covidNZ.R‘

Interactive query logic

Classic logical queries:

- NOT A
  - select A and invert
  - select A and deactivate A
- A OR B
  - select both A and B
  - select both A and B, invert and deactivate
- A AND B
  - select A, invert, deactivate not A; select B in what remains (or B then A)
  - do as above, then invert and deactivate.

The result of the query is either - what remains selected, or - what remains active

Back to ‘covidNZ.R‘

Interactive three dimensional scatterplots

Open up the file ‘plots3D.R‘

BREAK

End of Part 1?

Back in an hour.

Multi-dimensional visualization in loon

Beginning Part 2?

Open up the file 'higher_dimensional.R'

Computational pipelines

Imagine pipes joined end to end in a **pipeline** through which data (output) flows where the data is treated by different operations along the way.

Sometimes, this can be a **more natural program flow**.

For example, Hadley Wickham likes to illustrate this point using the nursery rhyme "Little Bunny foo foo" (sung to the tune of the traditional Canadian children's song "Alouette").

Little bunny foo foo

hopping through the forest

scooping up the field mice

bopping them on the head.

The putative pipeline payoff

How do we represent this natural language (English) expression

> *Little bunny foo foo / hopping through the forest / scooping up the field mice / bopping them on the head*

and represent it in code?

```
# Start with a little bunny use forward assignment to name it
foo_foo <- little_bunny()
```

Which is a more natural expression? This standard procedural version?

```
# Without pipes:
bop_on(scoop_up(hop_through(foo_foo, forest), field_mouse), head)
```

Or this pipelined version?

```
# Or with pipes:
foo_foo %>% hop_through(forest) %>% scoop_up(field_mouse) %>% bop_on(head)
```

Note: the assignment `little_bunny() -> foo_foo` appeared in neither expression.

Never mind . . .

Building loon plots with pipes

Open up the file ‘pipes.R‘

loon versus ggplot2?

Open up the file ‘loon.ggplot.R‘

Teaching concepts with loon

Open up the file ‘teachingDemo.R‘

Credits

`loon` has had a lot of person-years work into it:

- began with Adrian Waddell's 2016 PhD thesis (though we went through several design iterations in the preceding years)
- continues with Zehao Xu's current PhD thesis (loon <−> grid, `loon.ggplot` and soon `loon.shiny`)
- other named contributors are Martin Gauch (`l_plot3D`)

Many undergraduate and graduate students have also contributed in their own way, not least being the guinea pigs for testing code!

Some students might also be interested in my 2017 Data Visualization lectures on YouTube: https://youtu.be/Khj94kaJjWo (you can at least fast-forward through them!)

Fin

Better