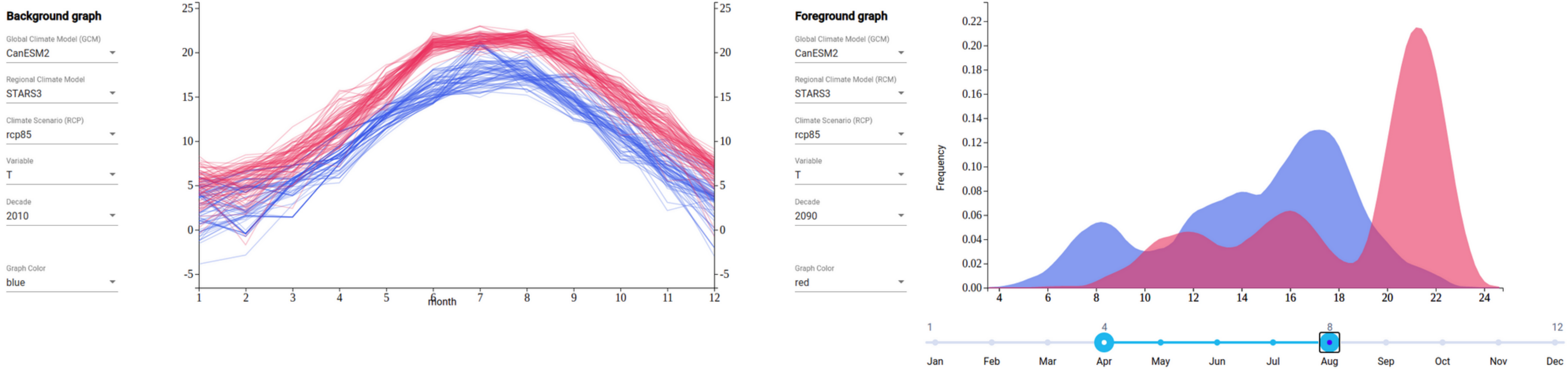
Program will start at 11:30:
Accessing Environmental Time Series Data for Data Analysis

52north
exploring horizons



Accessing Environmental Time Series Data for Data Analysis

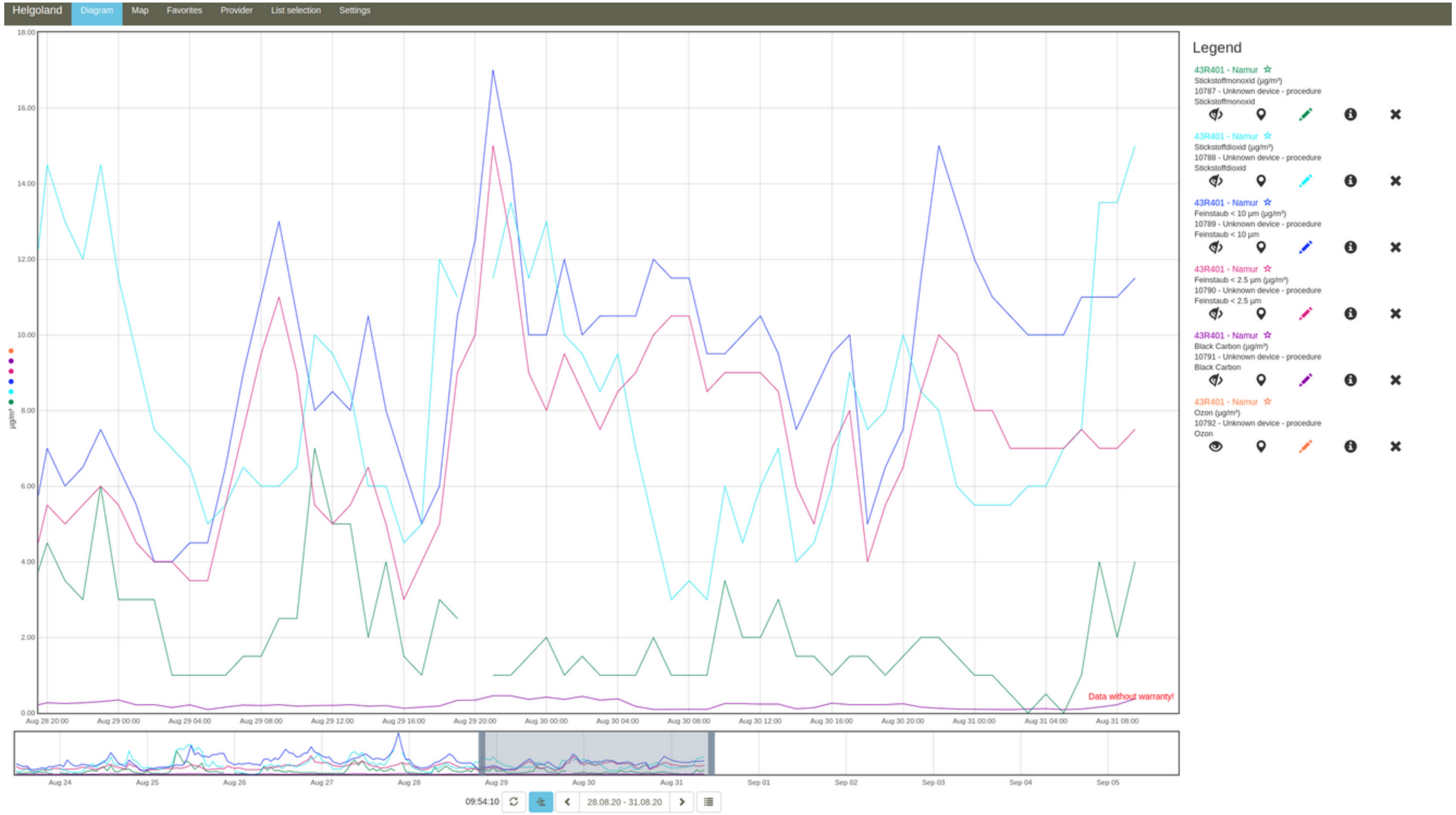
Benedikt Gräler, Andy Mc Kenzie (NIWA), Martin Pontius

Data access

- Data is generated in various Geospatial fields
- Stored in many flavours of data bases
- standardized APIs open up data silos
  - SOS
  - Sensorthings API
  - WFS, WCS, WMS
  - OGC APIs (Common, Features, Maps, Processes, Coverages, …)
- But also software internal (have a common “native” exchange format)

Data Visualization

- WebInterface (e.g. Helgoland)
- Local Clients
  - QGIS
  - ArcGIS
  - R
  - python



Climate model explorer

GCM

CanESM2

RCM

STARS3

RCP

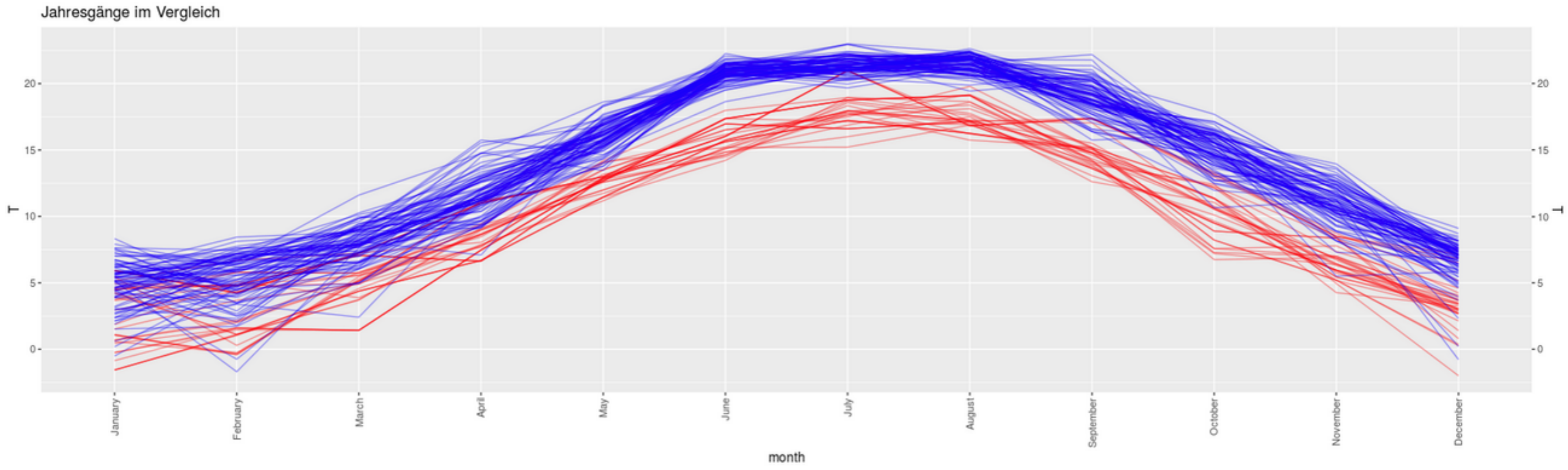
85

Variable

T

Dekade

2000



GCM

CanESM2

RCM

STARS3

RCP

85

Variable

T

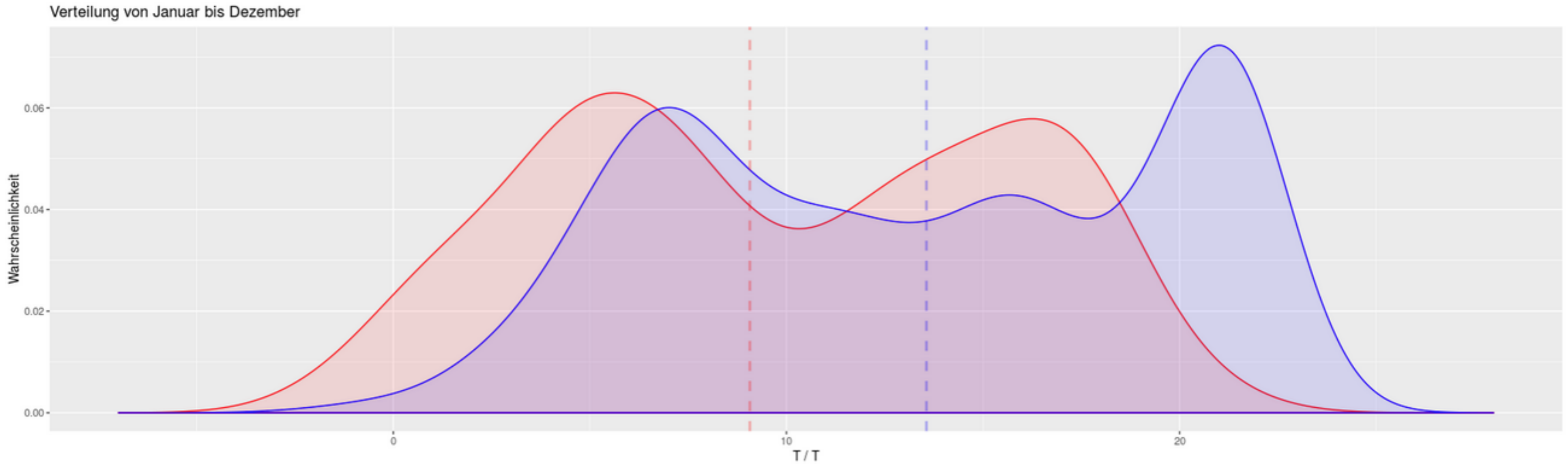
Dekade

2090

Auswahl von Monaten

1 12

☐ Invertiere Auswahl



Link: https://copulatheque.shinyapps.io/shinyruins/

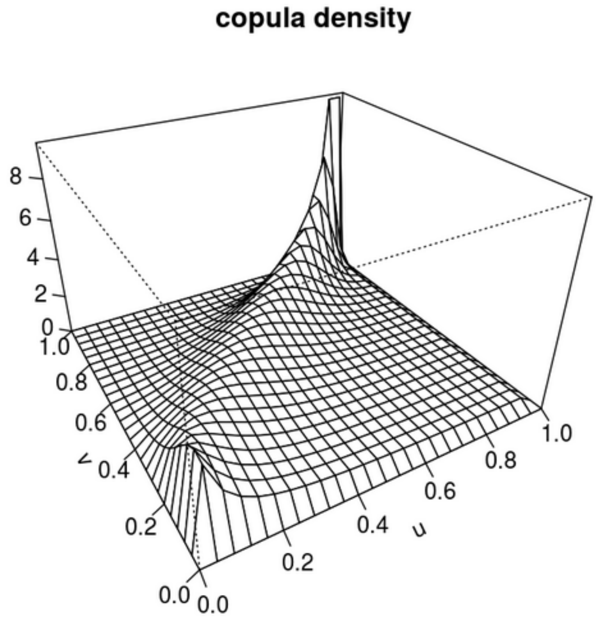
Data Analysis

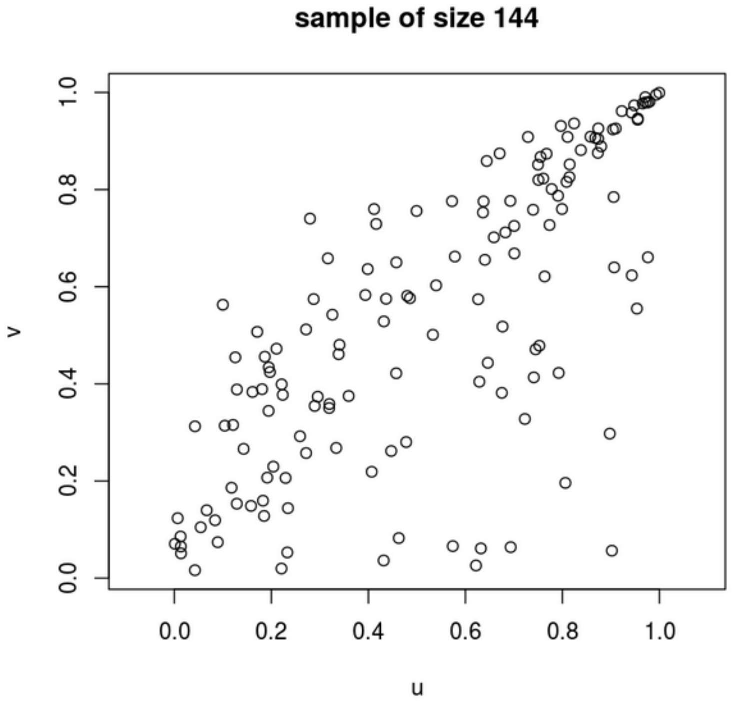
- enriches the value of the data
- helps to understand the observed phenomenons
- has many frameworks: R, python, …
- API clients are needed to ease the access for data scientists
  - sos4R
  - sos4py

Dependence of variables

- is key for prediction and interpolation
- such as correlation, regression models, …
- Probabilistic models

tawnT1Copula copula





(aka: strength of dependence)

Dependence properties

Kendall's tau: 0.5

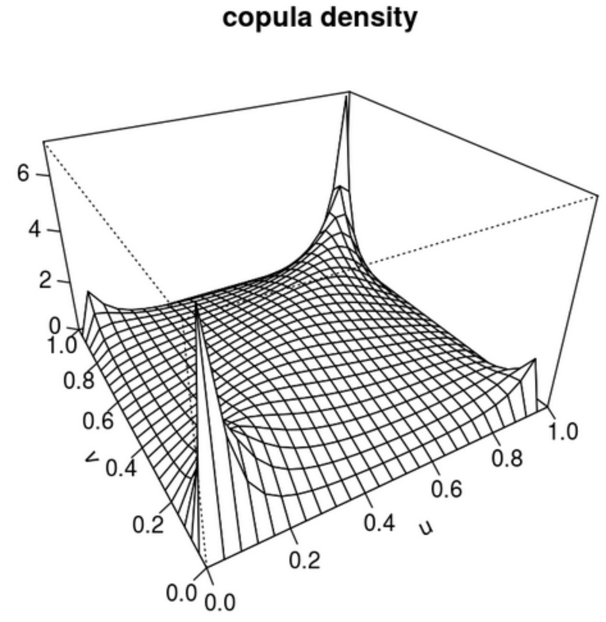
lower and upper tail dependence: 0, 0.58

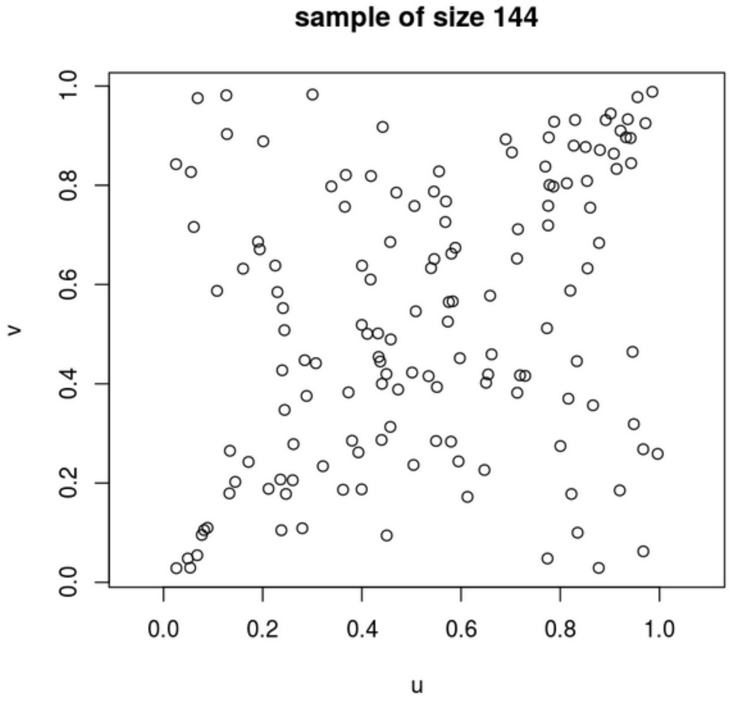
52north exploring horizons

Dependence of variables

- is key for prediction and interpolation
- such as correlation, regression models, …
- Probabilistic models

t-copula





(aka: strength of dependence)

Dependence properties

Kendall's tau: 0.28

lower and upper tail dependence: 0.43, 0.43

sos4R

- Implementation of SOS
- Additional “convenience” layer hiding the standard’s complexity
- (usually) on CRAN, use most recent version from https://github.com/52North/sos4R
- Demo and example will be given by Andy Mc Kenzie from NIWA

sos4py

- Recent Implementation of SOS
- Some “convenience” features hiding the standard’s complexity
- Part of pypi, use most recent version from https://github.com/52North/sos4py
- Demo and example will be given by Martin Pontius (52N)

exploring horizons

Accessing Environmental Time Series Data for Data Analysis

using sos4py

Martin Pontius, Benedikt Gräler, Alfredo Chavarria Vargas

Geospatial Sensing | Virtual 2020

Münster, 2020-08-31

Overview

1. Introduction

2. Software requirements

3. Demo – Jupyter notebook

Motivation

- SOS

  interoperable sharing of sensor data over the web

- Python

  taking advantage of python’s data analysis and visualization capabilities to exploit the full potential of your data

  → make data useful

Python SOS Client - owslib

- Github: https://github.com/geopython/OWSLib
- OGC web services library (wms, wfs, sos, …)
- Offers requests, parsers, get_observation(), …
- Some functions are missing
  - e.g. GetDataAvailability, GetFeatureOfInterest, functions for meta
- Not very convenient to use, especially for SOS non-experts

```
response = self.get_observation()             # xml as bytes
xml_tree = etree.fromstring(response)         # xml as tree
parsed_response = SOSGetObservationResponse(xml_tree)
# xml as observation response object
```

Python SOS Client - sos4py

- Github: https://github.com/52North/sos4py
- Inspired by sos4R
- Initiated during internship in May 2020 (Alfredo)
- Wraps owslib's SOS class and reuses other owslib-classes
- Adds functionality
  - `get_data_availabilty()`
  - `get_feature_of_interest()`
  - convenience functions (no/minimum knowledge of SOS required)
    - `get_data(), get_sites()`

Requirements

for running the demo jupyter notebook:

https://github.com/52North/sos4py/blob/master/examples/demo_data_access_plotting.ipynb

- Python (>=3.5 recommended)
- Jupyter Notebook
- Used python libraries
  - sos4py, pandas, matplotlib, seaborn, scipy, folium, contextily
  - some more libraries are implicitly needed through dependencies (e.g. geopandas)

DEMO

- Walk through jupyter notebook

  https://github.com/52North/sos4py/blob/master/examples/demo_data_access_plotting.ipynb

  - Metadata
  - Spatial data
  - Sensor data

52north exploring horizons

Thanks!

Martin Pontius

m.pontius@52north.org

Connecting to a SOS using the sos4R package: a short tutorial

Dr Andy McKenzie

31st August, 2020

Andy.McKenzie@niwa.co.nz

Climate, Freshwater & Ocean Science

NIWA
Taihoro Nukurangi

Some background

NIWA = National Institute of Water and Atmospheric Research

New Zealand based. Currently 9.30pm in New Zealand (yawn).

I work there as a population modeller. For example, estimating stock status for fish stocks.

But a broad focus on the use data science in the marine space, and I run the R users group at one of the sites.

sos4R background

Brent Wood (NIWA)[GIS + environmental data + money] +

52 north team (Benedikt Gräler, Daniel Nüst, Simon Jirka, Eike Jürrens).

→ upgrade older version of sos4R (easier to use functions + version 2.0 of SOS protocols)

→ Myself as the package crash-test dummy. In particular the SOS for some NIWA climate data

Focus of this short tutorial

- R users
- Getting data off the NIWA climate SOS
- Using the sos4R package to do this

Overview: getting data

NIWA has an SOS for some climate and hydrology data

Getting the data:

(a) Browser requests
(b) QGIS plugin: open source GIS program
(c) sos4R: an R package
(d) Use sos4R to make a Shiny app

NIWA Data on the Web

An overview and data available here (climate, water, marine, etc)

https://teamwork.niwa.co.nz/display/NEDA/NIWA+Environmental+Data+Access+through+standards+based+systems

Will concentrate on climate data via SOS (which is at monthly resolution)

https://climate-sos.niwa.co.nz

NIWA Climate Data Via SOS

What's there? A bunch of weather summaries from stations at a monthly resolution

https://climate-sos.niwa.co.nz/?service=SOS&version=2.0.0&request=GetCapabilities

```
▼<sos:Capabilities xmlns:sos="http://www.opengis.net/sos/2.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:ows="http://www.opengis.net/ows/1.1"
 xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:fes="http://www.opengis.net/fes/2.0" xmlns:swes="http://www.opengis.net/swes/2.0" xmlns:gml="http://www.opengis.net/gml/3.2"
 version="2.0.0" xsi:schemaLocation="http://www.opengis.net/sos/2.0 http://schemas.opengis.net/sos/2.0/sosGetCapabilities.xsd">
 ▼&lt;ows:ServiceIdentification&gt;
  ▼&lt;ows:Keywords&gt;
     &lt;ows:Keyword&gt;NIWA&lt;/ows:Keyword&gt;
```

```
▼&lt;swes:observableProperty&gt;
   MTHLY_STATS: DAYS OF OCCURRENCE (GUSTS over 23 knots) (MTHLY: GUST DAYS 24)
 &lt;/swes:observableProperty&gt;
▼&lt;swes:observableProperty&gt;
   MTHLY_STATS: DAYS OF OCCURRENCE (GUSTS over 32 knots) (MTHLY: GUST DAYS 33)
 &lt;/swes:observableProperty&gt;
▼&lt;swes:observableProperty&gt;
   MTHLY_STATS: DAYS OF OCCURRENCE (GUSTS over 50 knots) (MTHLY: GUST DAYS 51)
 &lt;/swes:observableProperty&gt;
```

NIWA Climate Data Via SOS

You can get the name and location for a feature of interest (e.g. the station 17244)

https://climate-sos.niwa.co.nz/?service=SOS&version=2.0.0&request=GetFeatureOfInterest&featureOfInterest=17244

Get the data available for a feature of interest (e.g. the station 17244)

https://climate-sos.niwa.co.nz/?service=SOS&version=2.0.0&request=GetDataAvailability&featureOfInterest=17244

Get the data for a feature of interest (17244), an observed property (MTHLY_STATS: EXTREME MAXIMUM TEMPERATURE (MTHLY: EXTR MAX TEMP) and a defined time period (e.g. 2018-06-01T00:00:00.000Z/2018-10-01T00:00:00.000Z)

https://climate-sos.niwa.co.nz/?service=SOS&version=2.0.0&request=GetObservation&featureOfInterest=17244&observedProperty=MTHLY_STATS:%20EXTREME%20MAXIMUM%20TEMPERATURE%20(MTHLY:%20EXTR%20MAX%20TEMP)&temporalFilter=om:phenomenonTime,2018-06-01T00:00:00.000Z/2018-10-01T00:00:00.000Z

Making all this easier

- The web browser syntax is pretty horrible (as is the output)
- Instead use sos4R package (or QGIS plug-in)
- And combine with a R Shiny interface to make it even easier (and prettier)

sos4R package: installation

From github repository for the latest developmental version

```
# install.packages("remotes")
remotes::install_github("52North/sos4R", ref = "dev")
```

Or from the usual R CRAN package repository (when the package is back there)

sos4R package: what it does

1. Explore phenomena (i.e. what climate data are there?)

2. Explore sites (i.e. what climate stations are there?)

3. Data download

4. Some GIS and time series plotting using other R packages

sos4R package: initial connection

```
library(sos4R)

mySos <- SOS(url = "https://climate-sos.niwa.co.nz",
             binding = "KVP",
             useDCPs = FALSE,
             version = "2.0.0")
```

sos4R package: phenomena

Basic summary of available types of climate data

```
phenomena <- phenomena(sos = mySos)

str(phenomena)
head(phenomena)
```

sos4R package: phenomena

What phenomena (i.e. climate data) are there?

```
phenomena <- phenomena(sos = mySos)
str(phenomena)
```

```
'data.frame':   84 obs. of  1 variable:
 $ phenomenon: chr  "MTHLY_STATS: DAYS OF DEFICIT (WBal AWC=150mm) (MTHLY: DAYS OF DEFICIT (WBAL))" "MTHLY_STATS: DAYS OF OC
CURRENCE (FOG) (MTHLY: FOG DAYS)" "MTHLY_STATS: DAYS OF OCCURRENCE (GALE) (MTHLY: GALE DAYS)" "MTHLY_STATS: DAYS OF OCCURREN
CE (GROUND FROST) (MTHLY: GROUND FROST DAYS)" ...
```

```
head(phenomena)
```

```
                                                                              phenomenon
1 MTHLY_STATS: DAYS OF DEFICIT (WBal AWC=150mm) (MTHLY: DAYS OF DEFICIT (WBAL))
2                       MTHLY_STATS: DAYS OF OCCURRENCE (FOG) (MTHLY: FOG DAYS)
3                     MTHLY_STATS: DAYS OF OCCURRENCE (GALE) (MTHLY: GALE DAYS)
4     MTHLY_STATS: DAYS OF OCCURRENCE (GROUND FROST) (MTHLY: GROUND FROST DAYS)
5   MTHLY_STATS: DAYS OF OCCURRENCE (GUSTS over 23 knots) (MTHLY: GUST DAYS 24)
6   MTHLY_STATS: DAYS OF OCCURRENCE (GUSTS over 32 knots) (MTHLY: GUST DAYS 33)
```

sos4R package: phenomena

What are the available climate data, and what time period do they cover?

```
phenomena <- phenomena(sos = mySos, includeTemporalBBox = TRUE)

head(phenomena)
```

sos4R package: phenomena

What climate data and what time period do they cover?

```
phenomena <- phenomena(sos = mySos, includeTemporalBBox = TRUE)
head(phenomena)
```

```
                                                                            phenomenon
1                              MTHLY_STATS: TOTAL RAINFALL (MTHLY: TOTAL RAIN)
2             MTHLY_STATS: WET DAYS with rainfall 1 mm or more (MTHLY: WET DAYS)
3      MTHLY_STATS: MEAN AIR TEMPERATURE;   0.5* (MAX + MIN) (MTHLY: MEAN TEMP)
4  MTHLY_STATS: MEAN MAXIMUM TEMPERATURE from daily Maxs (MTHLY: MEAN MAX TEMP)
5  MTHLY_STATS: MEAN MINIMUM TEMPERATURE from daily Mins (MTHLY: MEAN MIN TEMP)
6 MTHLY_STATS: MEAN DAILY GRASS-MIN from dly Grass-mins (MTHLY: MEAN GRASS-MIN)
   timeBegin    timeEnd
1 1960-08-01 2020-04-01
2 1960-08-01 2020-04-01
3 1960-08-01 2020-04-01
4 1960-08-01 2020-04-01
5 1960-08-01 2020-04-01
6 1960-08-01 2020-04-01
```

sos4R package: phenomena

Available data & time period. As before, but broken up by stations.

```
phenomena <- phenomena(sos = mySos,
                       includeTemporalBBox = TRUE,
                       includeSiteId = TRUE)
```

sos4R package: phenomena

What climate data and what time period do they cover, broken up by station.

```
phenomena <- phenomena(sos = mySos,
                       includeTemporalBBox = TRUE,
                       includeSiteId = TRUE)
head(phenomena)
```

```
                                                                     phenomenon
61  MTHLY_STATS: DAYS OF DEFICIT (WBal AWC=150mm) (MTHLY: DAYS OF DEFICIT (WBAL))
126 MTHLY_STATS: DAYS OF DEFICIT (WBal AWC=150mm) (MTHLY: DAYS OF DEFICIT (WBAL))
152 MTHLY_STATS: DAYS OF DEFICIT (WBal AWC=150mm) (MTHLY: DAYS OF DEFICIT (WBAL))
251 MTHLY_STATS: DAYS OF DEFICIT (WBal AWC=150mm) (MTHLY: DAYS OF DEFICIT (WBAL))
293 MTHLY_STATS: DAYS OF DEFICIT (WBal AWC=150mm) (MTHLY: DAYS OF DEFICIT (WBAL))
342 MTHLY_STATS: DAYS OF DEFICIT (WBal AWC=150mm) (MTHLY: DAYS OF DEFICIT (WBAL))
    siteID  timeBegin    timeEnd
61   17244 1999-08-01 2020-04-01
126  26958 2007-08-01 2020-04-01
152  25506 2004-12-01 2018-06-01
251  22719 2002-07-01 2018-11-01
293  37850 2011-11-01 2020-04-01
342  38224 2010-09-01 2020-04-01
```

sos4R package: sites

sites = climate stations

What climate stations are there with data? Note the sites output is a GIS object, a *spatial points data frame* → can input to spatial/GIS package functions (e.g. sp package)

```
sites <- sites(sos = mySos)
str(sites)
head(sites)
```

sos4R package: sites

```
str(sites)
```

```
Formal class 'SpatialPointsDataFrame' [package "sp"] with 5 slots
  ..@ data       :'data.frame': 60 obs. of  1 variable:
  .. ..$ siteID: chr [1:60] "1056" "11234" "12429" "12430" ...
  ..@ coords.nrs : num(0)
  ..@ coords     : num [1:60, 1:2] 174 173 173 174 169 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:2] "lon" "lat"
  ..@ bbox       : num [1:2, 1:2] -176.5 -77.8 177.9 -35.1
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : chr [1:2] "lon" "lat"
  .. .. ..$ : chr [1:2] "min" "max"
  ..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slot
  .. .. ..@ projargs: chr "+init=epsg:4326 +proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0"
```

```
head(sites)
```

```
            coordinates siteID
1     (173.926, -35.183)   1056
2   (172.851, -42.53433)  11234
3 (172.9716, -41.09798)  12429
4 (173.9628, -41.49891)  12430
5 (169.3148, -45.20724)  12431
6 (174.9844, -40.90392)  12442
```

sos4R package: sites

What climate stations are there? And what climate data do they have, and what time span does it cover?

```
sites_with_temporal_bbox <- sites(sos = mySos,
                                  includePhenomena = TRUE,
                                  includeTemporalBBox = TRUE)

# Pull out first three columns. Date displayed as seconds
head(sites_with_temporal_bbox@data[, 1:3])
```

sos4R package: sites

Data. First column = station. Following columns are types of climate data, with time period they're available for that station (time period in seconds!).

```
head(sites_with_temporal_bbox@data[, 1:3])
```

```
  siteID
1   1056
2  11234
3  12429
4  12430
5  12431
6  12442
  MTHLY_STATS: DAYS OF DEFICIT (WBal AWC=150mm) (MTHLY: DAYS OF DEFICIT (WBAL))
1                                                         376012800, 1585699200
2                                                         804556800, 1585699200
3                                                         799286400, 1585699200
4                                                         838857600, 1585699200
5                                                         870393600, 1346457600
6                                                         833587200, 1585699200
  MTHLY_STATS: DAYS OF OCCURRENCE (FOG) (MTHLY: FOG DAYS)
1                                   370742400, 815184000
2                                                     NA
3                                                     NA
4                                                     NA
5                                                     NA
6                                                     NA
```

sos4R package: data download

Download some climate data for a station and time interval

```
phenomena <- phenomena(sos = mySos)
# phenomena[18, 1] = "MTHLY_STATS: EXTREME MAXIMUM TEMPERATURE (MTHLY: EXTR MAX TEMP)"

library(parsedate)   # latter versions use as.POSIXct instead of parse_iso_8601
begin.date <- parse_iso_8601("1990-01-01")
end.date    <- parse_iso_8601("2000-01-02")

observationData <- getData(sos = mySos,
  phenomena = phenomena[18,1],
  sites = "1056",
  begin = begin.date,
  end     = end.date
  )

head(observationData)
```

sos4R package: data download

```
  siteID  timestamp
1   1056 1990-01-01
2   1056 1990-02-01
3   1056 1990-03-01
4   1056 1990-04-01
5   1056 1990-05-01
6   1056 1990-06-01
  MTHLY_STATS: EXTREME MAXIMUM TEMPERATURE (MTHLY: EXTR MAX TEMP)
1                                                            27.1
2                                                            28.5
3                                                            28.4
4                                                            24.2
5                                                            21.9
6                                                            21.3
```

sos4R package: useful GIS operations

Make a quick map (there’s lots of ways and packages)

```
# output is a spatial points data frame
sites <- sites(sos = mySos)

library(mapview)
mapview(sites, legend = FALSE, col.regions = "blue")
```

sos4R package: sites

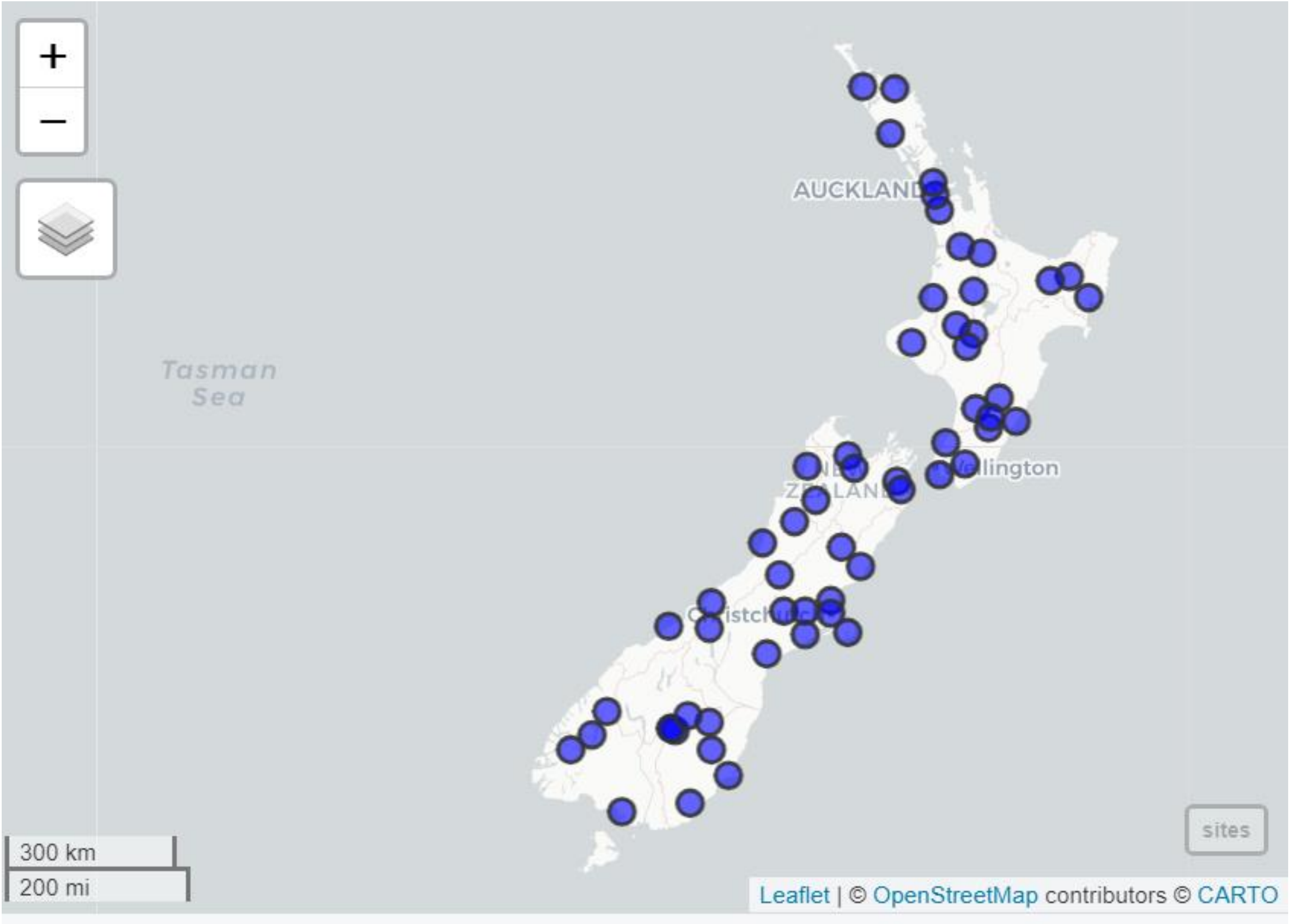
```
str(sites)
```

```
Formal class 'SpatialPointsDataFrame' [package "sp"] with 5 slots
  ..@ data       :'data.frame': 60 obs. of  1 variable:
  .. ..$ siteID: chr [1:60] "1056" "11234" "12429" "12430" ...
  ..@ coords.nrs : num(0)
  ..@ coords     : num [1:60, 1:2] 174 173 173 174 169 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:2] "lon" "lat"
  ..@ bbox       : num [1:2, 1:2] -176.5 -77.8 177.9 -35.1
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : chr [1:2] "lon" "lat"
  .. .. ..$ : chr [1:2] "min" "max"
  ..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slot
  .. .. ..@ projargs: chr "+init=epsg:4326 +proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0"
```

```
head(sites)
```

```
            coordinates siteID
1    (173.926, -35.183)   1056
2  (172.851, -42.53433)  11234
3 (172.9716, -41.09798)  12429
4 (173.9628, -41.49891)  12430
5 (169.3148, -45.20724)  12431
6 (174.9844, -40.90392)  12442
```

sos4R package: useful GIS operations



sos4R package: time series plots

Make a time series plot. Use the climate data from a few slides back (station 1056, extreme maximum temperature)

```
  siteID  timestamp
1   1056 1990-01-01
2   1056 1990-02-01
3   1056 1990-03-01
4   1056 1990-04-01
5   1056 1990-05-01
6   1056 1990-06-01
  MTHLY_STATS: EXTREME MAXIMUM TEMPERATURE (MTHLY: EXTR MAX TEMP)
1                                                            27.1
2                                                            28.5
3                                                            28.4
4                                                            24.2
5                                                            21.9
6                                                            21.3
```

sos4R package: time series plots

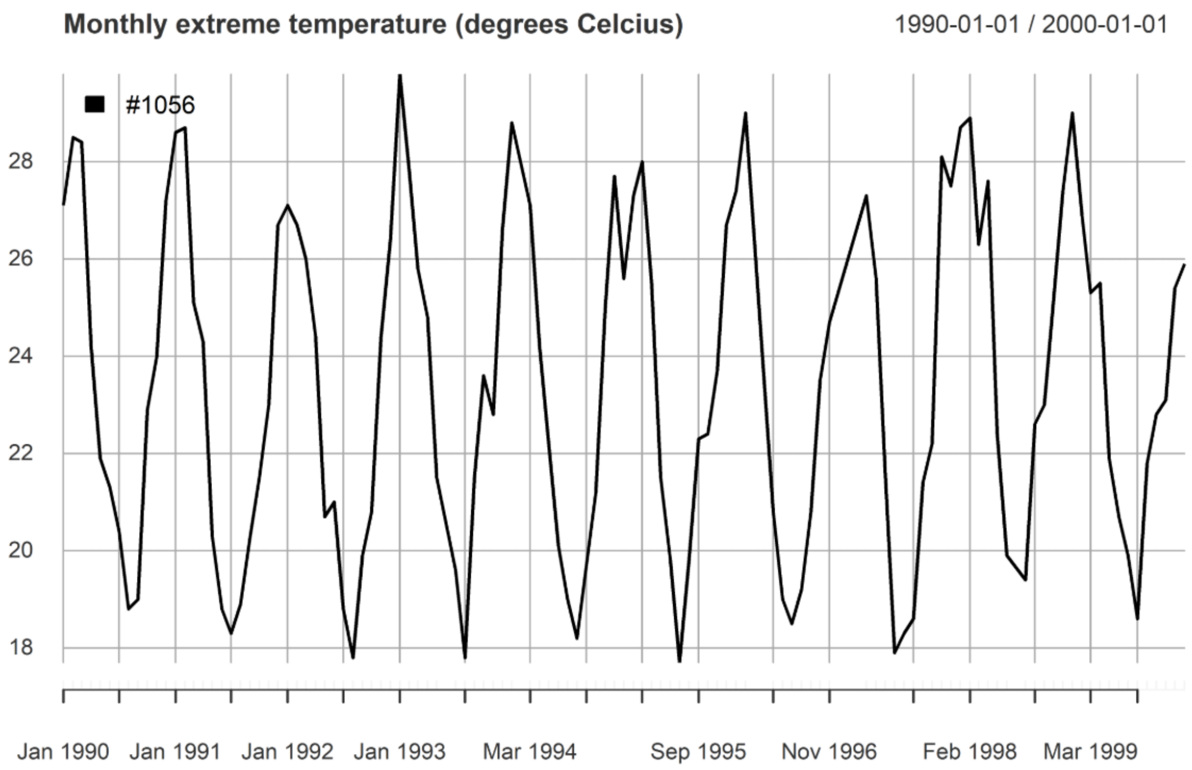
Make a time series plot. Use the climate data from a few slides back (station 1056, extreme temperature)

```
# Make a time series object. Third column has data in it
library(xts)
ts1056 <- xts(observationData[, 3], observationData$timestamp)
names(ts1056) <- "#1056"

plot(x = ts1056, main = "Monthly extreme temperature (degrees Celsius)",
    yaxis.right = FALSE, legend.loc = "topleft")
```

sos4R package: time series plots

Make a time series plot. Use the climate data from a few slides back (station 1056, extreme temperature)



The R Shiny Package

- Can relatively easily program up a “web site” in R
- Great for interactive data visualization (e.g. the climate data)
- Install the “shiny” package first

Start here for an introduction and tutorials

https://shiny.rstudio.com

R Shiny app with sos4R package: the code

About 300 lines of R code

Packages:

leaflet: easy to setup zoomable plots

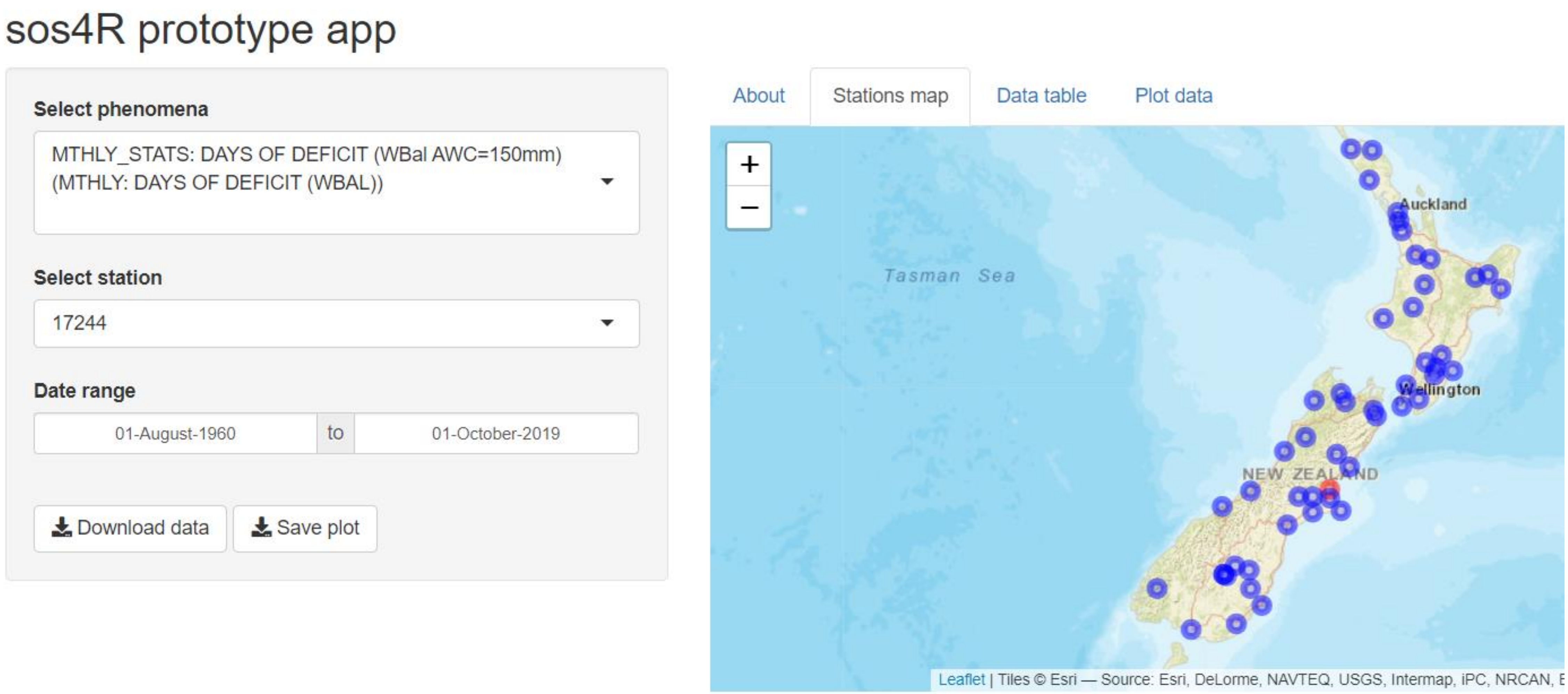
lubridate: very good at figuring out dates with just a few hints at the format (e.g ymd function).

dplyr, ggplot2: filter function, very nice plots

DT: better tables than the default

R Shiny App (very prototype)

https://niwa-apps.shinyapps.io/NIWAclimateSOSapp/



Contact details

Andy.McKenzie@niwa.co.nz

Feel free to get hold of me for questions or R code.

The End