How to write an academic paper using R Markdown

Ken Butler

August 30, 2016

Introduction

One of the principal assets of science is *reproducibility*: when you write a paper, or read one, you or anyone else should be able to reproduce what was done from the description given. In particular, if you have the same data as the original authors, you should be able to run the same analysis as they did, and get identical results. A different form of reproducibility is that of the *conclusions*: if the science stands up, you should be able to collect your own data, run the same analysis on your data, and get similar conclusions to the original authors (or be able to explain why not).

A similar issue involves making things reproducible *for yourself*: can you re-run your original analysis using updated data, or tweak things to consider some other factor, all the while leaving everything else the same?

The traditional way of writing a paper consists of grabbing the output you need from R Studio, and copying-and-pasting it into a Word document. But this is not reproducible: if anything changes, you have to find your code, re-run it (hoping you have the right code), and copy-paste again (hoping that you remembered to copy the right thing). The whole process is fraught with potential problems, and is itself *unscientific*. We can do much better.

In this Github repository, you will see all my files, including the output and the source. Have them open side by side, so you can see how to make a certain effect happen.

R Markdown

R Markdown is a "markup language": when you write your paper, you include not only the text, but also instructions for formatting that text, and then you process the R Markdown to obtain your formatted text. To make changes to how the formatted text appears, you go back and change the R Markdown and then process it again. This is unlike working in Word, where you see as you type what it will look like.

In R Studio, select File, New File and R Markdown. You can start a new project, or go to the one that has your data in it. This opens a template file that shows you some of the things you can do. (You can delete this file when you are done looking at it.) The top four or so lines, in between `--`, are called a YAML header. In your new document, this will show simple "meta-information" about the document like the author, title and date. We will see later that this is also where information about references goes.

You can choose an output format (which you can change later). HTML is good while you're writing, since is the quickest to re-run if you make changes. If you want PDF output, you will need to have LaTeX installed. Word output is probably best when you are basically happy with what you have and you're almost done.

The first thing you will want is to start a new section (called something like "Introduction"). Put the section heading on a line by itself (I like to leave blank lines before and after), and start the line with a `#`. To add a subsection, start the line with two of them, `##`. Should you wish to have a subsection of a subsection, start with three.

You may want to have some text in italics. To do that, surround the text with asterisks, *like this*. For bold, use two asterisks before and after, **in this fashion**.

Other things you might want to do are to link to another document, which you do by putting the link text in square brackets and following it immediately (no spaces) with the URL in brackets. You can include an image by the same sort of mechanism; for example the file `image.png` is included like this: `![](image.png)`. To include a mathematical formula, set it up as a LaTeX formula, and include as `$x^2+y^2$`, which looks like $x^2 + y^2$.

See the cheatsheet at ("R Markdown Cheat Sheet" 2015), which includes these ideas and more.

Producing the output document

At the top of the R Studio window in which you are composing your document, you will see a ball of wool, the word "Knit", and the document format you chose. Click this to produce the latest version of your document in your chosen format. When the document has finished processing, a previewer (HTML) or PDF viewer (PDF) or Word (.docx) will open with the processed version of your document. Review this for errors, or for changes that you want to make.

Remember, though, that the output document is the *final* stage of the chain. Make any actual changes in the `.Rmd` R Markdown document, and to see their effect, Knit the document again.

To change the output format, click the down-arrow to the right of Knit. This will display your choices. For me, this is HTML, PDF or Word. If you change it, you'll see that the Output: line in the YAML block at the top of your document has changed along with it.

Code and Code Chunks

Introduction

Perhaps the best feature of R Markdown is that you can include code in your document, and *the code will be run and the results inserted in your document.* This means the end of copying and pasting results to your paper. By way of example, let's create a small data frame and run a regression. To insert a code chunk, look for the green C icon with an arrow, to the left of Run at the top of your Markdown document. Click it (or type Control-Shift-I). This will insert an empty code chunk at the cursor. In the code chunk, place any R code you like, such as:

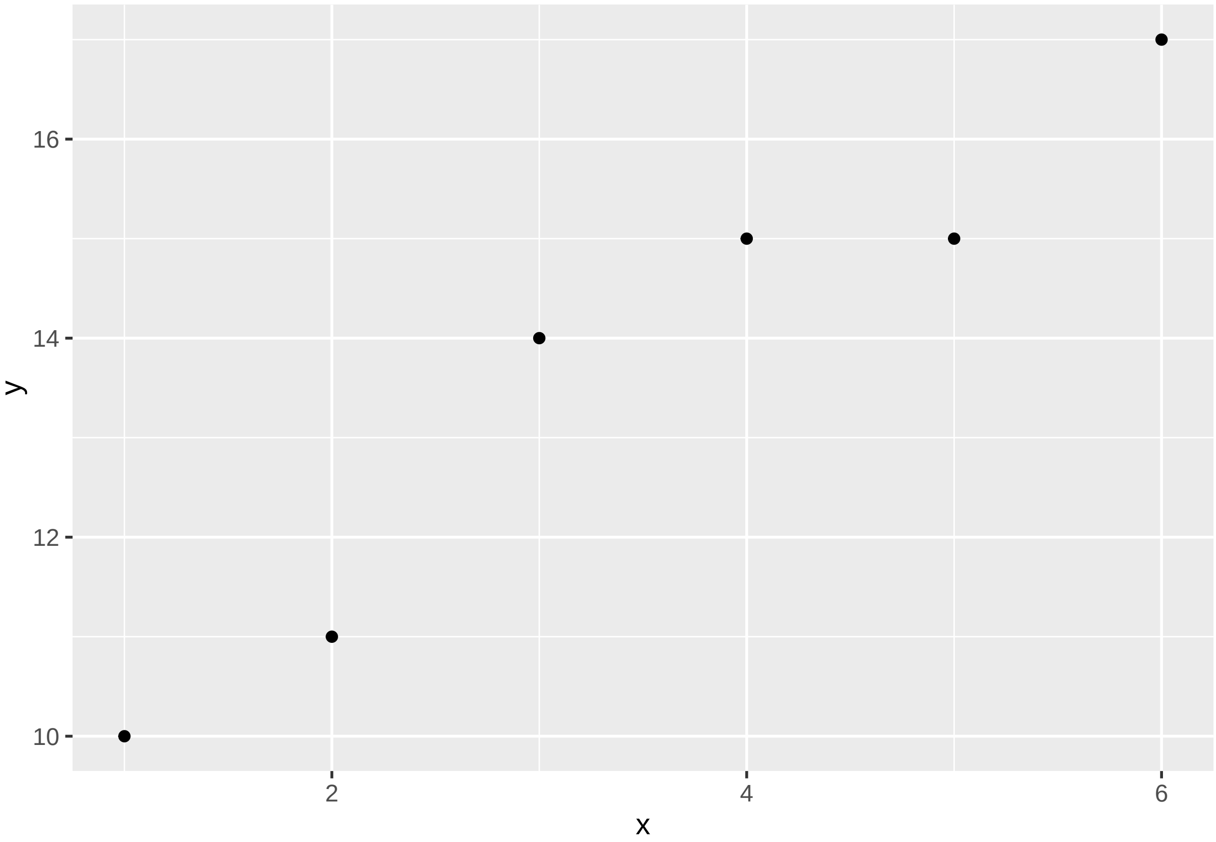
```r
x=1:6
y=c(10,11,14,15,15,17)
d=data.frame(x,y)
d
```

```
##   x  y
## 1 1 10
## 2 2 11
## 3 3 14
## 4 4 15
## 5 5 15
## 6 6 17
```

In the output, as you see, the value of the data frame `d` is shown.

Including graphs in output is no different than including text results, as we see:

```r
library(ggplot2)
ggplot(d,aes(x=x,y=y))+geom_point()
```



or we can do a regression:

```
y.1=lm(y~x,data=d)
summary(y.1)
```

```
##
## Call:
## lm(formula = y ~ x, data = d)
##
## Residuals:
##        1        2        3        4        5        6
## -0.23810 -0.60952  1.01905  0.64762 -0.72381 -0.09524
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   8.8667     0.7240  12.247 0.000255 ***
## x             1.3714     0.1859   7.377 0.001799 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7777 on 4 degrees of freedom
## Multiple R-squared:  0.9315, Adjusted R-squared:  0.9144
## F-statistic: 54.43 on 1 and 4 DF,  p-value: 0.001799
```

Including only some values from the output in text

A reference for these ideas is Xie (n.d.).

We may not need all that output; we may need only a couple of values from it. First, we figure out the values we need:

```
v=coef(y.1)
v
```

```
## (Intercept)           x
##    8.866667    1.371429
```

and then we include them in text using "backticks" (the symbol on the key below Esc on your keyboard) with `r` and then an expression inside:

The intercept of the regression line is 8.8666667 and the slope is 1.3714286.

This is better than copying the numbers into the text (for the same reason that calculating everything is better than copying and pasting: no possibility of error), because what if the intercept and slope change (perhaps because the data change) and we forget to change the text?

I might have wanted the P-value of the slope, rounded to 4 decimal places. First, I have to find out where it will be in `y.1`: it was actually in the `summary`:

```
names(summary(y.1))
```

```
##  [1] "call"          "terms"         "residuals"     "coefficients"
##  [5] "aliased"       "sigma"         "df"            "r.squared"
##  [9] "adj.r.squared" "fstatistic"    "cov.unscaled"
```

```
summary(y.1)$coefficients
```

```
##             Estimate Std. Error   t value     Pr(>|t|)
## (Intercept) 8.866667  0.7239661 12.247350 0.0002552248
## x           1.371429  0.1858973  7.377343 0.0017994572
```

That's the table that was in the output, and has the P-values in it. The one we want is in the second row and 4th column, so we need to do something like this:

The slope is significantly different from zero, with a P-value of 0.0018.

That is a bit unwieldy, so you can create a code chunk to calculate what you want to display first:

```
slope.pval=round(summary(y.1)$coefficients[2,4],4)
```

(we see in a moment how to hide this), and then say that the P-value is 0.0018.

Displaying, or not, the code and the output

The default is to display both the code and the output, with the output displayed in "typewriter" font. This may not be what you need. In a paper, there is probably no need to display the code, and some of the output might need to be calculated but not shown. Or you might want to display some code and talk about it, but not actually run it.

To not display the code, the magic word is `echo`. You put `echo=F` *inside the curly brackets at the top of the code chunk, after the* r: `{r, echo=F}`. To see the effect, compare these two code chunks (and go back to the source code to see how they were produced):

```
d
```

```
##   x  y
## 1 1 10
## 2 2 11
## 3 3 14
## 4 4 15
## 5 5 15
## 6 6 17
```

with

```
##   x  y
## 1 1 10
## 2 2 11
## 3 3 14
## 4 4 15
## 5 5 15
## 6 6 17
```

In the second chunk, the code `d` (to say what to print) does not appear.

To say not to evaluate a code chunk, the key word is `eval`. You might, for example, want to display some code without running it. I do not have a variable called `xx`:

```
xx
```

```
## Error in eval(expr, envir, enclos): object 'xx' not found
```

I might appear to be creating one here (but look at the R Markdown file):

```
xx=1:5
```

Does it exist?

```
xx
```

```
## Error in eval(expr, envir, enclos): object 'xx' not found
```

It doesn't, because the previous chunk had `eval=F` on it. If you looked at the R Markdown, you will see also that the chunks where I tried to display `xx` had an extra option `error=T` on them; this is to display the errors. Otherwise, when you try to Knit the file, it will stop at these errors. This is normally what you want: if there is an error, you want to fix it, not display it!

To control the output, use the option `results`. There are several possibilities here, but the most useful one for you is `'hide'`, which, as it suggests, does not display the output (though it does evaluate the code). Check the R Markdown to see what happened here:

```
coef(y.2)
```

```
## (Intercept)           x      I(x^2)
##    7.700000    2.246429   -0.125000
```

The first chunk (in the R Markdown) displays neither the output nor the code (thus it does not appear at all in the output). The second chunk displays the intercept and slopes (for x and x^2), but if you look only at the output it appears to have come from nowhere!

When you're writing a paper, you might want to start by displaying everything, so that you can follow the logic, and only when you're convinced it's correct do you start hiding any of it, so that the final paper contains only the relevant results and not any of the code.

Another thing you might want to do with output is to make it look nicer. You have a couple of choices here. The simplest is `kable` from the `knitr` package. Feed this a data frame, such as the one `d` that we made earlier:

```
library(knitr)
kable(d)
```

x	y

x	y
1	10
2	11
3	14
4	15
5	15
6	17

or the table of regression slopes and P-values:

```r
kable(summary(y.1)$coefficients)
```

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	8.866667	0.7239661	12.247350	0.0002552
x	1.371429	0.1858973	7.377344	0.0017995

`kable` will also take an option "digits" that gives a number of decimal places to use for each entry in the data frame:

```r
kable(summary(y.1)$coefficients,digits=4)
```

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	8.8667	0.7240	12.2473	0.0003
x	1.3714	0.1859	7.3773	0.0018

`kable` is simple by design. The next step up in complexity is `pander`, which is intelligent enough to print the output of a regression, or a data frame, or whatever you feed it:

```r
library(pander)
pander(y.1)
```

Table 4: Fitting linear model: y ~ x

	Estimate	Std. Error	t value	Pr(>|t|)
x	1.371	0.1859	7.377	0.001799
(Intercept)	8.867	0.724	12.25	0.0002552

```r
pander(y.2)
```

Table 5: Fitting linear model: y ~ x + I(x^2)

	Estimate	Std. Error	t value	Pr(>|t|)
x	2.246	0.9155	2.454	0.09137
I(x^2)	-0.125	0.128	-0.9764	0.4009
(Intercept)	7.7	1.399	5.503	0.01181

```
pander(d)
```

x	y
1	10
2	11
3	14
4	15
5	15
6	17

`pander` has a large number of options for customizing the output. But it tries to do something intelligent in any case.

If you wish to venture beyond this, you can investigate `xtable` as well. But these two should take care of most of your needs. I got most of this from Broman (n.d.). See also Humburg (2014).

In a paper, you probably want to number and refer to figures and tables. This doesn't seem to happen naturally in R Markdown; there is a mechanism called `bookdown` (Xie 2016) that incorporates these ideas. I talk about this later. In `bookdown`, you will probably want to use `kable` for your tables, since the mechanism for numbering and referencing those tables is built in.

Citations and references

One of the fundamental things in academic work is referencing the work of others who came before you, as the answer to the question "How do I know?". There are actually two parts to referencing: there is the **citation** in the text, which is a piece of text of the form "Butler (2016)", and there is a **reference**, which is in the list of references at the end, which enables the reader to translate a citation into something that can actually be found (online or in a library), such as "Butler, K. (2016) Something really quite interesting, *Journal of Interesting Things* 24 (9) 360–361".

The way to arrange this in R Markdown is to start with the reference. This needs to live in a file with extension `.bib` (in "BibTeX format", which is one standard way of collecting bibliographic information). My entry for my own (fake) article looks like this:

```
@article{butler16,
date={2016},
title={Something really quite interesting},
journal={Journal of Interesting Things},
volume={24},
number={9},
pages={360--361}
}
```

One way of citing websites is like this (when you know the name of the person writing the web page). You might know the date when the web page was written; if not, leave out the `date`:

```
@misc{humburg14,
date={2014},
title={Using knitr and pandoc to create reproducible reports},
author={Peter Humburg},
howpublished={\url{http://galahad.well.ox.ac.uk/repro/}},
note={Accessed: 2016-09-01}
}
```

The first line says what kind of thing it is you are citing: an `article` in a journal or a miscellaneous thing like a website or a `book`. Then an open curly bracket and then a "key", with a comma after it. The key is what you use to cite the item. After that come some lines that identify the whatever-it-is: the URL of a web page, the name of the author, the date a web page or article was published, the journal name, the volume (typically year) and number (which issue in a year) and page numbers. Each of these lines begins with what-it-is, an equals, the value inside curly brackets, and a comma to end the line. The last line has no comma, and after that a curly bracket by itself to balance the one at the beginning.

A BibTeX file has a list of these entries one after the other, as many as you like (in no particular order). See BibTeX (n.d.) for how to specify different kinds of citeable items.

Then go back to the YAML front matter of your R Markdown document, and add a line like

```
bibliography: myrefs.bib
```

where `myrefs.bib` is the name of your BibTeX file.

Now that you have the reference for each item that you wish to make a citation for, you can cite them in your R Markdown document. This is done by an at-sign followed by the key of the item you wish to cite (as it appears in the `.bib` file), as: see Butler (2016), or it is believed that some things are really quite interesting (Butler 2016). See "R Studio" (n.d.) for more details.

Finally, add an empty section to the end of your document called References or Bibliography or similar. This will be completed with references for the list of items you have cited. This seems to be hard-coded to go at the end, which isn't helpful if your journal requires your figures and tables to go at the end, or if you want to add an Appendix (which, in many journals, goes after the References). There can be other items in your `.bib` file, but if they're not cited, they won't go into the references. (Unless you want them to, which you can make happen.)

Where does that BibTeX information come from? You can type it yourself, which is tedious. Or you can use Google Scholar. Go to `scholar.google.ca` and type an author or title into the search box. I entered `gough jien`, hoping to turn up something I knew, and I got this:

Strangely enough, I do know a fair few of those papers!

References

BibTeX. n.d. "BibTeX Format Description." http://www.bibtex.org/Format/.

Broman, Karl. n.d. "Figures and Tables." http://kbroman.org/knitr_knutshell/pages/figs_tables.html.

Butler, Ken. 2016. "Something Really Quite Interesting." *Journal of Interesting Things* 24 (9): 360–61.

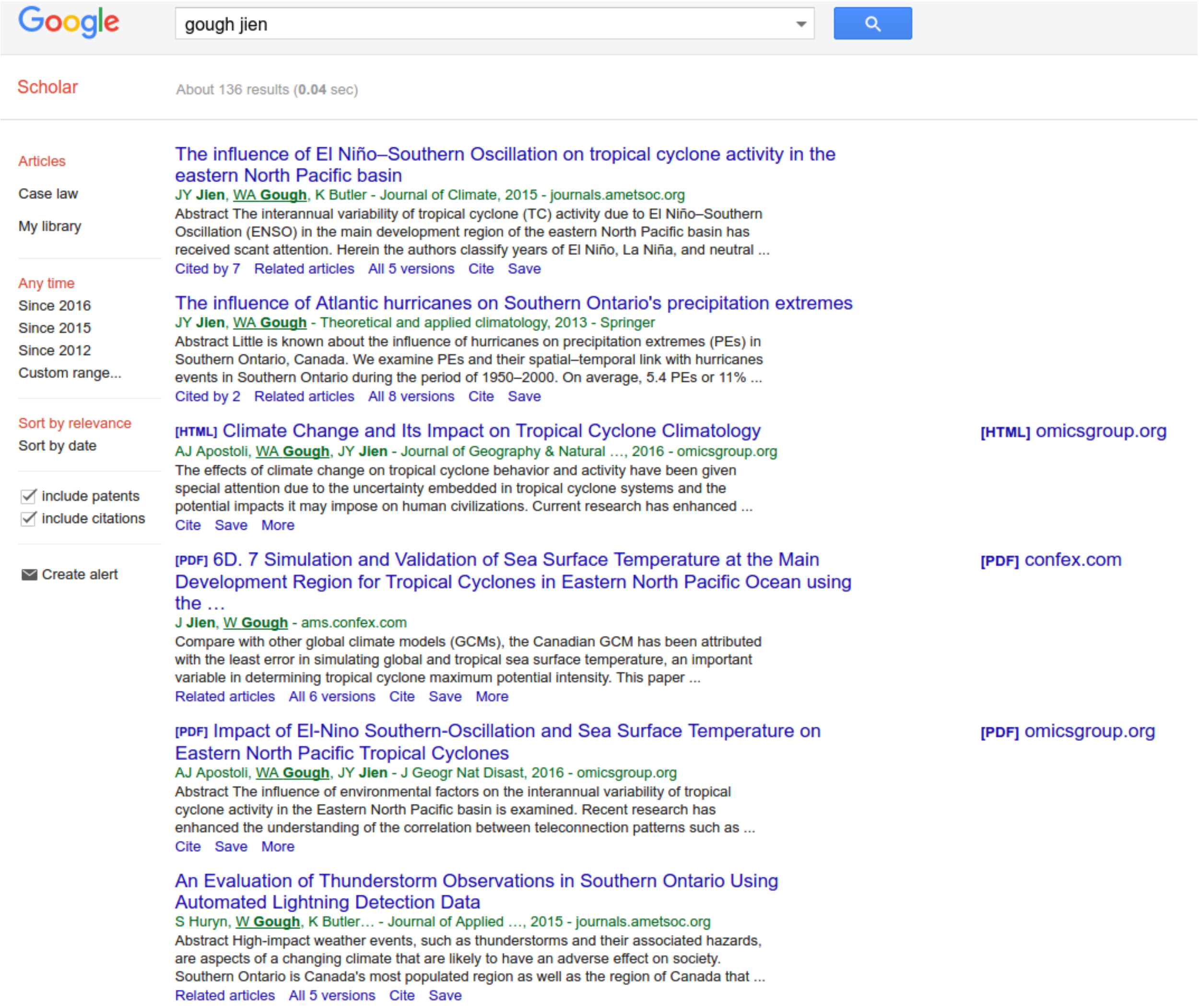
Humburg, Peter. 2014. "Using Knitr and Pandoc to Create Reproducible Reports." http://galahad.well.ox.ac.uk/repro/.

"R Markdown Cheat Sheet." 2015. https://www.rstudio.com/wp-content/uploads/2015/02/rmarkdown-cheatsheet.pdf.

"R Studio". n.d. "Bibliographies and Citations." http://rmarkdown.rstudio.com/authoring_bibliographies_and_citations.html.

Xie, Yihui. 2016. "Bookdown: Authoring Books with R Markdown." https://bookdown.org/yihui/bookdown/.

———. n.d. "Chunk Options and Package Options." http://yihui.name/knitr/options/.



About 136 results (**0.04** sec)

The influence of El Niño–Southern Oscillation on tropical cyclone activity in the eastern North Pacific basin
JY **Jien**, WA **Gough**, K Butler - Journal of Climate, 2015 - journals.ametsoc.org
Abstract The interannual variability of tropical cyclone (TC) activity due to El Niño–Southern Oscillation (ENSO) in the main development region of the eastern North Pacific basin has received scant attention. Herein the authors classify years of El Niño, La Niña, and neutral ...
Cited by 7 Related articles All 5 versions Cite Save

The influence of Atlantic hurricanes on Southern Ontario's precipitation extremes
JY **Jien**, WA **Gough** - Theoretical and applied climatology, 2013 - Springer
Abstract Little is known about the influence of hurricanes on precipitation extremes (PEs) in Southern Ontario, Canada. We examine PEs and their spatial–temporal link with hurricanes events in Southern Ontario during the period of 1950–2000. On average, 5.4 PEs or 11% ...
Cited by 2 Related articles All 8 versions Cite Save

[HTML] Climate Change and Its Impact on Tropical Cyclone Climatology
AJ Apostoli, WA **Gough**, JY **Jien** - Journal of Geography & Natural …, 2016 - omicsgroup.org
The effects of climate change on tropical cyclone behavior and activity have been given special attention due to the uncertainty embedded in tropical cyclone systems and the potential impacts it may impose on human civilizations. Current research has enhanced ...
Cite Save More

[HTML] omicsgroup.org

[PDF] 6D. 7 Simulation and Validation of Sea Surface Temperature at the Main Development Region for Tropical Cyclones in Eastern North Pacific Ocean using the …
J **Jien**, W **Gough** - ams.confex.com
Compare with other global climate models (GCMs), the Canadian GCM has been attributed with the least error in simulating global and tropical sea surface temperature, an important variable in determining tropical cyclone maximum potential intensity. This paper ...
Related articles All 6 versions Cite Save More

[PDF] confex.com

[PDF] Impact of El-Nino Southern-Oscillation and Sea Surface Temperature on Eastern North Pacific Tropical Cyclones
AJ Apostoli, WA **Gough**, JY **Jien** - J Geogr Nat Disast, 2016 - omicsgroup.org
Abstract The influence of environmental factors on the interannual variability of tropical cyclone activity in the Eastern North Pacific basin is examined. Recent research has enhanced the understanding of the correlation between teleconnection patterns such as ...
Cite Save More

[PDF] omicsgroup.org

An Evaluation of Thunderstorm Observations in Southern Ontario Using Automated Lightning Detection Data
S Huryn, W **Gough**, K Butler… - Journal of Applied …, 2015 - journals.ametsoc.org
Abstract High-impact weather events, such as thunderstorms and their associated hazards, are aspects of a changing climate that are likely to have an adverse effect on society. Southern Ontario is Canada's most populated region as well as the region of Canada that ...
Related articles All 5 versions Cite Save

Figure 1: