JAMIA Open, 00(0), 2021, 1–8
doi: 10.1093/jamiaopen/ooab007
Application Notes

Application Notes

Research and Exploratory Analysis Driven—Time-data Visualization (read-tv) software

John Del Gaizo, [1] Ken R. Catchpole, [2] and Alexander V. Alekseyenko[1,3,4,5]

[1]Biomedical Informatics Center, Department of Public Health Sciences, Medical University of South Carolina, Charleston, South Carolina, 29425, USA, [2]Department of Anesthesia and Perioperative Medicine, Medical University of South Carolina, Charleston, South Carolina, 29425, USA, [3]Department of Public Health Sciences, Medical University of South Carolina, Charleston, South Carolina, 29425, USA, [4]Department of Oral Health Sciences, Medical University of South Carolina, Charleston, South Carolina, 29425, USA, and [5]Department of Healthcare Leadership and Management, Medical University of South Carolina, Charleston, South Carolina, 29425, USA

Corresponding Author: John Del Gaizo, 22 West Edge St, Suite 2B, Charleston, South Carolina, 29403, USA; delgaizoj@musc.edu

Received 16 October 2020; Revised 14 January 2021; Editorial Decision 21 January 2021; Accepted 4 February 2021

ABSTRACT

Motivation: Research & Exploratory Analysis Driven Time-data Visualization (*read-tv*) is an open source R Shiny application for visualizing irregularly and regularly spaced longitudinal data. *read-tv* provides unique filtering and changepoint analysis (CPA) features. The need for these analyses was motivated by research of surgical work-flow disruptions in operating room settings. Specifically, for the analysis of the causes and characteristics of periods of high disruption-rates, which are associated with adverse surgical outcomes.
Materials and Methods: *read-tv* is a graphical application, and the main component of a package of the same name. *read-tv* generates and evaluates code to filter and visualize data. Users can view the visualization code from within the application, which facilitates reproducibility. The data input requirements are simple, a table with a time column with no missing values. The input can either be in the form of a file, or an in-memory data-frame– which is effective for rapid visualization during curation.
Results: We used *read-tv* to automatically detect surgical disruption cascades. We found that the most common disruption type during a cascade was training, followed by equipment.
Discussion: *read-tv* fills a need for visualization software of surgical disruptions and other longitudinal data. Every visualization is *reproducible*, the exact source code that *read-tv* executes to create a visualization is available from within the application. *read-tv* is *generalizable*, it can plot any tabular dataset given the simple requirements that there is a numeric, datetime, or datetime string column with no missing values. Finally, the tab-based architecture of *read-tv* is easily *extensible*, it is relatively simple to add new functionality by implementing a tab in the source code.
Conclusion: *read-tv* enables quick identification of patterns through customizable longitudinal plots; faceting; CPA; and user-specified filters. The package is available on GitHub under an MIT license.

Key words: longitudinal visualization, change-point analysis, change point analysis, changepoint analysis, forecasting, R, Shiny, surgical safety

LAY SUMMARY

We developed Research & Exploratory Analysis Driven Time-data Visualization (*read-tv*), an open source web application for visualizing time-series data. More specifically, event data. Our application provides unique filtering and plotting capabilities. It also incorporates changepoint analysis (CPA) techniques, or the detection of where in a series the event rate changes. While *read-tv* was originally developed for the research of surgical work-flow disruptions in operating room settings, this tool can be applicable to many forms of time-data.Our use case dataset consisted of 1,900+ surgical disruptions form 41 surgical cases. We wanted to find which disruption types are the most common during cascades, or periods of high disruption rates, as high rates are associated with negative surgical outcomes. Using *read-tv*, we identified an appropriate CPA method that finds cascades by detecting where the disruption rate changes. We then wrote code that identified the most common disruption types that were present during the cascades. This was relatively easy to perform as we had the changepoint markers output from *read-tv*. Nearly half of the disruptions during cascades (48%) can be categorized as either training or equipment disruptions. This suggests that a focus on reducing training and equipment disruptions may have a high-impact on improving patient outcomes.

BACKGROUND AND SIGNIFICANCE

Longitudinal modeling analyses are widely employed among biomedical disciplines. Common techniques include visualization, and algorithms for changepoint analysis (CPA). Interpretation of the results often involves collaboration between computational practitioners and domain experts. Although it is difficult for a research tool to simultaneously serve both user types; there is a need to facilitate collaborative interactions, and ultimately the quality of their research products.

The tool is more likely to be adopted by other academic labs if it (1) supports their domain-specific data; (2) is open source and transparent; (3) provides a graphical UI for non-technical (subject domain expert) users; and (4) is built in a widely-used programming language, such as R. Several open-source R tools provide intuitive interfaces and plotting capabilities: esquisse[1], ggplotgui[2], ggThemeAssist[3], SLIDER[4], and TumGrowth[5]. However, they do not provide CPA, or interpolation functions to regularize irregularly-spaced event data.

While data cleaning and curation are vital components of data science projects, we did not aim to implement a tool that performs these steps. There are already powerful R tidyverse[6] packages that provide this functionality. However, we did desire a tool that is written in R, so that it can readily inter-operate with these packages; which enables rapid development and visualization as the data is curated. Therefore, the tool complements, but not substitutes, the role of a data scientist on a research project.

Changepoint Analysis (CPA)

Changepoint Analysis (CPA) refers to a set of a methods and algorithms that find where in a sequence, often a time-series, that a change occurs. Specifically, CPA algorithms try to find where a summary statistic, θ (e.g. mean or variance), changes value. Most CPA algorithms assume that the data is regularly spaced in time. Since events typically occur at irregular time intervals, *read-tv* provides functionality to regularize event data.

We did not develop novel CPA techniques, but incorporated 4 algorithms from the R `changepoint` package[7]: (1) **AMOC** (at most one change)[8], which uses a likelihood ratio test to find at most a single point where the statistic changes. (2) **BINSEG** (binary segmentation) finds multiple changepoints through hierarchical clustering[9,10]. (3) **SEGNEIGH** (Segment Neighborhoods) finds an exact solution. In comparison, BINSEG finds an approximate solution that may not be the global minimum[11]. SEGNEIGH employs dynamic programming to reduce computationally time, but it still scales quadratically with sequence length. (4) **PELT** (pruned exact linear time)[11] employs optimal partitioning techniques[12], and a pruning algorithm to filter changepoint candidates. Like SEGNEIGH, PELT has the advantage of finding an exact solution; but with relatively low computational costs that approach linearity with sequence length given certain data assumptions.[7]Killick et al[7,11,13] provide informative descriptions of these CPA algorithms.

Motivating Example: Surgical Flow Disruptions

read-tv was originally designed as a surgical safety research tool for analyzing surgical work-flow disruption (FD) sequences. There is compelling evidence that FD sequences are informative indicators of error causation[14–26]. In addition, the data collection is repeatable[27], and the analyses are quantitative[28–31]. Joseph et al.[19] identify the following adverse outcomes that increase with high FD rates: patient mortality[22], procedure duration[23,24], procedure errors[20,21], OR member stress[25], and perceived workload[26]. Additionally, a snowballing effect may occur, where a series of FDs leads to a high rate of subsequent disruptions, or cascades[32,33]. Collection of these data is via direct observation; which requires careful training of observers, and close attention to sampling and coding strategies[34] to balance data quality with the cognitive demands of data collection. A set of analytical tools are needed to explore both the quality of the observational method, and a wide variety of contextual, process, and outcome effects for exploring and improving systems safety in surgical procedures. Doing so serves to enhance the overall reliability and utility of this data collection approach by encouraging a standardized data format for event recording, while facilitating powerful and flexible analyses.

Therefore, we developed *read-tv* to simultaneously show the relationships between time, surgical case, event (disruption) occurrence/rate, and event properties such as surgical phase and disruption type. The surgical-disruption data includes (1) timestamp; (2) disruption type (communication, coordination, patient factors, etc.)[35], and (3) observer notes. Certain recent datasets also include a severity metric. The user can explore different CPA techniques to detect changes in FD rates, and hence detect cascades. Current FD analysis is often performed with spreadsheets, which leaves advanced visualization and CPA analyses largely out of reach.

However, we realized that this application can support a wide audience outside the domain of surgical safety. This generalizability is further described in later sections.

OBJECTIVES

The main objectives for *read-tv* are largely portrayed in Figures 1–3, and are listed as follows.

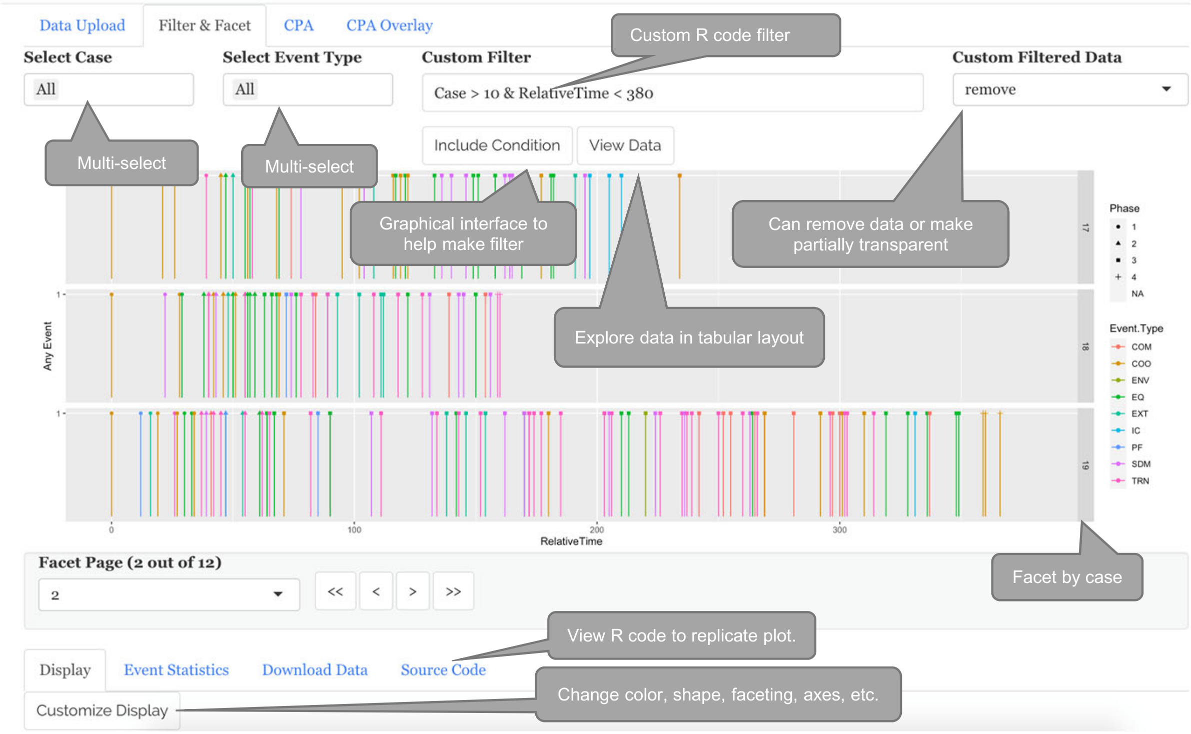
- Web-support:
  - *read-tv* can be installed on a server so that is accessible to non-technical team members through the web interface.
- Event plots (Figure 1):
  - For this category of longitudinal plots, the x-axis and y-axis correspond to time and event presence, respectively.
    - The y-axis is 1 if there was any event at the given time, otherwise 0.
    - *read-tv* creates a column, "Any Event", which is simply 1 for each event and is plotted by default.
- Simple, tabular data input
  - Each row is an event or disruption, and each column is a property such as Time, Case, etc.
  - Supported file types are CSV, TSV, and RDS.
  - The user may pass in-memory dataframe structures from the R console if *read-tv* is on a local computer, which facilitates rapid analysis.
- Data filter (Figure 1):
  - *read-tv* supports custom user-specified filters in R, which can also be generated through the GUI interface.
  - The filter should remove data from both the visualization and downstream analyses.
- Facet (Figure 1):
  - We needed each surgical case to have a subplot.
  - Any column with 500 unique values or less can be used as a facet column.
- Facet pagination (Figure 1):
  - This lets the user specify the number of subplots per page so that each subplot has sufficient space.
  - The pages can be scrolled.
- Color and shape (Figure 1):
  - Events can be color or shape coded by properties such as disruption type.
  - Any column may be used for color coding.
  - Any column with 6 unique values or less can be used for shapes.
- Source code generation (Figure 2)
  - A user can view the R source code that *read-tv* generates to create an associated plot.
- Changepoint analyses (CPA)[36](Figure 3)
  - Most events are irregularly spaced in time, i.e. inter-event time intervals are not consistent.
  - *read-tv* provides options to regularize the data, by counting the number of events at fixed intervals within a user-specified range. This can then be input into CPA algorithms.
  - *read-tv* enables the user to verify common CPA algorithms through output visualization.

MATERIALS AND METHODS

Design Approach

read-tv is a Shiny[37] application that is distributed as a component of a package of the same. A how-to readme document and tutorial are publicly available: https://github.com/JDMusc/read-tv (last accessed January 04, 2021). It is developed as a package to facilitate installation and maintainable code organization.

The application is divided into tabs: Data Upload, Filter & Facet, CPA, and CPA Overlay (Figures 1–3). The tab outputs flow from left to right; in other words, the tabs on the left do not know about the tabs on the right. The CPA tab performs CPA on the out-



Select Case: All

Select Event Type: All

Custom Filter: Case > 10 & RelativeTime < 380

Custom Filtered Data: remove

Facet Page (2 out of 12): 2

Custom R code filter

Multi-select

Multi-select

Graphical interface to help make filter

Can remove data or make partially transparent

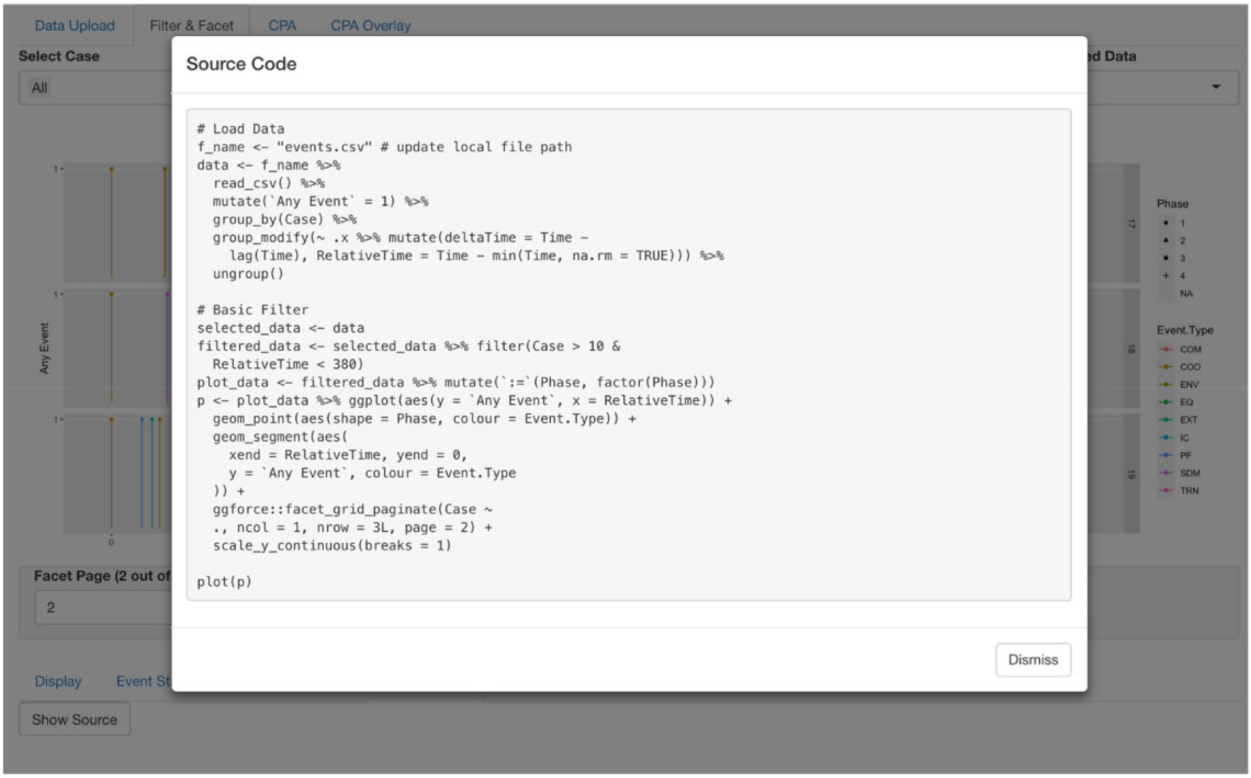
Explore data in tabular layout

Facet by case

View R code to replicate plot.

Change color, shape, faceting, axes, etc.

Figure 1. Flow Disruption by Time after Surgery Start, Faceted. *The filter and facet tab of read-tv supports faceting (Case), custom filters (Case > 10 and Relative Time < 380), different event colors (Event Type), shape (Phase), source code visualization, and other features shown above. This filtered and faceted data is passed to the CPA tab. The tabs to the right receive data from the tabs to the left.*



```
# Load Data
f_name <- "events.csv" # update local file path
data <- f_name %>%
  read_csv() %>%
  mutate(`Any Event` = 1) %>%
  group_by(Case) %>%
  group_modify(~ .x %>% mutate(deltaTime = Time -
    lag(Time), RelativeTime = Time - min(Time, na.rm = TRUE))) %>%
  ungroup()

# Basic Filter
selected_data <- data
filtered_data <- selected_data %>% filter(Case > 10 &
  RelativeTime < 380)
plot_data <- filtered_data %>% mutate(`:=`(Phase, factor(Phase)))
p <- plot_data %>% ggplot(aes(y = `Any Event`, x = RelativeTime)) +
  geom_point(aes(shape = Phase, colour = Event.Type)) +
  geom_segment(aes(
    xend = RelativeTime, yend = 0,
    y = `Any Event`, colour = Event.Type
  )) +
  ggforce::facet_grid_paginate(Case ~
  ., ncol = 1, nrow = 3L, page = 2) +
  scale_y_continuous(breaks = 1)

plot(p)
```

Figure 2. Source Code Generation. *Every read-tv visualization is the result of code generated through meta programming techniques in response to user input. The code that creates the plot can be visualized by pressing the "Show Source" button. The code above will recreate the plot in Figure 1.*

put of the Filter & Facet. The CPA Overlay tab overlays the CPA output changepoints on top of the raw data from Data Upload. CPA Overlay also provides support for filtering and faceting data. However, the data filtered in CPA Overlay will not alter CPA tab's changepoints (which are calculated from Filter & Facet's output). This can be beneficial for seeing if certain event types are near changepoints, because event types not in question can be removed from the CPA Overlay tab's visualization.

*read-tv*utilizes the changepoint[7,11,13] R package and the tidyverse[6] set of packages. These packages provide an infrastructure that supports reproducible visualizations, data input that is generalizable to diverse datasets, and an extensible graphic interface.

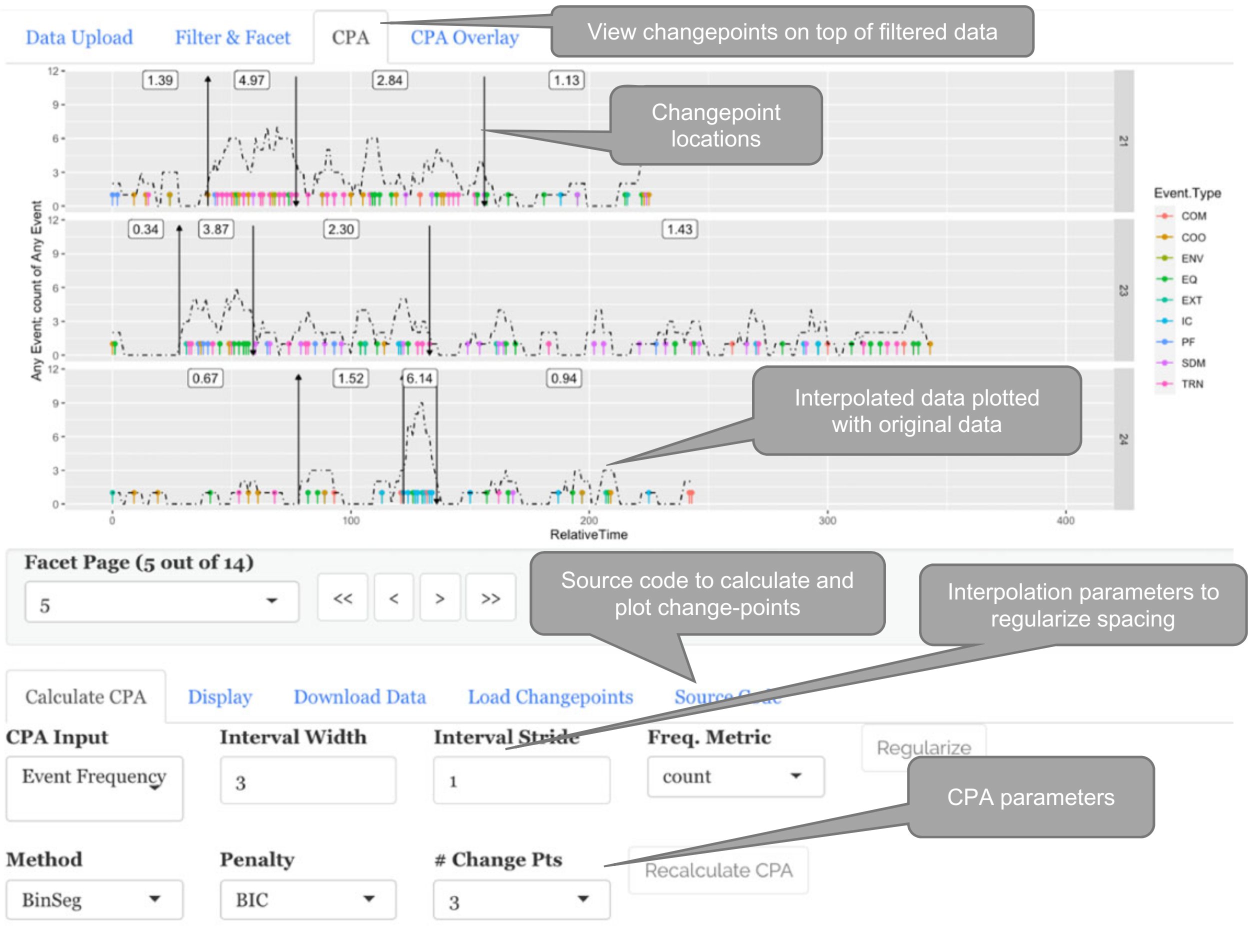
Reproducibility

It can be challenging to document the steps to recreate a GUI-based visualization. Therefore, *read-tv* uses metaprogramming to generate code in response to user input. The code is evaluated to create visualizations. The user can view the code from *read-tv*'s "Source Code" option (Figure 1, 2). Therefore, any visualization can be reproduced by executing this code within an R environment.

Generalizability

Though originally designed for surgical data, the application can be generalized for a wide range of longitudinal analyses through the principles of data simplicity and flexibility:

- Data simplicity:
  - *read-tv*ingests tabular data, with the only requirement that there is at least one column for time. Most data scientists know how to convert a given domain-specific model to this format.
  - Two optional data columns are Case and Event Type.
    - The Case column is used to calculate a relative time property, which is time since first event in the given case.
    - *read*-tv looks for Case and Event Type for default multi-select controls (Figure 1).
    - If these columns are absent, the user has an option to map them to other columns through a modal input.
  - If the column is neither provided nor mapped, then it's multi-select will not be available (Figure 1).
  - If the Case column is neither provided nor mapped, then the relative time property will be calculated from the earliest point in the data.
- Data flexibility:
  - Time column:
    - The column for time does not need to be labeled as "Time", the user can specify the column through a modal window.
    - The data format for the time column can be a datetime string, datetime (if the data is an R object), or numeric—such as number of minutes.
    - Most of the common string formats are supported as *read-tv* uses the tidyverse "readr" package to load data, which has functions that automatically detect most time formats.
- Flexible feature support:
  - As described in the objectives section, different columns can be used for faceting, filtering, color coding, or shape coding of events.



View changepoints on top of filtered data

Changepoint locations

Interpolated data plotted with original data

Facet Page (5 out of 14): 5

Source code to calculate and plot change-points

Interpolation parameters to regularize spacing

CPA parameters

CPA Input: Event Frequency

Interval Width: 3

Interval Stride: 1

Freq. Metric: count

Method: BinSeg

Penalty: BIC

Change Pts: 3

Figure 3. CPA Tab. *The CPA tab provides options to regularize timing intervals, prepare data for CPA input, and execute CPA algorithms. This is the same dataset as Figure 1, we just removed the custom filter in the "filter and facet tab" before calculating the changepoints.*

Extensibility

read-tv has an *extensible* tab-based architecture. To add a custom tab to the application, a developer only has to modify 2 - 5 files in the original code base:

- *R/mainDisplayUI.R*
  - https://github.com/JDMusc/READ-TV/blob/master/R/mainDisplayUI.R (last accessed January 04, 2021)
  - The developer would add a line to instantiate and specify a name for the new tab.
- *R/mainDisplayServer.R*
  - https://github.com/JDMusc/READ-TV/blob/master/R/mainDisplayUI.R (last accessed January 04, 2021)
  - The developer would add a few lines to instantiate the tab's logic, gain access to data and functionality from the previous tabs, and expose data or functionality that is output from the tab.

The following need to be modified to (1) import from another package, or (2) add sample data.

- *DESCRIPTION*
  - https://github.com/JDMusc/READ-TV/blob/master/DESCRIPTION (last accessed January 04, 2021)
  - This file lists dependencies and function imports.
- data folders
  - These folders contain sample data.
- *data*
  - https://github.com/JDMusc/READ-TV/tree/master/data (last accessed January 04, 2021)
- *data-raw*
  - https://github.com/JDMusc/READ-TV/tree/master/data-raw (last accessed January 04, 2021)

Sample Data Sets

The read-tv package includes 3 sample data sets:

- *covid_usa*
  - NY Times data of daily new COVID-19 infections and deaths for each US state , 1/21/2020 - 9/21/2020[38].
- *covid_global*
  - "Our World in Data" dataset of daily COVID-19 current cases, new cases, and deaths by country[39].
- *japan_eq_3_11*
  - USGS provided seismic recordings from Japan during the Great Tohoku Earthquake in 2011[40], ±1 day.

Motivating Example

As mentioned in Background and Significance, the project's motivating example was surgical disruption analysis. We used *read-tv* to develop a CPA model that quickly identifies surgeries where the

Table 1. Cascade disruption types and patterns

Event Type	In-Cascade Count	Disruption Count	$P(cascade \vert ET)$	$P(ET)$	$P(ET \vert cascade)$	$P(ET \vert notcascade)$
TRN	117	376	0.31	0.19	0.27	0.17
COM	64	278	0.23	0.14	0.15	0.14
ENV	4	18	0.22	0.01	0.01	0.01
SDM	32	148	0.22	0.07	0.07	0.08
IC	26	126	0.21	0.06	0.06	0.06
EQ	90	460	0.20	0.23	0.21	0.24
PF	11	58	0.19	0.03	0.03	0.03
COO	71	385	0.18	0.20	0.17	0.20
EXT	13	125	0.10	0.06	0.03	0.07
Any Event	**428**	**1974**	**0.22**	**1.0**	**1.0**	**1.0**

ET stands for Event Type. The disruption type abbreviations are as follows: TRN, Training; COM, Communication; ENV, Environment; SDM, Surgical Decision Making; IC, Instrument Change; EQ, Equipment; PF, Patient Factors; COO, Coordination; EXT, External. The bottom row is bolded as the sum of above rows.

disruption rate increases for a period of time, or a cascade. Figure 3 shows the CPA tab with specific CPA parameters. We defined a cascade as a period with an event rate of at least 1 event per minute over a 3-minute window (Figure 3). The cascade analysis source code can be found at: https://github.com/JDMusc/sugery-analysis (last acessed January 04, 2021).

Our dataset consisted of 41 RAS (robot assisted surgery) cases and 1,900+ disruptions. The RAS cases came from 4 sites and span a variety of surgical types. The FDs are categorized into 9 types, 7 from previous research[35]: (1) communication, (2) coordination, (3) equipment, (4) external factors, (5) instrument changes, (6) surgical decision making, (7) training: and 2 additional types: environment and patient factors.

RESULTS

read-tv ingests relatively simple, tabular data where each row is an event. The data must have a time column without missing values. *read-tv* calculates 2 new columns: (1) Relative Time, the time since earliest event (within each case if a case column is provided), and (2) a column of all ones, Any Event. By default, *read-tv* produces simple stem plots of Any Event by Relative Time. However, the user can customize the x and y-axes, plot size, number of items per page for faceted plots, and map columns to colors or shapes. *read-tv* also supports data filtering through a graphical interface and with R queries (Figure 1). The Source Code sub-tab displays code to recreate the plot (Figure 2).

Motivating Example

We found 25 cascades out of 41 surgeries; that 23 surgeries (56.1%) had 1 cascade, and that 1 surgery (2.4%) had 2 cascades; and that the training disruption type was the most common trigger of a cascade, followed by communication and equipment factors (7, 5, and 5 out of 25, respectively); the main FD type that is overrepresented during a cascade is training (Table 1). Training disruptions account for more than one-fourth of cascade disruptions (27%), 19% of general disruptions, and only 17% of non-cascade disruptions. As high disruption rates are correlated with negative outcomes, system changes that focus on decreasing training disruptions may have the highest impact on patient safety. As stated earlier, this analysis can be found at: https://github.com/JDMusc/surgery-analysis

DISCUSSION

Data visualization serves as an interdisciplinary communication bridge between quantitative scientists and domain experts. *read-tv* can (1) create *reproducible* visualizations through source code generation, (2) be *extended* through custom tab addition, and (3) be *generalized* to different datasets.

It was simple to both (1) heuristically set the CPA parameters for sample cases with the *read-tv* interface for exploring CPA plots, and (2) verify the validity of the CPA model simultaneously for many cases with faceted and paginated plots. Based on the literature reports that high disruption rates are associated with negative outcomes[20–26], an algorithm that automatically detects and filters cases with high rates can help surgical safety researchers know which cases merit focus in order to identify system-level patterns that lead to error.

We ensured that *read-tv* was suitable for the exemplar purpose of analyzing flow disruptions in surgical care by developing it in close collaboration with the research team members who perform the observational data collection. The functional flexibility of the tool has substantially facilitated the range and speed with which analyses can be conducted. For example, we are in-process of publishing a separate inter-rater analysis. *read-tv* allows the research team to ascertain each individual event, faceted by rater. We can quickly compare the ratings (either severity or disruption type) for a series of disruptions, and heuristically inspect inter-rater reliability. This complements the more traditional approach of comparing summary statistics, by providing a granularity and rigor to disruption assessment that is usually unavailable.

LIMITATIONS AND FUTURE WORK

The code was written by one person, under helpful guidance from mentors, and feedback from lab members. While the application was tested on surgical data, and the main developer tested it on some non-surgical data, we hope to receive feedback on GitHub as it is tested on a broad and diverse collection of datasets. We aim to turn *read-tv* into a community supported, open-source project.

The main future work is the incorporation of forecasting techniques into the application, which is currently under development.

CONCLUSION

read-tv is an open-source, web application developed in R that provides features that fill a current research tool gap in longitudinal modeling. It has been positively received by surgical safety end users in our lab, and it can be applied to a wide array of use cases including general, urological, cardiac and cancer surgeries. As well as non-surgical medical data, and even general longitudinal data.

ACKNOWLEDGEMENTS

None.

DATA AVAILABILITY

There were four mentioned datasets, they are all accessible in public repositories.

- *covid_usa*
    - NY Times data of daily new COVID-19 infections and deaths for each US state , 1/21/2020 - 9/21/2020[38].
    - https://raw.githubusercontent.com/nytimes/covid-19-data/master/us-states.csv (last accessed December 2020)
- *covid_global*
    - "Our World in Data" dataset of daily COVID-19 current cases, new cases, and deaths by country[39].
    - https://ourworldindata.org/coronavirus-source-data (last accessed December 2020)
- *japan_eq_3_11*
    - USGS provided seismic recordings from Japan during the Great Tohoku Earthquake in 2011[40], ±1 day.
    - https://earthquake.usgs.gov/fdsnws/event/1/query.csv?starttime=2011-03-10%2000:00:00&endtime=2011-03-12%2023:59:59&maxlatitude=45.919&minlatitude=26.861&maxlongitude=149.414&minlongitude=129.023&minmagnitude=2.5&orderby=time (last accessed December 2020)
- *PHI-Free Disruptions*[41]
    - https://github.com/JDMusc/surgery-analysis/blob/master/data/events_phi_free.csv (last accessed February 04, 2021)
    - DOI : 10.5061/dryad.d51c5b02g

AUTHORS' CONTRIBUTIONS

The main author, John Del Gaizo, developed the software. Dr. Alexander Alekseyenko managed development through weekly meetings and feature requests. Dr. Kenneth Catchpole also provided feedback, and his lab shared the observational data of surgical flow disruptions.

CONFLICT OF INTEREST STATEMENT

None declared.

FUNDING

South Carolina Translational Research (SCTR) institute TL1 training grant: TL1 TR001451, UL TR001450.This project was funded under grant number HS026491-01 from the Agency for Healthcare Research and Quality (AHRQ), U.S. Department of Health and Human Services (HHS). The authors are solely responsible for this document's contents, findings, and conclusions, which do not necessarily represent the views of AHRQ. Readers should not interpret any statement in this report as an official position of AHRQ or of HHS.

REFERENCES

1. Fanny Meyer and Victor Perrier. esquisse: Explore and Visualize Your Data Interactively [program], 2020. R package version 0.3.0. https://CRAN.R-project.org/package=esquisse
2. Gert Stulp. ggplotgui: Create Ggplots via a Graphical User Interface [program], 2017. R package version 1.0.0. https://CRAN.R-project.org/package=ggplotgui
3. Calli Gross and Phillip Ottolinger. ggThemeAssist: Add-in to Customize 'ggplot2' Themes [program], 2016. R Package Version 0.1.5. https://cran.r-project.org/package=ggThemeAssist
4. Davis Vaughan. silder: Sliding Window Functions [program], 2020. R Package Version 0.1.5. https://cran.r-project.org/package=slider
5. Enot DP, Vacchelli E, Jacquelot N *et al.* TumGrowth: An open-access web tool for the statistical analysis of tumor growth curves. *Oncoimmunology*. 2018; 7 (9): e1462431.
6. Wickham H, Averick M, Bryan J *et al.* Welcome to the Tidyverse.*J. Open Source Softw*.2019; 4(43): 1686.
7. Killick R, Eckley IA. changepoint: an R package for changepoint analysis.*J Stat Soft*.2014 2014; 58(3): 1–19.
8. Hinkley DV. Inference about the change-point in a sequence of random variables.*Biometrika*1970; 57(1): 1–17.
9. Scott AJ, Knott M. A cluster analysis method for grouping means in the analysis of variance. *Biometrics* 1974; 30(3): 507–512.
10. Edwards AWF, Cavalli-Sforza LL. A method for cluster analysis.*Biometrics*.1965; 21(2): 362–375.
11. Killick R, Fearnhead P, Eckley IA. Optimal detection of changepoints with a linear computational cost.*J Am Stat Assoc*. 2012; 107(500): 1590–1598.
12. Jackson B, Scargle JD, Barnes D, *et al.* An algorithm for optimal partitioning of data on an interval.*IEEE Signal Process Lett*.2005; 12(2): 105–108.
13. Eckley IA, Fearnhead P, Killick R. *Analysis of Changepoint Models Bayesian Time Series Models*. Cambridge, UK; New York: Cambridge University Press; 2011, pp. 205–25.
14. Shouhed D, Gewertz B, Wiegmann D, *et al.* Integrating human factors research and surgery a review.*Arch Surg*. 2012; 147(12): 1141–146.
15. Catchpole K, Perkins C, Bresee C, *et al.* Safety, efficiency and learning curves in robotic surgery: a human factors analysis. *SurgEndosc*. 2016; 30(9): 3749–3761.
16. Cohen TN, Cabrera JS, Sisk OD, *et al.* Identifying workflow disruptions in the cardiovascular operating room. *Anaesthesia*.2016; 71(8): 948–954.
17. Parker SE, Laviana AA, Wadhera RK, *et al.* Development and evaluation of an observational tool for assessing surgical flow disruptions and their impact on surgical performance.*World J Surg*.2010; 34(2): 353–361.
18. Catchpole K, Ley E, Wiegmann D, *et al.*A human factors subsystems approach to trauma care. *JAMA Surg*. 2014; 149(9): 962–968.
19. Joseph A, Khoshkenar A, Taaffe KM, *et al.* Minor flow disruptions, traffic-related factors and their effect on major flow disruptions in the operating room. *BMJ QualSaf*.2019; 28(4): 276–283.
20. Wiegmann DA, Elbardissi AW, Dearani JA, *et al.* Disruptions in surgical flow and their relationship to surgical errors: an exploratory investigation. *Surgery*.2007; 142(5): 658–665.
21. Catchpole KR, Giddings AE, de Leval MR, *et al.* Identification of systems failures in successful paediatric cardiac surgery.*Ergonomics*.2006; 49(5–6): 567–588.
22. deLeval MR, Carthey J, Wright DJ, *et al.* Human factors and cardiac surgery: a multicenter study. *J. Thoracic Cardiovasc. Surg.* 2000; 119(4): 661–672.
23. Gillespie BM, Chaboyer W, Fairweather N. Factors that influence the expected length of operation: results of a prospective study. *BMJ QualSaf*.2012; 21(1): 3–12.
24. Zheng B, Martinec DV, Cassera MA, *et al.*A quantitative study of disruption in the operating room during laparoscopic antireflux surgery.*SurgEndosc*. 2008; 22(10): 2171–2177.
25. Wheelock A, Suliman A, Wharton R, *et al.* The impact of operating room distractions on stress, workload, and teamwork.*AnnSurg*.2015; 261(6): 1079–1084.
26. Weigl M, Antoniadis S, Chiapponi C, *et al.* The impact of intra-operative interruptions on surgeons' perceived workload: an observational study in elective general and orthopedic surgery.*SurgEndosc*. 2015; 29(1): 145–153.
27. Parker S, Laviana A, Wadhera R, *et al.* Development and evaluation of an observational tool for assessing surgical flow disruptions and their impact on surgical performance. World J Surg. 2010; 34: 353–361.
28. Shao X, Zhong X, Li J, *et al.* Bottleneck analysis to reduce surgical flow disruptions: theory and application. *IEEE Trans Automat Sci Eng*.2015; 12(1): 127–139.

29. Catchpole K, Bisantz A, Hallbeck MS, *et al.* Human factors in robotic assisted surgery: lessons from studies in the wild. *Appl Ergonomics.* 2019; 78: 270–276.
30. Shouhed D, Blocker R, Gangi A, *et al.* Flow disruptions during trauma care. *World J Surg.* 2014; 38(2): 314–21.
31. Weigl M, Stefan P, Abhari K, *et al.* Intra-operative disruptions, surgeon's mental workload, and technical performance in a full-scale simulated procedure.*SurgEndosc.* 2016; 30(2): 559–566.
32. Woolf SH, Kuzel AJ, Dovey SM, *et al.*A string of mistakes: the importance of cascade analysis in describing, counting, and preventing medical errors. *Ann Fam Med.* 2004; 2(4): 317–26.
33. Woods D, Patterson E. *How unexpected events produce an escalation of cognitive and coordinative demands. Stress Workload and Fatigue*. Hillsdale, NJ: Lawrence Erlbaum; 2004, pp. 290–304.
34. Catchpole K, Neyens DM, Abernathy J, *et al.* Framework for direct observation of performance and safety in healthcare.*BMJ QualSaf.*2017; 26(12): 1015–1021.
35. Catchpole KR, Hallett E, Curtis S, *et al.* Diagnosing barriers to safety and efficiency in robotic surgery. *Ergonomics.*2018; 61(1): 26–39.
36. Taylor W A. *Change-Point Analysis: A Powerful New Tool for Detecting Changes*. Deerfield, IL: Baxter Healthcare Corporation; 2000.
37. Chang W, Cheng J, Allaire JJ, et al. *shiny: Web Application Framework for R*; 2021. R package version 1.6.0. https://CRAN.R-project.org/package=shiny
38. Smith M, Yourish K, Almukhtar S, *et al.*Data from The New York Times, based on reports from state and local health agencies. *The New York Times*. 2020. https://github.com/nytimes/covid-19-data Accessed January 4, 2021.
39. Hasell J, Mathieu E, Beltekian D, *et al.* A cross-country database of COVID-19 testing.*Sci Data.* 2020; 7(1): 345.
40. USGS Interactive Earthquake Map: United States Geologic Survey; 2020 [Available from: https://earthquake.usgs.gov/earthquakes/map/]. Accessed January 4, 2021.
41. Del Gaizo J, Alekseyenko A, Catchpole K. Dataset for Research and Exploratory Analysis Driven – Time-data Visualization (read-tv) Software. 2020. https://github.com/JDMusc/surgery-analysis/blob/master/data/events_phi_free.csv. Accessed February 2, 2021.