# calcIVT

## calcIVT

### Description

The calcIVT package is intended to provide functions useful in calculating Intermediate Value Tables (IVTs) from a time series dataset. The calcIVT package has been designed to facilitate the analysis of high resolution time interval datasets by establishing typical patterns found within the data (either for distinct phases of the data or for the overall dataset) and detecting anomalous signatures. The package accomplishes this by creating a series of sequential snapshots of the data with summary statistics for either distinct phases within the data, or for the overall dataset in its entirety.

This package was created in an effort to analyze flight data in order to better identify abnormalities as they arise in the data. Because a typical flight is composed of several very different phases, this package allows allows for the data to be subsetted by discrete phases so that abnormalities can be identified at specific phases of the flight rather than trying to treat the data as a single continuous entity. Data lacking these discrete phases can still be handled using the package by simply setting the discrete phases to 1.

### Getting Started

This guide is structured with the assumption that you are beginning with a basic understanding of R, and aims to provide an understandable blueprint for implementing the calcIVT functionality within your own data.

The first thing you will need to do is install the calcIVT package. You will need access to the [martingale repository](http://martingale.pnl.gov/computing/repos/) in order to obtain the package. After permissions for access have been granted, the package can then be installed by:

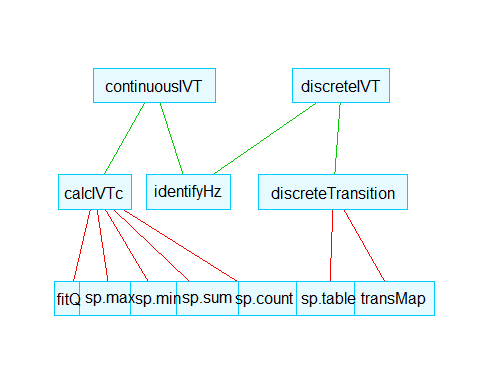
install.packages("calcIVT", repos="http://martingale.pnl.gov/computing/repos")

Once the package has been installed, the library can be loaded so that the functions can be used. A complete list of the functions contained within the calcIVT package can be seen using:

library(calcIVT)  
help(package=calcIVT)

### Package Structure

The graphic below provides a heirarchical depiction of how the functions contained within the calcIVT package are related.



### continuousIVT() Example

The continuousIVT() function is intended to calculate IVTs for multiple continuous variables. These IVTs are important because they can be used in conjunction with several analysis methods as a means to identify anomalies within the data. Let's look at an example.

Suppose you have a set of dataset that contains measurements of the continuous variables speed, altitude, and the tail rudder angle of a particular flight. These measurements are take once per second and associated with the corresponding phase of the flight; in this case the phases are take-off, flight, and landing. We will create our fake dataset as follows:

#Flight Variables#  
  
# vector containing speed at each second of the flight  
speed <- c(rnorm(600,60,5),rnorm(6000,300,10),rnorm(1800,60,5))   
  
# vector containing altitutde at each second of the flight  
altitude <- c(rep(10,600),rnorm(6000,20000,1000),rep(10,1800))   
  
# vector containing tail rudder angle of the plane at each second of the flight  
trAngle <- rnorm(8400,0,20)   
  
# integer vector identifying phase at each second of the flight   
#(1 = Take off, 2 = Flight, 3 = Landing)  
phase.vec <- c(rep(1,600),rep(2,6000),rep(3,1800))

Now we can define some of the function parameters that we will need in order to use the continuousIVT() function.

#Function Parameters  
  
# vector with names of flight variables  
cNames <- c("speed","altitude","trAngle")   
  
# vector identifying unique phases of the flight  
phases.to.process <- c(1,2,3)   
  
# Define number of seconds in the flight  
seconds <- 8400

Remember that you can see a list of arguments required for the continuousIVT() function using:

?continuousIVT

It should be noted that there are additional arguments nominal.skip, nominal.bw, and actual.min.max however these arguments have default values so we are going to leave them alone. Additional information on these can be found within the help document.

Now we can use the continuousIVT() function to obtain the IVTs for each of the continuous variables.

cResultIVT <-continuousIVT(cNames, phase.vec, phases.to.process, seconds)  
cResultIVT

## speed altitude trAngle  
## p.1.a.total 36126 14869 146  
## p.1.b.total 120 9928 14  
## p.1.c.total 21 1797 -5  
## p.1.d.total 3182 22443 11935  
## p.1.a.ss 2189623 72289617 38553  
## p.1.b.ss 3404 22266116 1861  
## p.1.c.ss 117 699414 290  
## p.1.d.ss 29952 101449471 250328  
## p.1.a.min 45 10 -58  
## p.1.b.min -3 0 -5  
## p.1.c.min 0 0 -3  
## p.1.d.min 2 0 6  
## p.1.a.max 74 10 68  
## p.1.b.max 33 2834 9  
## p.1.c.max 6 459 2  
## p.1.d.max 66 5200 30  
## p.1.n.count 600 600 600  
## p.1.start.value 58 10 6  
## p.1.end.value 65 10 1  
## p.2.a.total 1800902 119956541 -2930  
## p.2.b.total -2 150 1  
## p.2.c.total -42 -3535 1  
## p.2.d.total 57732 5856699 116514  
## p.2.a.ss 540689294 2399677703735 490303  
## p.2.b.ss 11879 98166781 20883  
## p.2.c.ss 908 8510723 2790  
## p.2.d.ss 614436 6250124016 2411027  
## p.2.a.min 264 16182 -91  
## p.2.b.min -34 -2739 -6  
## p.2.c.min -6 -525 -2  
## p.2.d.min 3 265 6  
## p.2.a.max 334 23563 66  
## p.2.b.max 32 2784 7  
## p.2.c.max 1 148 2  
## p.2.d.max 68 5864 42  
## p.2.n.count 6000 6000 6000  
## p.2.start.value 281 19035 -4  
## p.2.end.value 311 21378 -16  
## p.3.a.total 108088 28219 -572  
## p.3.b.total -117 -10078 21  
## p.3.c.total 21 1738 4  
## p.3.d.total 9003 25825 35427  
## p.3.a.ss 6511531 88216028 137027  
## p.3.b.ss 3605 22318856 6340  
## p.3.c.ss 160 694399 850  
## p.3.d.ss 63058 134394391 743785  
## p.3.a.min 42 10 -78  
## p.3.b.min -33 -2725 -7  
## p.3.c.min -1 0 -2  
## p.3.d.min 1 0 5  
## p.3.a.max 79 10 75  
## p.3.b.max 2 0 8  
## p.3.c.max 6 495 3  
## p.3.d.max 69 5904 38  
## p.3.n.count 1800 1800 1800  
## p.3.start.value 65 10 39  
## p.3.end.value 63 10 2

You may notice that the output shown above is slightly different than your own. I rounded the values and removed the scientific notation to hopefully make the results more readable. If you look at the rownames you can see that they are named systematically and in the following form:

**p.[phase].[coefficient].[intermediate value]**

* **[phase]** Designates the phase of the flight
  + 1 - Take off
  + 2 - Flight
  + 3 - Landing
* **[coefficient]** Designates the average of the coefficients from the quadratic line fits in all of the defined windows for the indicated phase
  + a - intercept
  + b - slope
  + c - acceleration
  + d - error term
* **[intermediate value]** Defines the specific Intermediate Value for the indicated coefficient and phase
  + total - sum of the RMSE's (root mean squared error) of the regression fits for the given phase
  + ss - sum of squares of the coeffcient for the given phase
  + min - minimum value if actual.min.max is TRUE, otherwise minimum coefficient
  + max - maximum value if actual.min.max is TRUE, otherwise maximum coefficient
* **Additionally...**
  + n.count rows denote the duration of the phase in seconds
  + start.value rows denote the first data point in the phase
  + end.value rows denote the last data point in the phase

### discreteIVT() Example

The discreteIVT() function is intended to calculate the percentage of time in each state of a discrete variable as well as the number and nature of transitions between states. This information is summarized by phase and can handle multiple discrete variables. Let's take a look at an example.

We will continue with the flight theme and create a dataset containing a set of discrete variables collected each second for the duration of the flight.

#Flight Variables  
  
# vector containing discrete state of forward thrust (On=1 Off=0)  
fwdThrust <- c(rep(0,300),rep(1,6300),rep(0,60),rep(1,1740))   
  
# vector containing discrete state of forward thrust (On=1 Off=0)  
rvseThrust <- c(rep(0,6600),rep(1,60),rep(0,1740))  
  
# vector containing discrete state of landing gear (Down=1 Up=0)  
lgearDown <- c(rep(1,720),rep(0,5760),rep(1,1920))  
  
# vector containing discrete state of direction (N=1 E=2 S=3 W=4)  
direction <- c(rep(1,2000),rep(2,5000),rep(3,1400))  
  
# integer vector identifying phase at each second of the flight  
phase.vec <- c(rep(1,600),rep(2,6000),rep(3,1800))

Now we can define some of the function parameters that we will need in order to use the discreteIVT() function.

#Function Parameters  
  
# vector with names of flight variables  
dNames <- c("fwdThrust","rvseThrust","lgearDown","direction")   
  
# vector identifying unique phases of the flight  
phases.to.process <- c(1,2,3)   
  
# Define number of seconds in the flight  
seconds <- 8400

Remember that you can see a list of arguments required for the discreteIVT() function using:

?discreteIVT

Now that we have our data we can use the discreteIVT() function to get a summary of the discrete states and transitions over each phase of the data.

dResult <- discreteIVT(dNames, phase.vec, phases.to.process, seconds)  
dResult

## 1 2 3  
## fwdThrust.percent.00\_00 0.5 0.00 0.03  
## fwdThrust.percent.01\_01 0.5 1.00 0.97  
## fwdThrust.transition.00\_01 1.0 0.00 1.00  
## fwdThrust.transition.01\_00 0.0 0.00 1.00  
## rvseThrust.percent.00\_00 1.0 1.00 0.97  
## rvseThrust.percent.01\_01 0.0 0.00 0.03  
## rvseThrust.transition.00\_01 0.0 0.00 1.00  
## rvseThrust.transition.01\_00 0.0 0.00 1.00  
## lgearDown.percent.00\_00 0.0 0.96 0.00  
## lgearDown.percent.01\_01 1.0 0.04 1.00  
## lgearDown.transition.00\_01 0.0 1.00 0.00  
## lgearDown.transition.01\_00 0.0 1.00 0.00  
## direction.percent.01\_01 1.0 0.23 0.00  
## direction.percent.02\_02 0.0 0.77 0.22  
## direction.percent.03\_03 0.0 0.00 0.78  
## direction.transition.01\_02 0.0 1.00 0.00  
## direction.transition.01\_03 0.0 0.00 0.00  
## direction.transition.02\_01 0.0 0.00 0.00  
## direction.transition.02\_03 0.0 0.00 1.00  
## direction.transition.03\_01 0.0 0.00 0.00  
## direction.transition.03\_02 0.0 0.00 0.00

The output shown will be slightly different than yours because I rounded the values and transposed the matrix in order to facilitate better readability for the sake of this guide. As you can see there is a clear structure in the row names (column names in your case) which is described below:

**[variable].[frequency].[from].[to]**

* **[variable]** Designates the discrete variable of interest from the data
  + fwdThrust
  + rvseThrust
  + lgearDown
  + direction
* **[frequency]** Either a duration or count summarizing discrete states
  + percent - displays the percent of time during a phase the discrete state did not change
  + transition - records a count of each type of unique discrete state transitions as specified by the from and to components for each phase.
* **[from]** Key indicating the discrete state of the variable before
  + XX - Number ranging over all possible discrete states of the variable of interest defined
* **[to]** Key indicating the discrete state of the variable after
  + XX - Number ranging over all possible discrete states of the variable of interest defined