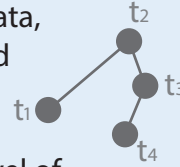


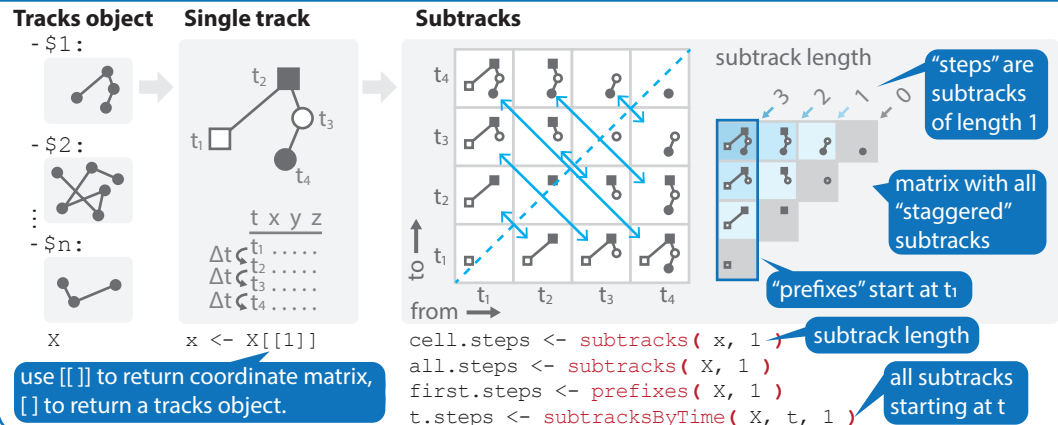
Analysing cell migration data in R

CelltrackR cheat sheet

To analyze cell movement, we record a cell's coordinates in time-lapse videos to obtain a cell *track*. To facilitate the interpretation of tracking data, **celltrackR** implements a large variety of methods for the fast and flexible analysis of track data in R. Load data from a text file, get rid of artefacts and tracking errors by performing quality controls proposed in literature, and analyze any metric on the level of individual tracks, steps, or subtracks. **CelltrackR** supports angle analyses and allows rapid visualization, clustering, and simulation of tracks. Let's get started!



3. Subsetting data: single tracks, (staggered) subtracks, and steps



1. Loading & converting tracks

Generate tracks object from a csv file:

```
mydata.csv: - $cell1: t x y z  
ID t x y z  
cell1 t1 . . .  
cell1 t2 . . .  
cell2 t1 . . .  
cell2 t2 . . .
```

tracks object contains a matrix for each cell

```
read.tracks.csv( mydata.csv,  
  id.column = 1, time.column = 2,  
  pos.columns = 3:5 )
```

Concatenate two tracks objects:

```
c( X1, X2 )
```

Convert between data structures:

```
dataframe as.data.frame( X )  
ID t x y z tracks to dataframe  
cell1 t1 . . . as.tracks( D )  
cell1 t2 . . . dataframe to tracks  
cell2 t1 . . . as.list( X )  
cell2 t2 . . . tracks to regular R list  
- $cell1:  
  t x y z  
  t1 . . .  
  t2 . . .  
wrapSingleTrack( x )  
wrap single track matrix  
into a track object
```

Sort tracks by time-order:

```
t . . . t . . .  
t8 . . . t1 . . .  
t3 . . . t2 . . .  
sort( X )  
Output of read.tracks.csv() and as.tracks.data.frame() is time-ordered by default.
```

2. Quality control & preprocessing

Longer tracks allow better inference of the cell's behavior, especially in **cell-based analyses** (box 4).

```
# tracks  
# steps  
hist( apply( X, nrow ) )  
length distribution  
maxTrackLength( X )  
longest track (# steps) must return TRUE/FALSE
```

```
filterTracks( function(x) nrow(x)>n, X )
```

keep only tracks of at least n steps
Filtering can cause bias. Consider a step-based analysis (box 4) instead of removing short tracks.

Check for **unequal Δt** between steps, or gaps:

```
Δt-avg(Δt)  
position  
split into two tracks  
interpolate @fixed Δt
```

```
avdt <- timeStep( x ); hist( apply( subtracks( x, 1 ), duration ) - avdt )
```

Fix this issue automatically for all tracks in X with an irregular Δt above some threshold: or: "split"
fix1 <- repairGaps(X, "interpolate")

Adjust **time resolution Δt**:
subsample every k-th timepoint
t . . . t . . .
t1 . . . t1 . . .
t2 . . . t3 . . .
t3 . . .
subsample(x, k = 2)
interpolateTrack(x, dtvec)
interpolate at times in dtvec

Angle analyses (box 6) can help detect artifacts, drift, and tracking errors (Beltman et al, 2009).

4. Analysis types: cell-based, step-based, and staggered metrics

Track properties can be computed in a cell-based, step-based, or staggered fashion. For more information, please refer to (Beltman et al, 2009). Examples are shown for the analysis of speed, but can also be performed with other analysis measures (box 5).

Cell-based

Find average speed of each individual cell (track):

```
# cells  
mean cell speed  
mean( apply( X, speed ) )  
cells have equal weights; steps from short tracks weigh more
```

Get instantaneous/"step" speed distribution for each cell (track):

```
# steps  
# steps  
step speed  
steps <- subtracks( x, 1 )  
hist( apply( steps, speed ) )  
steps of one cell x
```

Step-based

Average speed over all steps, pooled from all tracks together:

```
- $cell1:  
- $cell2:  
- ...  
aggregate( X, speed,  
  subtrack.length = 1,  
  FUN = mean )$value  
steps have equal weights; cells with longer tracks weigh more
```

To get the distribution over all steps instead of only the mean:

```
# steps  
step speed  
steps <- subtracks( X, 1 )  
hist( apply( steps, speed ) )  
all steps in object X
```

Staggered

Measure speed on all subtracks in the staggered matrix:

```
image( applyStaggered( x, speed, matrix = TRUE ) )  
symmetrical matrix. 0-step subtracks have no speed (NA).  
speed →  
if FALSE: return only the matrix mean, which is dominated by short (more frequent!) subtracks.
```

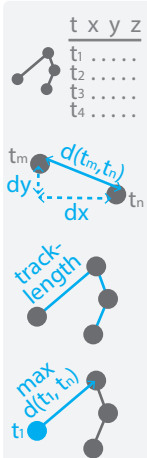
Directly get all mean cell speeds (over the staggered subtracks):

```
-  
-  
- ...  
supply( X,  
  staggered( speed ) )
```

5. Analysis measures

(see also ?TrackMeasures)

Speed and displacement



`duration(x)`
= $t_{end} - t_1$

`displacement(x, from = m, to = n)`
= $d(t_m, t_n)$

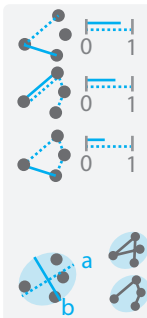
see also:
`squareDisplacement()`
`displacementVector()`
`normalizeToDuration()`

`trackLength(x)`
= $d(t_1, t_2) + \dots + d(t_{end-1}, t_{end})$

`speed(x)`
= $trackLength / duration$

`maxDisplacement(x)`
= $\max d(t_i, t_n)$

Track straightness



`displacementRatio(x)`
= $d(t_1, t_{end}) / \max d$

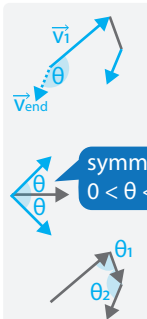
`outreachRatio(x)`
= $\max d / trackLength$

`straightness(x)`
= $d(t_1, t_{end}) / trackLength$

note that asphericity ignores time-ordering

`asphericity(x)`
= $(a^2 - b^2) / (a^2 + b^2)^2$

Turning angles



`overallAngle(x)`
= angle $\theta(\vec{v}_1, \vec{v}_{end})$ (first & last step)

`overallDot(x)`
= dot product $\vec{v}_1 \cdot \vec{v}_{end} = \frac{\cos \theta}{|\vec{v}_1| |\vec{v}_{end}|}$

symmetric
 $0 < \theta < \pi$

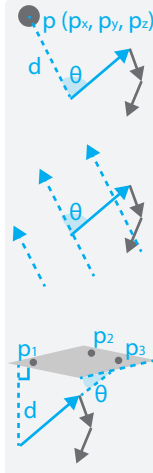
useful for autocorrelation/
autocovariance plots

`meanTurningAngle(x)`
= $\text{mean}(\theta_1, \dots, \theta_{end})$

6. Angles & Directionality

(see also ?AngleAnalysis)

Angles to a reference point, direction, or plane



`angleToPoint(x, p)`
= angle θ between first step
and reference point

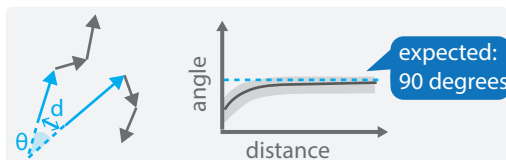
`distanceToPoint(x, p)`
= distance d between first step
and reference point

`angleToDir(x, dvec)`
= angle θ between first step
and reference direction

`angleToPlane(x, p1, p2, p3)`
= angle θ between first step
and plane with points p1-p3

`distanceToPlane(x, p1, p2, p3)`
= distance d between first step
and plane with points p1-p3

Angles between pairs of steps or tracks can help identify directional biases or artefacts (Beltman et al, 2009):



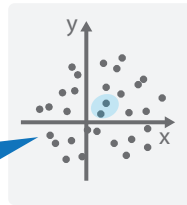
or try `analyzeCellPairs()`

`step.pairs <- analyzeStepPairs(X)`
`plot(step.pairs$dist, step.pairs$angle)`

Hotelling's test can help detect global directionality in a dataset in an unbiased fashion (Textor et al, 2011):

`hotellingsTest(X, plot = TRUE)`

does the average
step displacement
differ from the null
vector?



7. Visualization & Clustering:

detecting patterns in track data

Visualizing tracks in space

3D tracks? see `plot3d()`
& `projectDimensions()`

`plot(X)`
`plot(normalizeTracks(X))`

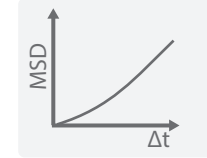
overlay track starting points

boundingBox(X)



Track measures by subtrack Δt : mean square displacement (MSD) & autocovariance plots

`plot(aggregate(X, squareDisplacement))`
`plot(aggregate(X, overallDot))`



Plot `cos()` of
overallAngle()
for an autocorrelation plot

Tracks in feature space:

Visualize two measures in a scatterplot:

`plotTrackMeasures(X, speed, meanTurningAngle)`

Or subset tracks by one feature first:

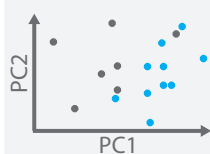
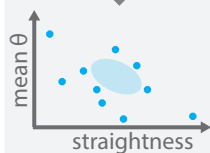
`minv <- median(apply(X, speed))`
`fast <- selectTracks(X, speed, minv, Inf)`

Or visualize higher dimensional feature sets with dimensionality reduction:

`trackFeatureMap(X, c(speed, straightness, meanTurningAngle), method = "PCA")`

Cluster tracks by features:

`clusterTracks(X, c(speed, straightness, meanTurningAngle), method = "hclust")`



Other methods:
"UMAP" / "MDS"



Or: "kmeans"

8. Simulating tracks:

Models & bootstrapping

Comparing observed data to idealized models is useful for interpretation. CelltrackR supports several methods for simulating tracks.

A random walk in dim dimensions:

`brownianTrack(nsteps, dim, mean=c(0,0), sd=c(1,1))` non-zero for directional bias

A "stop-and-go" model designed for T cells (Beauchemin et al, 2007). Cells move at speed v_{free} for time t_{free} , and then pause for a time t_{pause} before changing direction (can be with directional persistence or directional bias):

`beaucheminTrack(sim.time, delta.t, p.persist, p.bias, bias.dir, taxis.mode, t.free, v.free, t.pause)`

unlike `brownianTrack()`, `beaucheminTrack()` has an explicit definition of time.

A bootstrapped track matches speeds and turning angles to those observed in data:

`bootstrapTrack(nsteps, X)`

Simulate multiple tracks at once:

`simdata <- simulateTracks(10, bootstrapTrack(nsteps, X))`

or another simulation method

References

- Beauchemin et al (2007). Characterizing T cell movement within lymph nodes in the absence of antigen. *Journal of Immunology*.
- Beltman et al (2009). Analysing Immune cell migration. *Nature Reviews Immunology*.
- Mokhtari et al (2013). Automated characterization and parameter-free classification of cell tracks based on local migration behavior. *PLoS ONE*.
- Textor et al (2007). Defining the quantitative limits of intravital two-photon lymphocyte tracking. *PNAS*.



Learn more?

Check out the detailed examples in the package vignettes:

`browseVignettes(package = "celltrackR")`