Package 'textureAnalyzerGels'

March 10, 2021

Type Package

Title Reading and analyzing texture analyzer data

Version 1.0

Date 2013-10-18

Author Patrick Burch, Marina Braschler, Thomas Braschler

Maintainer Thomas Braschler <thomas.braschler@gmail.com>

Depends gsl, gdata, xlsx

Description Reads tabulated force-distance files made by the Exponent Software for the Texture Analyzer machines (export to text format), and Mecmesin (export from VectorPro).

License GPL (¿= 3)

LazyLoad yes

R topics documented:

`textureAnalyzerGels-package`
textureAnalyzerGels

Description

Reading and analyzing files written by the Exponent software of the Texture Analyzer. The Texture Analyzer compresses materials, and records Force - Distance curves. This package reads these force distance curves from tabulated text files exported from the Exponent software, and also contains routines for basic analysis of the data, namely in terms of the Young modulus.

Details

A typical workflow for reading mechanical compression data into R is:
1. Carry out mechanical measurement. Typically, a measurement is a single uniaxial compression/release cycle, with a maximum compression to some 20 percent of the original height to avoid mechanical damage. To reduce the influence of capillary pressure and fluid retention, the measurement is carried out with the sample immersed. The piston displacement speed should be as low as possible to reduce the influence of viscous drag of the liquid in the pore space. We use a TextureAnalyzer machine from Stable Microsystems for this purpose
2. Export data in tab format. Keep the maximum number of decimal places and time points.
3. Read the data into R. For reading a single file, read_texture_analyzer_tab can be used, but more generally, one will tabulate file information and metadata in a spreadsheet, and use the spreadsheet in R (see read.xls to direct the lecture of a multitude of .tab file with read_texture_analyzer_tab_list.
4. For soft gels in particular, the mechanical noise can be relatively high. It is therefore

advisable to carry out low-pass filtering to reduce noise. This can be done explicitly with smooth_texture_analyzer_data. If read_texture_analyzer_tab_list is used, this is more typically done directly by read_texture_analyzer_tab_list.
In this workflow, the smoothing steps are especially slow, and for a larger collection of files, it is clearly advisable to write the smoothened curves into R datafiles. For this, the save command can be used.

Analysis depends on the type of gel from which one wants to extract information.

For cryogels of sufficient interconnectivity and porosity, foam mechanical analysis may be appropriate. Under certain circumstances (among them, sufficient porosity and interconnectivity), these gels have a mechanical behaviour akin to classical foam-like materials in compression (stress-strain diagram): there is an initial elastic compression segment, followed by a wall-buckling plateau, followed by steeply increasing stress in densification (Gibson 1997). For such gels, mechanical analysis in terms of foam-mechanics is most suitable. In this case, the workflow can be continued as follows:
1. foam_mechanical_analysis respectively foam_mechanical_analysis_list for an set of measurements can be used to extract characteristic points and data from the compression / relaxation curves
2. write_foam_mechanical_analysis_list_to_xlsx can be used to write the extracted data to an Excel file for external analysis.
3. process_foam_mechanical_analysis_list_to_dataframe processes the nested list output given by foam_mechanical_analysis_list into a more standard R dataframe from which the desired values can be aggregated and plotted in more standard R manner.

For classical hydrogels without or with limited macroporosity, no plateau exists, and other analysis is appropriate (for example, simple linear fitting (lm), or analysis in more sophisticated frameworks such as Ogden model (Ogden 1972, ogden_model).

Author(s)

Patrick Burch
Marina Braschler
Thomas Braschler

Maintainer: Thomas Braschler ¡thomas.braschler@gmail.com¿

References

Gibson, L. J. & Ashby, M. F. Cellular Solids: Structure and properties. (Cambridge University Press, 1997).

Ogden, R. W. (1972), Large Deformation Isotropic Elasticity - On the Correlation of Theory and Experiment for Incompressible Rubberlike Solids,Proceedings of theRoyal Society A: Mathematical, Physical and Engineering Sciences, 326(1567), 565-584.

Beduer A, Braschler T, Peric O, Fantner GE, Mosser S, Fraering PC , Bencherif SA, Mooney DJ, Renaud P: A compressible scaffold for minimally invasive delivery of large intact neuronal networks, Adv Healthc Mater, 2015, 4(2):301-12, https://doi.org/10.1002/adhm.201400250

Examples

```
## Not run:
path = system.file('sampleData/2016_08_30_Patrick_sample', package = 'textureAnalyzerGels');
CMC_concentration_file_info = read.xls(file.path(path,"file_listing.xlsx") )
CMC_concentration_data=read_texture_analyzer_tab_list(file_info=CMC_concentration_file_info,root_folder=pat
analysis_CMC_concentration_list=foam_mechanical_analysis_list(CMC_concentration_file_info,CMC_concentration
analysis_CMC_concentration_dataframe=analysis_list_to_dataframe(CMC_concentration_file_info,analysis_CMC_co


# Plotting the initial Young moduli on the elastic segment before the plateau. The Young moduli depend strongly o
# the initial Young moduli are higher than linear fits that include plateau effects (c.f. Beduer 2015)
plot(value ~ CMC_conc, analysis_CMC_concentration_dataframe[analysis_CMC_concentration_dataframe$direction==

## End(Not run)
```

`analysis_list_to_dataframe`

analysis_list_to_dataframe

Description

Converts the output of `foam_mechanical_analysis_list`, which is a nested list (list of list), into a flat dataframe for use in plotting and statistical analysis.

Usage

```
analysis_list_to_dataframe(file_info,output_foam_mechanical_analysis_list)
```

Arguments

`file_info`	Description of the compression curves. This is a dataframe, which needs to contain as many lines as there are elements in `data_list`
`output_foam_mechanical_analysis_list`	Output by foam_mechanical_analysis_list. This is a list of lists

Value

A dataframe containing one line per estimated value. The values are described by the columns `measurement`, `direction` and `point`. The `measurement` indicates what is measured (stress, strain, distance, and so forth), the `direction` indicates whether this was obtained in the "down" or "up" cycle (compression or release), while the `point` column indicates the relevant segment on the foam compression curve (touchpoint, linear elastic segment, transition to plateau, plateau, transition to densification). As there are many measurements and points, as well as the "down" and "up" cycle, there are many lines in the return value for each line in `file_info`. The

Author(s)

Thomas Braschler

See Also

read_texture_analyzer_tab, used internally

Examples

```
data(CMC_concentration_data)
foam_mechanical_analysis_list_result=foam_mechanical_analysis_list(file_info=CMC_concentration_file_info, c


analyzed_data = analysis_list_to_dataframe(file_info=CMC_concentration_file_info,output_foam_mechanical_ana



# Usage of the dataframe: one has to select specific measurements at specific points and a specific direction

E_elastic_down = analyzed_data[analyzed_data$measurement=="E" & analyzed_data$point=="elastic" & analyzed_dat

plot(value ~ CMC_conc, E_elastic_down[order(E_elastic_down$CMC_conc),],type="b",log="xy")
```

CMC_concentration_data *Sample set of cryogel compression data, with different concentration of carboxymethylcellulose*

Description

Sample compression curves for CMC/AAD cryogels ranging from 0.65 percent to 2.6 percent of CMC, with always a nominal crosslinking degree of every tenth residue crosslinked by AAD.

Usage

```
data(CMC_concentration_data)
```

Format

Contains three variables:

`CMC_concentration_data` List of 6 elements, containing the compression curves as read by read_texture_analyzer_tab and smoothened by smooth_texture_analyzer_data

`CMC_concentration_file_info` Dataframe with 6 lines, desribing the conditions associated with the 6 compression curves. Here, it is primarily the CMC concentration that varies

`CMC_concentration_blank` Dataframe as produced by by read_texture_analyzer_tab and smoothened by smooth_texture_analyzer_data. This the blank without gel that should be used to subtract the Archimedes force (this is particularly important for the softer gels)

Author(s)

Patrick Burch, Thomas Braschler

Examples

```
# Generation of the data
path = system.file('sampleData/2016_08_30_Patrick_sample', package = 'textureAnalyzerGels');
CMC_concentration_file_info = read.xls(file.path(path,"file_listing.xlsx") )
CMC_concentration_data=read_texture_analyzer_tab_list(file_info=CMC_concentration_file_info,root_folder=pat

# The blank, for buyuoancy with gel, is common to all samples
CMC_concentration_blank=read_texture_analyzer_tab_list(file_info=CMC_concentration_file_info[1,],folder_col


for(ind in 1:length(CMC_concentration_data))
{
theFile = CMC_concentration_file_info[ind,]
theData=CMC_concentration_data[[ind]]

CMC_concentration_data[[ind]]$pressure_reduced = theData$pressure - CMC_concentration_blank$pressure[match(p


CMC_concentration_data[[ind]]$pressureSlope_reduced = theData$pressureSlope - CMC_concentration_blank$pressu

CMC_concentration_data[[ind]]$height = theFile$start_height_mm-theData$Distance



}

# ============================================================
# Loading of the data and elementary plotting
data(CMC_concentration_data)
plot(CMC_concentration_data[[1]]$Distance, CMC_concentration_data[[1]]$pressure/1e3,type="l",xlab="Distance

for(ind in 2:length(CMC_concentration_data))
{
lines(CMC_concentration_data[[ind]]$Distance, CMC_concentration_data[[ind]]$pressure/1e3,type="l",col=palet
}

lines(CMC_concentration_blank$Distance,CMC_concentration_blank$pressure/1e3,type="l",col=palette()[length(C

legend("topleft",legend=c(paste("c(CMC) = ",round(CMC_concentration_file_info$CMC_conc*100)/10, " mg/mL",sep
```

---

detect_compression_cycles
detect_compression_cycles

Description

Detects cycle of back-and-forth movement in texture Analyzer data

Usage

```
detect_compression_cycles(textureAnalyzerData,do_aggregation=TRUE)
```

Arguments

`textureAnalyzerData`	Data as read by read_texture_analyzer_tab
`do_aggregation`	Whether or not to do aggregation. If TRUE, aggregate by the `Distance` column within each phase ("up" or "down") in each of the cycles. Is set to TRUE by default

Details

The routine looks for up and down movements by analyzing the "Distance" column in the `textureAnalyzerData` data frame.

Value

The same dataframe that was passed to the function, with two supplementary (or overwritten) columns: "direction" indicates the current movement, "cycle" indicates the current compression cycle.

Author(s)

Thomas Braschler

Examples

```
# This is destructive testing on a classical (nanoporous) CMC gel, for illustration of the multi-cycle analysis
path = system.file('sampleData', package = 'textureAnalyzerGels');
sampleGel=detect_compression_cycles(read_texture_analyzer_tab(paste(path,"sampleDataMultipleCycles.tab",sep

sampleGelSmooth = smooth_texture_analyzer_data(sampleGel,boundary_extension_mm=0.2,sd=0.0025)

cycle = 1

col=palette()[cycle]

plot(pressure ~ Distance, sampleGelSmooth[sampleGelSmooth$cycle==cycle ,],type="l",col=col,ylim=c(-750,1250)


for(cycle in 2:max(sampleGelSmooth$cycle))
{
col=palette()[cycle]

lines(pressure ~ Distance, sampleGelSmooth[sampleGelSmooth$cycle==cycle ,],type="l",col=col,ylim=c(-500,1000


}
```

---

diffusion_from_disk *diffusion_from_disk*

Description

Remaing total fraction of solute in a originally homogeneously loaded disk, with the boundary condition of c=0 on the portour

Usage

```
diffusion_from_disk(r,D,t)
```

Arguments

r	Radius of the disk
D	Diffusion coefficient
t	Time since the beginning of the diffusion

Details

The general solution of the diffusion equation in cylindrical coordinates can be expressed in terms of Bessel functions. It can for instance be found in Keil 1971. For a disk with initially homogenous concentrations, the coefficients take a slightly particular form (e.g. https://mathworld.wolfram.com/HeatConductionEquationDisk.html) such that the integral giving the remaining solute fraction becomes:

```
B(tau)=4*SUM_n 1/jon^2 * exp(-jon^2*tau)
```

where `tau=D*t/r^2` and where the `jon` terms indicate the n-th zero of the Bessel function of the first kind, order 0 (e.g. `J0`). The 4 arises due to the particular value of the infinite series for `tau=0`, where the exponential terms all evaluate to 1 (c.f. deLyra 2013).

The time-dependency of the remaining solute in a disk was used to evaluate hydrogel pore size from water loss under compressive load Braschler et al. (2015).

Value

Fraction of solute still remaining in the disk

Author(s)

Thomas Braschler

References

E. Keil, 13.12.1971, Solution of the diffusion equation in cylindrical coordinates and comparison with experiments, https://cds.cern.ch/record/1129859/files/CM-P00065889.pdf

Wolfram web ressources: https://mathworld.wolfram.com/HeatConductionEquationDisk.html

Jorge L. deLyra, On the Sums of Inverse Even Powers ofZeros of Regular Bessel Functions, https://arxiv.org/abs/1305.0228 Braschler T, Songmei W, Wildhaber F, Bencherif SA, Mooney DJ, "Soft nanofluidics governing minority ion exclusion in charged hydrogels", Soft Matter, Issue 20, 2015, Supplementary 2.

See Also

The function makes use of the bessel_zero_J0 function

Examples

```
t=seq(from=0,to=1,by=0.01)
remaining=diffusion_from_disk(1,1,t)
plot(remaining~t)
```

`find_densification_limit` *find_densification_limit*

Description

For porous (foam-like) materials, estimates the end of the plateau segment where the compression force starts to raise dramatically. In foams, this marks the end of pore space compression in favour of increasing wall material loading (Gibson&Ashby, 1997).

Usage

```
find_densification_limit(theData,touch_point,elastic_limit,gel_thickness=NULL)
```

Arguments

`theData`	Smoothened texture analyzer data, as produced by the smooth_texture_analyzer_data function; should contain only a one-way movement (down or up) for the chuck
`touch_point`	Point where the chuck touches the gel. This can be found from the Force - Distance diagram. where the force curve in a Force - Distance diagram raises abruptly
`elastic_limit`	Output of find_elastic_limit, which should be run before this function.
`gel_thickness`	If an estimation of the overall gel thickness is known, this is used to generate a better initial guess of the densification limit.

Details

The function uses both the slope and the actual stress (force per area, pressure) to find the initial touch point, hence the `theData` argument must have at least the columns Distance, pressure and pressureSlope. The estimation is based on the idea to find the point where the measured pressure starts to rise substantially above the expected plateau pressure. Quantitatively, this is judged by comparing measured pressure to extrapolated plateau pressure (linear extrapolation based on minimal slope), with the requirement that the excess pressure exceeds the actual plateau pressure value.

Value

Numerical value, indicating the position of the densification limit

Author(s)

Thomas Braschler

References

Gibson, L. J. & Ashby, M. F. Cellular Solids: Structure and properties. (Cambridge University Press, 1997).

See Also

smooth_texture_analyzer_data

Examples

```
data(sampleGelSmooth)
touch_point=find_initial_touch_point(sampleGelSmooth[sampleGelSmooth$direction=="down",])
elastic_limit = find_elastic_limit(sampleGelSmooth,touch_point)
densification_limit=find_densification_limit(sampleGelSmooth[sampleGelSmooth$direction=="down",],touch_poi
plot(Force ~ Distance, sampleGelSmooth[sampleGelSmooth$direction=="down",])
lines(rep(touch_point,2),c(0,11),col="black")
lines(rep(elastic_limit,2),c(0,11),col="black")
lines(rep(densification_limit,2),c(0,11),col="red")
text(touch_point,11.5,labels="Touch point",adj=0.5)
text(elastic_limit,11.5,labels="Elastic limit",adj=0.5)
text(densification_limit,11.5,labels="Densification limit",adj=0.5,col="red")
```

---

find_elastic_limit *find_elastic_limit*

Description

Estimates the end of the linear elastic segment where the compression force starts to plateau. In prototypical foams, this marks the transition from largely undeformed linear compression to wall buckling (Gibson&Ashby, 1997).

Usage

```
find_elastic_limit(theData,touch_point,gel_thickness=NULL)
```

Arguments

theData	Smoothened texture analyzer data, as produced by the smooth_texture_analyzer_data function; should contain only a one-way movement (down or up) for the chuck
touch_point	Point where the chuck touches the gel. This can be found from the Force - Distance diagram. where the force curve in a Force - Distance diagram raises abruptly
gel_thickness	Optionally, thickness of the gel to help make an educated guess of the extent of the elastic segment (20% of the gel height).

Details

The function uses both the slope and the actual force to find the initial touch point, hence the theData argument must have at least the columns Distance, pressure and pressureSlope

Value

Numerical value, indicating the position of the surface of the gel

Author(s)

Thomas Braschler

References

Gibson, L. J. & Ashby, M. F. Cellular Solids: Structure and properties. (Cambridge University Press, 1997).

See Also

smooth_texture_analyzer_data

Examples

```
data(sampleGelSmooth)
touch_point=find_initial_touch_point(sampleGelSmooth[sampleGelSmooth$direction=="down",])
elastic_limit = find_elastic_limit(sampleGelSmooth,touch_point)
plot(Force ~ Distance, sampleGelSmooth[sampleGelSmooth$direction=="down",])
lines(rep(touch_point,2),c(0,11),col="black")
lines(rep(elastic_limit,2),c(0,11),col="red")
text(touch_point,11.5,labels="Touch point",adj=0.5)
text(elastic_limit,11.5,labels="Elastic limit",adj=0.5,col="red")
```

---

find_initial_touch_point

find_initial_touch_point

Description

Estimates the point where the chuck touches the gel for the first time

Usage

```
find_initial_touch_point(theData, approximate_touch_point = NULL,free_region=NULL)
```

Arguments

theData	Smoothened texture analyzer data, as produced by the smooth_texture_analyzer_data function; should contain only a one-way movement (down or up) for the chuck
approximate_touch_point	In initial guess of the touch point. If necessary, this can be obtained visually by finding the point where the force curve in a Force - Distance diagram raises abruptly
free_region	Below this distance, the gel does not touch the compression piston, such that only buoyancy effects are observed

Details

The function uses both the slope and the actual force to find the initial touch point, hence the `theData` argument must have at least the columns Distance, Force and ForceSlope

Value

Numerical value, indicating the position of the surface of the gel

Author(s)

Thomas Braschler

See Also

smooth_texture_analyzer_data

Examples

```
data(sampleGelSmooth)
touch_point=find_initial_touch_point(sampleGelSmooth[sampleGelSmooth$direction=="down",])
plot(Force ~ Distance, sampleGelSmooth[sampleGelSmooth$direction=="down",])
lines(rep(touch_point,2),c(0,11),col="red")
text(touch_point,11.5,labels="Touch point",adj=0.5,col="red")
```

`find_landmarks_gel_compression`

find_landmarks_gel_compression

Description

Finds landmark points in as porous gel compression curve (Force - Distance curve). Currently, detection of the gel touch point in both the "down" and "up" cycle of the chuck is implemented. The analysis is based on the general curve for foams, comprising an initial steep elastic segment, followed by a plateau and then a densification region (Gibson &Ashby, 1997).

Usage

```
find_landmarks_gel_compression(theData, approximate_gel_touch_point = NULL,gel_thickness=NULL)
```

Arguments

`theData`	Force-Distance data, typically from a "down"-"up" cycle of the chuck; the data needs to be smooth and contain already the ForceSlope column, which can be obtained by smooth_texture_analyzer_data
`approximate_gel_touch_point`	Initial guess of the gel touch point
`gel_thickness`	Macroscopic gel thickness estimation, can be used to improve initial guess for the densification limit (via find_densification_limit)

Details

The function makes use of find_initial_touch_point for each movement cycle

Value

Vector of estimated touch points, one for the down movement, one for the up movement

Author(s)

Thomas Braschler

References

Gibson, L. J. & Ashby, M. F. Cellular Solids: Structure and properties. (Cambridge University Press, 1997).

See Also

`smooth_texture_analyzer_data`

And the functions used internally: `find_initial_touch_point`
`find_elastic_limit`
`find_densification_limit`

Examples

```
data(sampleGelSmooth)
find_landmarks_gel_compression(sampleGelSmooth)
```

---

fit_young_modulus_polymer_from_bulk

fit_young_modulus_polymer_from_bulk

Description

Estimation of the intrinsic Young modulus of the polymer phase in a cryogel from the bulk Young modulus

Usage

```
fit_young_modulus_polymer_from_bulk(polymer_volume_fraction,E_bulk,return_model=FALSE)
```

Arguments

`polymer_volume_fraction`	Volume fraction occupied by the polymer phase
`E_bulk`	Apparent Young modulus of the bulk structure as measured by a mechanical test equipment with a chuch size much larger than the pore size
`return_model`	If TRUE, return the nls model used for fitting as an attribute to the result (e.g. attr(result,"model"))

Details

The model used for the estimation is explained in more detail in young_modulus_cryogel_theory; it is given by `E_bulk = E_polymer*polymer_volume_fraction^3`, as described in Beduer et al., 2015. The fitting is carried out by taking the log of both the `polymer_volume_fraction` and `E_bulk`, and the by linear fitting of a straight line of slope 3.

If both `polymer_volume_fraction` and `E_bulk` are numeric (length=1), the calculation is carried out exactly.

Value

Estimation of the microscopic Young modulus of the polymer phase

Author(s)

Thomas Braschler

References

Beduer A, Braschler T, Peric O, Fantner E, Mosser S, Fraering PC, Bencherif SA, Mooney DJ, Renaud P: A Compressible Scaffold for Minimally Invasive Delivery of Large Intact Neuronal Networks, Advanced Healthcare Materials, 2015, 4(2), 301-312

Examples

```
fit_young_modulus_polymer_from_bulk(0.01,1e3)

fit_young_modulus_polymer_from_bulk(c(0.01,0.02,0.03),c(0.2e3,0.7e3,2e3))
```

`foam_mechanical_analysis` *foam_mechanical_analysis*

Description

Analysis of foam compression curves. This function assumes a classical foam compression behaviour, with a steeper elastic, a flatter plateau and then again a steep densification region in stress-strain (or equivalently, force-distance) diagrams (Gibson&Ashby, 1997). This function separates the prototypical foam compression curve into different zones by indicating the transition points: [free zone (no contact between chuck and foam)] - touchpoint - elastic compression region - transition to plateau - plateau region - transition to densification- densification. Localization and mechanical properties are analyzed at the points from touchpoint to the transition to densification.

Usage

```
foam_mechanical_analysis(theData,start_height,approximate_gel_touch_point=NULL,landmarks=NULL,ge
```

Arguments

theData	Force-Distance data, typically from a "down"-"up" cycle of the chuck; the data needs to be smooth and contain already at the very least the columns Distance and direction, in addition to the colums described by the arguments stress_column and stress_slope_column. Such a variable is typically obtained by using read_texture_analyzer_tab and then smooth_texture_analyzer_data on the result. Care should be taken for the smoothing parameter sd in smooth_texture_analyzer_data, both the pressure and the pressureSlope should appear smooth in a diagram versus distance, while overall look of the curve with a flat part, a steeper elastic part, a less steep plateau and then again rising densitifcation in pressure vs. Distance should be conserved.
start_height	The texture Analyzer typically starts a file by Distance=0, so we loose the information on how high the chuck is above the substrate before starting compression. This needs to be noted separately and transmitted to this function in the start_height argument.
approximate_gel_touch_point	Initial guess for the gel touch point (will be refined by a call to find_landmarks_gel_compression
landmarks	Possibility to manually provide landmarks. Typically, there is no need to provide this as the function uses find_landmarks_gel_compression internally. If provided, numeric vector with elements named "touch_point", "elastic_limit", "densification_limit". If an incomplete set of named elements is given, the function will try to find the missing values via find_landmarks_gel_compression
gel_thickness	Possibility to manually provide gel thicknessm in the same units as is the "Distance" column in theData. More typically, there is no need to provide this argument as gel_thickness will be estimated from start_height and the distance the chuck needs to travel down until contact with the gel is established (the touchpoint distance).
stress_column	Name of the column in theData that contains the stress information. By default, this is the "pressure" column as produced by read_texture_analyzer_tab and smoothened by smooth_texture_analyzer_data. An alternative column can be used, for instance if chuck buoyancy is substracted with the creation of a new column (done manually).
stress_slope_column	Name of the column in theData that contains the slope of the stress. The units are the units of the stress column devided by the units of the Distance column. By default, this is the "pressureSlope" column as produced by read_texture_analyzer_tab and smoothened by smooth_texture_analyzer_data. An alternative column can be used, for instance if chuck buoyancy is substracted with the creation of a new column (done manually).
do_plot	If TRUE, plots the stress-Distance curve and indicates the landmarks found

Details

The algorithm sequentially evaluates the positions of the analysis points (touchpoint, mid elastic, transition to plateau, plateau, transition to densification), and also evaluates the different geometric and mechanical measures at these points. First, the function needs the landmarks available either manually or more typically through find_landmarks_gel_compression, called internally if the landmarks argument is not provided.

Within the elastic region delimited by the landmarks, the algorithm then looks for a local maximum of the slope (in the column designated by the argument `stress_slope_column` in `theData`. For this it is important that the curve is appropriately smoothened beforehand, in general by using smooth_texture_analyzer_data before using this function. Appropriately means here that noise should be removed so that the slope appears smooth, but the smoothing should still conserve the overall features of the curve and namely the existence of a local maximum in the elastic compression region.

The touchpoint is then sought using the data from the elastic midpoint. For this, the algorithm considers the tangent at the elastic midpoint in the Force vs. Distance diagram, and looks for its intersection with x-axis. This defines the touchpoint where the chuck is assumed to substantially enter into interaction with the gel.

On the other side of the midpoint of the elastic region defined by the local maximum slope, the algorithm then looks for a minimum slope at higher compression. For this it is again important to have the curve appropriately smoothened, otherwise the minimum will be defined by noise rather than the global shape of the curve. The plateau is a feature of foams with sufficiently high pore fraction, it may not exist in all experimental configurations. If no local minimum is found, the plateau midpoint is set identical to the elastic compression midpoint, and `plateau_detected=FALSE` will be returned for the corresponding direction ("down" or "up", see value section). In this case, the analysis is not appropriate and the caution should be applied in the interpretation of the values.

The transition point between elastic compression and elastic compression region and the plateau region is then sought as the intersection between the tangent at the plateau midpoint and the tangent at the elastic midpoint, in the Force vs. Distance diagram. The transition point is noted "transition_E_P" for transition from elastic to plateau.

Finally, the function determines the transition point from plateau to densification ("transition_P_D"). For this, it first determines the slope at the transition point from elastic compression to plateau, and then looks for the first point beyond the plateau midpoint where this slope is again exceeded. This first point is the transition point from plateau to densification.

The above algorithm is carried out on both the descending "down" part of the compression cycle and the ascending ("up") part of the compression cycle. Usually, the positions and values for the individual points differ slightly. For the determination of the gel height, the average of the touchpoint Distance is evaluated. This is then substracted from the starting height, which needs to be supplied (`start_height`).

With the positions of the different points known, stress, strain, and Young moduli are then evaluated for the points for both the "down" and "up" part of the compression cycle and tabulated as described in the value section.

Value

A list with different geometric and mechanical measures:

Distance Position of the analyzed points. The starting point of the compression is 0. This

is a matrix with 2 rows (for the "down" and "up" part of the compression cycle"), and 5 columns. There is a column for each characteristic point of the curve: "touchpoint" for the point where the chuck touches the gel; "elastic" for the midpoint of the elastic compression part; "transition_E_P" for the transition point between elastic compression and plateau; "plateau" for the plateau midpoint; "transition_P_D" transition point from the plateau to densification.

strain Relative compression. This is distance relative to the touchpoint, divided by gel height. This variable has the same structure as the Distance element described above.

strain Relative compression. This is distance relative to the touchpoint, divided by gel height. This variable has the same structure as the Distance element described above.

stress Mechanical stress (force per area) at the different points. Again same matrix layout.

E Local Young moduli. The local Young moduli are the local slopes (derivative) in strain-stress diagrams. Same structure. The Young modulus on the elastic segment ("elastic") in the compression direction ("down" if using compression and relaxation) can be used to define the Young modulus as it corresponds to the zone of linear elasticity (Gibson&Ashby 1997).

E_from_touchpoint Young moduli as measured from the touchpoint.The slope is taken from the straight line connecting the touchpoint to the desired second point on the curve in the strain-stress diagram. Same matrix structure as the variables before.

plateau_detected Whether or not the was a plateau detected. A plateau is detected if the smoothened slope has a minimum beyond the elastic compression, otherwise no plateau is detected. Logical vector of two elements, for the down and up part of the compression cycle.

plateau_width In terms of strain, the width of the plateau. Numerical vector of two values, for the up and down part of the compression cycle.

gel_height Height of the gel. Single numerical value, in mm

Author(s)

Thomas Braschler

References

Gibson, L. J. & Ashby, M. F. Cellular Solids: Structure and properties. (Cambridge University Press, 1997).

See Also

find_landmarks_gel_compression, used internally

Examples

```
data(sampleGelSmooth)
results=foam_mechanical_analysis(
sampleGelSmooth,start_height=start_height_mm,approximate_gel_touch_point=0.15)
results
Distance = results[["Distance"]]
stress = results[["stress"]]
plot(pressure ~ Distance, sampleGelSmooth,type="l",xlab="Distance [mm]", ylab="Stress [Pa]",ylim=c(0,2500))
lines(Distance["down",],stress["down",],type="p",col="red")
lines(Distance["up",],stress["up",],type="p",col="green")
text(x=Distance["up","touchpoint"]+0.22,y=stress["up","touchpoint"],labels="Touchpoint")
```

```
text(x=Distance["up","elastic"]+0.3,y=stress["up","elastic"],labels="Linear elasticity")
text(x=Distance["up","transition_E_P"]+0.2,y=stress["up","transition_E_P"]-30,labels="Transition")
text(x=Distance["up","plateau"]+0.2,y=stress["up","plateau"]-30,labels="Plateau")
text(x=Distance["up","transition_P_D"]+0.45,y=stress["up","transition_P_D"]-30,labels="Transition to densif

legend("topleft",pch=c(1,1),col=c("red","green"),legend=c("Compression","Relaxation"))
```

`foam_mechanical_analysis_list` *foam_mechanical_analysis_list*

Description

Analysis of list of foam compression curves. This function applies foam_mechanical_analysis to each compression curve and collects the results

Usage

```
foam_mechanical_analysis_list(file_info,data_list,start_height_column="start_height_mm",approxim
```

Arguments

`file_info`	Description of the compression curves. This is a dataframe, which needs to contain at least the columns with names given by the arguments `start_height_column` and `approximate_gel_touch_point_column`, but usually contains more information such as the fabrication conditions. `file_info` needs to contain as many lines as there are elements in `data_list`
`data_list`	List of compression curves, typically produced by read_texture_analyzer_tab_list
`start_height_column`	Name of the starting height column in `file_info`. This column describes the starting height of the chuck at the beginning of compression, in millimeters
`approximate_gel_touch_point_column`	Name of the column in `file_info` that contains a manual estimation of the approximate gel touchpoint, in mm.
`...`	Possibility to pass down further arguments to foam_mechanical_analysis, used internally

Value

A list with as many elements as there are lines `file_info`. Each of these elements is the output of foam_mechanical_analysis. As the output of foam_mechanical_analysis is a list itself, the return value is a list of lists.

Author(s)

Thomas Braschler

See Also

foam_mechanical_analysis, used internally

Examples

```
data(CMC_concentration_data)
results=foam_mechanical_analysis_list(file_info=CMC_concentration_file_info, data_list=CMC_concentration_da
```

`foam_mechanical_analysis_mecmesin`
foam_mechanical_analysis_mecmesin

Description

Specific adaption of foam_mechanical_analysis to Mecmesin file format.

Usage

```
foam_mechanical_analysis_mecmesin(theData,offset_height=0,approximate_gel_height=NULL,stress_col
```

Arguments

`theData`	Force-Distance data, typically from a "down"-"up" cycle of the chuck; the data needs to be smooth and contain already at the very least the columns `Distance` and `direction`, in addition to the colums described by the arguments `stress_column` and `stress_slope_column`. Such a variable is typically obtained by using read_mecmesin_tab.
`offset_height`	If zeroing of the z-position ("Distance" column in `theData`) is performed at some point above the bench surface, the height offset needs to be added to obtain the true height above the bench surface. Provide this information in the `offset_height` argument if necessary.
`approximate_gel_height`	Approximate (typically, caliper-based) estimation of the gel height in mm
`stress_column`	Name of the column in `theData` that contains the stress information. By default, this is the "pressure" column as produced by read_texture_analyzer_tab and smoothened by smooth_texture_analyzer_data. An alternative column can be used, for instance if chuck buoyancy is substracted with the creation of a new column (done manually).
`stress_slope_column`	Name of the column in `theData` that contains the slope of the stress. The units are the units of the stress column devided by the units of the Distance column. By default, this is the "pressureSlope" column as produced by read_texture_analyzer_tab and smoothened by smooth_texture_analyzer_data. An alternative column can be used, for instance if chuck buoyancy is substracted with the creation of a new column (done manually).

Details

The usage of the Mecmesin MultiTest 2.5dV apparatus we used is a bit different from the TextureAnalyzer TA.XT. At least with our present settings, the TextureAnalyzer TA.XT files indicate distances relative to the starting point, so that the "Distance" values rise from 0 to some maximum positive value before descending to zero again in the retreat phase of the test. On the contrary, the Mecmesin apparatus is calibrated to absolute height above the bench surface and so the "Distance" values in compression commence at some positive value, decrease, and increase again. This function takes the "Mecmesin" distances and recalculates 0-based distance-values, and then uses foam_mechanical_analysis for the actual mechanical analysis.

Another particularity to describe in further detail is the offset height. Especially with the more sensitive strain gauges, it is in fact a bit risky to perform zeroing on a hard surface (the gauges easily break at even slight overloads, while the test bench is capable to literally crush them), and so we typically zero on softer surface or by manually measuring the position of the chuck above the bench surface. As a result, the distance is correct only to within some positive offset value, which has to be specified as the apparatus has no way of knowing it.

This function internally calculates key arguments to feed to foam_mechanical_analysis. First, the `start_height` is thus given by the initial distance value plus the `offset_height`. Second, the initial estimate of the gel touch point (i.e. `approximate_gel_touch_point` in foam_mechanical_analysis) is calculated from a macroscopic (caliper) measurement of the gel height provided as argument `approximate_gel_height` here, as `approximate_gel_touch_point=start_height-ap`

Value

A list with different geometric and mechanical measures, as for foam_mechanical_analysis :

Distance Position of the analyzed points. The starting point of the compression is 0. This is a matrix with 2 rows (for the "down" and "up" part of the compression cycle"), and 5 columns. There is a column for each characteristic point of the curve: "touchpoint" for the point where the chuck touches the gel; "elastic" for the midpoint of the elastic compression part; "transition_E_P" for the transition point between elastic compression and plateau; "plateau" for the plateau midpoint; "transition_P_D" transition point from the plateau to densification.

strain Relative compression. This is distance relative to the touchpoint, divided by gel height. This variable has the same structure as the Distance element described above.

strain Relative compression. This is distance relative to the touchpoint, divided by gel height. This variable has the same structure as the Distance element described above.

stress Mechanical stress (force per area) at the different points. Again same matrix layout.

E Local Young moduli. The local Young moduli are the local slopes (derivative) in strain-stress diagrams. Same structure. The Young modulus on the elastic segment ("elastic") in the compression direction ("down" if using compression and relaxation) can be used to define the Young modulus as it corresponds to the zone of linear elasticity (Gibson&Ashby 1997).

E_from_touchpoint Young moduli as measured from the touchpoint.The slope is taken from the straight line connecting the touchpoint to the desired second point on the curve in the strain-stress diagram. Same matrix structure as the variables before.

plateau_detected Whether or not the was a plateau detected. A plateau is detected if the smoothened slope has a minimum beyond the elastic compression, otherwise no plateau is detected. Logical vector of two elements, for the down and up part of the compression cycle.

plateau_width In terms of strain, the width of the plateau. Numerical vector of two values, for the up and down part of the compression cycle.

gel_height Height of the gel. Single numerical value, in mm

Author(s)

Thomas Braschler

References

Gibson, L. J. & Ashby, M. F. Cellular Solids: Structure and properties. (Cambridge University Press, 1997).

See Also

foam_mechanical_analysis, used internally
read_mecmesin_tab that can be used for reading Mecmesin text files to import the relevant data into R.

Examples

```
# This example is to illustrate the Mecmesin and TextureAnalyzer formats with the aid of a single example (measur
data(sampleGelSmooth)
start_height_mm=2.5
plot(pressure~Distance, sampleGelSmooth,type="l",xlim=c(0,3),xlab="Distance [mm]",ylab="Stress [Pa]")
results_texture_analyzer=foam_mechanical_analysis(sampleGelSmooth,start_height=start_height_mm)
sampleGelSmooth$Distance=start_height_mm-sampleGelSmooth$Distance
# The slope columns would have opposite signs with an inverted distance variable
sampleGelSmooth$pressureSlope=-sampleGelSmooth$pressureSlope
sampleGelSmooth$ForceSlope=-sampleGelSmooth$ForceSlope
lines(pressure~Distance, sampleGelSmooth,type="l",col="red")
legend("topleft",legend=c("TextureAnalyzer format","Mecmesin format"),lty=c(1,1),col=c("black","red"))
results_mecmesin=foam_mechanical_analysis_mecmesin(sampleGelSmooth)

cat("Evaluation by foam_mechanical_analysis")
results_texture_analyzer
cat("Evaluation by foam_mechanical_analysis_mecmesin")
results_mecmesin
```

`foam_mechanical_analysis_mecmesin_list`
foam_mechanical_analysis_mecmesin_list

Description

Analysis of list of foam compression curves. This function applies foam_mechanical_analysis_mecmesin to each compression curve and collects the results

Usage

```
foam_mechanical_analysis_mecmesin_list(file_info,data_list,offset_height_column="offset_height_m
```

Arguments

file_info	Description of the compression curves. This is a dataframe, which needs to contain at least the columns with names given by the arguments start_height_column and approximate_gel_touch_point_column, but usually contains more information such as the fabrication conditions. file_info needs to contain as many lines as there are elements in data_list
data_list	List of compression curves, typically produced by read_mecmesin_tab_list
offset_height_column	Name of the off set height column in file_info. This column describes the offset between indicated and absolute height in millimeters.
approximate_gel_height_column	Name of the column in file_info that contains a manual estimation of the approximate gel height, in mm.
...	Possibility to pass down further arguments to foam_mechanical_analysis_mecmesin, used internally

Value

A list with as many elements as there are lines file_info. Each of these elements is the output of foam_mechanical_analysis_mecmesin. As the output of foam_mechanical_analysis_mecmesin is a list itself, the return value is a list of lists.

Author(s)

Thomas Braschler

See Also

foam_mechanical_analysis_mecmesin, used internally

---

force_Hertz_AFM_indentation

force_Hertz_AFM_indentation

Description

Calculation of the force necessary to indent an AFM tip of radius R_tip into a flat substrate to a depth of h_indentation.
This is the classical Hertzian sphere indentation model (Popov 2017).

Usage

```
force_Hertz_AFM_indentation(E_substrate=1e8,nu_substrate=0.5,h_indentation=20e-9,R_tip=8e-9)
```

Arguments

E_substrate	Young modulus of the substrate to be compressed
nu_substrate	Poisson ratio of the substrate to be compressed
h_indentation	Total depth of indentation. This is the indentation due to the physical contact plus the free space due to the deformation of the neighbouring substrate surface. In the Hertz approximation, the two contributions are exactly equal.
R_tip	Tip radius

Details

The formula used here applies to an indentation by a spherical indentor into a substrate of isotropic, homogeneous composition occupying the lower half-plane (Popov 2017).

Value

Force necessary to produce the desired indentation.

Author(s)

Thomas Braschler

References

Popov, L. V. Contact Mechanics and Friction, Springer, Berlin-Heidelberg, 2017

Examples

```
force_Hertz_AFM_indentation()
```

---

force_Sneddon_cone_AFM_indentation

force_Sneddon_cone_AFM_indentation

Description

Calculation of the force necessary to indent a conical tip into a homogeneous material (Sneddon 1965). This typically used in AFM, but can also be applied to a macroscopic indentor.

Usage

```
force_Sneddon_cone_AFM_indentation(E_substrate=1e8,nu_substrate=0.5,h_indentation=20e-9,alpha_co
```

Arguments

E_substrate	Young modulus of the substrate to be compressed
nu_substrate	Poisson ratio of the substrate to be compressed
h_indentation	Depth of the cone indentation
alpha_cone	Half-opening angle of the cone

Details

The formula used here applies to an indentation by a conical indentor into a substrate of isotropic, homogeneous composition occupying the lower half-plane (Sneddon 1965w).

Value

Force necessary to produce the desired indentation.

Author(s)

Thomas Braschler

References

I. N. Sneddon, International Journal of Engineering Science 1965, 3, 47.

Examples

```
force_Sneddon_cone_AFM_indentation()
```

---

gaussianSmoothing *gaussianSmoothing*

Description

Smoothes a curve using a gaussian filter. Using default values, produces classical gaussian smoothing (Blinchikoff and Zverev 1976), but permits additional flexibility to specify the desired output points and specficic handling of the edges to reduce edge artefacts.

Usage

```
gaussianSmoothing(x,y,sd=1,xout=NULL,boundary_extension=0,lm_region_lower=boundary_extension,lm_
```

Arguments

x	The x values formatted as a vector
y	The y values, formatted as a vector of length identical to the length of x
sd	Standard deviation for the smoothing, in terms of x-coordinates
xout	The x values where an output is desired; if NULL is provided, xout is set to x (default)
boundary_extension	Number of elements to linearly extrapolate at the boundaries, to avoid edge effects. Default is 0, meaning no extension
lm_region_lower	Indicates the number of elements on the initial portion of the curve that should be used to set the slope for extrapolation. In more detail, the extrapolation (if boundary_extension¿0) is performed by linear extension of both the x and y values by N=boundary_extension elements. For this a linear model (lm) is calibrated on the first N=lm_region_lower elements of x or y and their indices as a regressor, and then used to extend x and y using continuation of the indices to the negative side.

`lm_region_upper` Indicates the number of elements on the final portion of the curve that should be used to set the slope for extrapolation. In more detail, the extrapolation (if `boundary_extension`¿0) is performed by linear extension of both the `x` and `y` values by `N=boundary_extension` elements. For this a linear model (lm) is calibrated on the last `N=lm_region_upper` elements of `x` or `y` and their indices as a regressor, and then used to extend `x` and `y` using continuation of the indices to the negative side.

Value

A numerical vector of the same length as xout

Author(s)

Thomas+Marina Braschler

References

Filtering in the Time and Frequency Domains, Herman J. Blinchikoff, Anatol I. Zverev, Wiley, 1976

Examples

```
xout=seq(from=0,to=6,length.out=500)
x=1:5
y=runif(length(x))
sd=2
smoothed=gaussianSmoothing(x=x,y=y,sd=sd,xout=xout)
plot(x=x,y=y,type="p")
plot.xy(xy.coords(x=xout,y=smoothed),type="l",col="red")
```

`gaussianSmoothingBlock`

gaussianSmoothingBlock

Description

Smoothes a curve using a gaussian filter; works like gaussianSmoothing, but uses only neighboring points for the smoothing for performance purposes

Usage

```
gaussianSmoothingBlock(x,y,sd=1,xout=NULL,boundary_extension=0,lm_region_lower=boundary_extensio
```

Arguments

`x`	The x values formatted as a vector
`y`	The y values, formatted as a vector of length identical to the length of `x`
`sd`	Standard deviation for the smoothing, in terms of x-coordinates
`xout`	The x values where an output is desired; if NULL is provided, `xout` is set to `x` (default)

`boundary_extension`
: Number of elements to linearly extrapolate at the boundaries, to avoid edge effects. Default is 0, meaning no extension

`lm_region_lower`
: Indicates the number of elements on the initial portion of the curve that should be used to set the slope for extrapolation. In more detail, the extrapolation (if `boundary_extension`¿0) is performed by linear extension of both the `x` and `y` values by `N=boundary_extension` elements. For this a linear model (lm) is calibrated on the first `N=lm_region_lower` elements of `x` or `y` and their indices as a regressor, and then used to extend `x` and `y` using continuation of the indices to the negative side.

`lm_region_upper`
: Indicates the number of elements on the final portion of the curve that should be used to set the slope for extrapolation. In more detail, the extrapolation (if `boundary_extension`¿0) is performed by linear extension of both the `x` and `y` values by `N=boundary_extension` elements. For this a linear model (lm) is calibrated on the last `N=lm_region_upper` elements of `x` or `y` and their indices as a regressor, and then used to extend `x` and `y` using continuation of the indices to the negative side.

`block_size`
: To speed up calculation, use only neighbouring values in the smoothing calculation. `block_size` indicates how many standard deviations away the `x` values are allowed to be from the `xout` value under consideration to be taken into account in the calculation

Value

A numerical vector of the same length as xout

Author(s)

Thomas+Marina Braschler

Examples

```
# First: example without boundary extension

# First, without boundary extension

x=seq(from=0,to=1500,by=0.25)
xout=x
y=runif(length(x))+x/100
sd=50
smoothed=gaussianSmoothing(x=x,y=y,sd=sd,xout=xout)
slope=gaussianSmoothingSlope(x=x,y=y,sd=sd,xout=xout)
slopeBlock=gaussianSmoothingSlopeBlock(x=x,y=y,sd=sd,xout=xout,block_size=6)
plot(x=x,y=y,type="p")
plot.xy(xy.coords(x=xout,y=smoothed),type="l",col="red")
plot.xy(xy.coords(x=xout,y=slope*100),type="l",col="green")
plot.xy(xy.coords(x=xout,y=slopeBlock*100),type="l",col="blue",lty=2)

# Second with boundary extension to avoid edge effects

x=seq(from=0,to=1500,by=0.25)
xout=x
y=runif(length(x))+x/100
sd=50
```

```
boundary_extension=100
smoothed=gaussianSmoothing(x=x,y=y,sd=sd,xout=xout,boundary_extension=boundary_extension)
slope=gaussianSmoothingSlope(x=x,y=y,sd=sd,xout=xout,boundary_extension=boundary_extension)
slopeBlock=gaussianSmoothingSlopeBlock(x=x,y=y,sd=sd,xout=xout,block_size=6,boundary_extension=boundary_ext
plot(x=x,y=y,type="p")
plot.xy(xy.coords(x=xout,y=smoothed),type="l",col="red")
plot.xy(xy.coords(x=xout,y=slope*100),type="l",col="green")
plot.xy(xy.coords(x=xout,y=slopeBlock*100),type="l",col="blue",lty=2)
```

`gaussianSmoothingSlope` *gaussianSmoothingSlope*

Description

Slope of a curve smoothed by a gaussian filter; this is a classical difference of gaussian filter (Gonzalez and Woods 2008) with the possibility to indicate specific output points.

Usage

```
gaussianSmoothingSlope(x,y,sd=1,xout=x,...)
```

Arguments

`x`	The x values formatted as a vector
`y`	The y values, formatted as a vector of length identical to the length of `x`
`sd`	Standard deviation for the smoothing, in terms of x-coordinates
`xout`	The x values where an output is desired
`...`	Supplementary arguments to be passed to gaussianSmoothing, used internally

Value

A numerical vector of the same length as `xout`, giving the estimated slope at each point given in `xout`

Author(s)

Thomas+Marina Braschler

References

Digital Image Processing, 3rd edition, Gonzalez R.C. and Woods R.E., Pearson, 2008, ISBN 9780133002324

Examples

```
# First, without boundary extension. In theory, the smoothened curve should rise linearly, and the slope be cons

x=seq(from=0,to=1500,by=0.25)
xout=x
y=runif(length(x))+x/100
sd=50
smoothed=gaussianSmoothing(x=x,y=y,sd=sd,xout=xout)
slope=gaussianSmoothingSlope(x=x,y=y,sd=sd,xout=xout)
plot(x=x,y=y,type="p")
plot.xy(xy.coords(x=xout,y=smoothed),type="l",col="red")
plot.xy(xy.coords(x=xout,y=slope*1000),type="l",col="green")

# In particular the slope in the above example nicely shows the edge problem. This can be first-order corrected w


x=seq(from=0,to=1500,by=0.25)
xout=x
y=runif(length(x))+x/100
sd=50
boundary_extension=500
smoothed=gaussianSmoothing(x=x,y=y,sd=sd,xout=xout,boundary_extension=boundary_extension)
slope=gaussianSmoothingSlope(x=x,y=y,sd=sd,xout=xout,boundary_extension=boundary_extension)
plot(x=x,y=y,type="p")
plot.xy(xy.coords(x=xout,y=smoothed),type="l",col="red")
plot.xy(xy.coords(x=xout,y=slope*1000),type="l",col="green")
```

`gaussianSmoothingSlopeBlock`
gaussianSmoothingSlopeBlock

Description

Slope of a curve smoothed by a gaussian filter, done blockwise to reduce calculation cost

Usage

```
gaussianSmoothingSlopeBlock(x,y,sd=1,xout=x,block_size=6,...)
```

Arguments

`x`	The x values formatted as a vector
`y`	The y values, formatted as a vector of length identical to the length of `x`
`sd`	Standard deviation for the smoothing, in terms of x-coordinates
`xout`	The x values where an output is desired. If no value is provided, x is used for xout

block_size	To speed up calculation, use only neighbouring values in the smoothing calculation. block_size indicates how many standard deviations away the x values are allowed to be from the xout value under consideration to be taken into account in the calculation
...	Additional arguments passed to gaussianSmoothing, used internally

Value

A numerical vector of the same length as xout, giving the estimated slope at each point given in xout

Author(s)

Thomas+Marina Braschler

Examples

```
x=seq(from=0,to=1500,by=0.25)
xout=x
y=runif(length(x))+x/100
sd=50
smoothed=gaussianSmoothing(x=x,y=y,sd=sd,xout=xout)
slope=gaussianSmoothingSlope(x=x,y=y,sd=sd,xout=xout)
slopeBlock=gaussianSmoothingSlopeBlock(x=x,y=y,sd=sd,xout=xout,block_size=5)
plot(x=x,y=y,type="p")
plot.xy(xy.coords(x=xout,y=smoothed),type="l",col="red")
plot.xy(xy.coords(x=xout,y=slope*1000),type="l",col="green")
plot.xy(xy.coords(x=xout,y=slopeBlock*1000),type="l",col="blue")

# In particular the slope in the above example nicely shows the edge problem. This can be first-order corrected w


x=seq(from=0,to=1500,by=0.25)
xout=x
y=runif(length(x))+x/100
sd=50
boundary_extension=500
smoothed=gaussianSmoothing(x=x,y=y,sd=sd,xout=xout,boundary_extension=boundary_extension)
slope=gaussianSmoothingSlope(x=x,y=y,sd=sd,xout=xout,boundary_extension=boundary_extension)
slopeBlock=gaussianSmoothingSlopeBlock(x=x,y=y,sd=sd,xout=xout,boundary_extension=boundary_extension,block_
plot(x=x,y=y,type="p")
plot.xy(xy.coords(x=xout,y=smoothed),type="l",col="red")
plot.xy(xy.coords(x=xout,y=slope*1000),type="l",col="green")
plot.xy(xy.coords(x=xout,y=slopeBlock*1000),type="l",col="blue")
```

Description

Provides a compensation function that allows to estimate how much distance is lost at a given force due to TextureAnalyzer compliance. Relevant only for very hard subtrates, for hydrogels, this is typically negligeable.

Usage

```
get_frame_compensation_function(filename,min_force=5, max_force = 20)
```

Arguments

filename	Texture analyzer tab file containing compression data without sample, solely against the bench or other holder surface
min_force	Minimum force required for inclusion into slope regression region
max_force	Maximum force for inclusion into slope regression region

Details

The intended use of this function is to adjust the distance measurements to compensate for frame compliance. Indeed, frame compliance means that the set (=reported) travelling distance (motor steps) is slightly larger than the distance actually travelled by the chuck surface, a small part being lost due to frame compliance. This small part can be obtained from the Force if passed to the output function provided by the `get_frame_compensation_function`.

For soft samples, the frame compensation makes only a very minor difference and is not generally necessary; for very soft samples, buoyancy and capillary effects are generally more important.

Value

Distance compensation function. Takes the measured force as an argument, and indicates the equivalent chuck movement distance arising from frame compliance

Author(s)

Thomas+Marina Braschler

References

Filtering in the Time and Frequency Domains, Herman J. Blinchikoff, Anatol I. Zverev, Wiley, 1976

Examples

```
# Get the frame compensation function
path=system.file("sampleData",package="textureAnalyzerGels")
frame_compensation_function=get_frame_compensation_function(paste(path,"frame_stiffness.tab",sep="/"))

# Comparative plotting
data(sampleGel)
frame_stiffness = read_texture_analyzer_tab(paste(path,"frame_stiffness.tab",sep="/"),chuck_diameter=4e-3)
frame_stiffness_smooth=smooth_texture_analyzer_data(frame_stiffness, sd = 0.005,lm_region_upper_mm=0.05)
touch_point_frame_compensation=find_initial_touch_point(frame_stiffness_smooth)
```

```
plot(pressure ~ Distance, sampleGel,type="l",xlim=c(0,start_height_mm+0.5),main="Frame stiffness compensatio
# This is the corresponding plot without sample
lines(frame_stiffness$Distance-touch_point_frame_compensation+start_height_mm, frame_stiffness$pressure,typ
# Frame compensation
sampleGel$Distance_compensated = sampleGel$Distance-frame_compensation_function(sampleGel$Force)
lines(pressure ~ Distance_compensated, sampleGel,type="l",col="red")
# Frame compensation on the frame compensation measurement
frame_stiffness$Distance_compensated = frame_stiffness$Distance-frame_compensation_function(frame_stiffness
lines(frame_stiffness$Distance_compensated-touch_point_frame_compensation+start_height_mm, frame_stiffness$

legend("topleft",legend=c("sample Gel","sample Gel compensated", "no sample control","no sample control compen
```

`get_young_modulus` *get_young_modulus*

Description

Estimates the Young modulus from force-distance curves obtained by a compression cycle

Usage

```
get_young_modulus(theData, approximate_gel_touch_point = NULL, gel_thickness = 1, strain = 0.1, dop
```

Arguments

`theData`	Force-Distance data, typically from a "down"-"up" cycle of the chuck; the data needs to be smooth and contain already the ForceSlope column, which can be obtained by smooth_texture_analyzer_data
`approximate_gel_touch_point`	Initial guess for the gel touch point (will be refined by a call to find_landmarks_gel_compression
`gel_thickness`	Gel thickness, in the same units as is the "Distance
`strain`	Compressive strain on which the evaluation of the Young modulus should be centered
`doplot`	Whether or not the routine should provide graphical output (regression lines in the pressure as a function of distance graph)
`do_touch_point_estimation`	Whether or not the touch point should be estimated from the data. If `do_touch_point_estimation=FALSE`, `approximate_gel_touch_point` needs to be provided and will be used as is for the gel touch point
`subtract_chuck_buoyancy`	If TRUE, subtract the effect of the chuck buoyancy in water (this assumes the density of the medium to be 1000kg/mˆ3, and the distance to be in mm, and also that the chuck has uniform cross section area identical to sample cross section area.

Details

The Young modulus estimation here is by attempting to evaluate the slope of the stress-strain curve (ATSM E111, via lm) up to a given maximum `strain` value. No correction for possible non-linearity is attempted. If significant non-linearity exists, non-linear models such as the Ogden model (Ogden 1972, ogden_model), or, in the case of the existence of a plateau (Gibson 1997), foam_mechanical_analysis should be used instead.

Value

Vector of young moduli, one per direction (up or down)

Author(s)

Thomas Braschler

References

ATSM standard E111: https://www.astm.org/Standards/E111.htm

Gibson, L. J. & Ashby, M. F. Cellular Solids: Structure and properties. (Cambridge University Press, 1997).

Ogden, R. W. (1972), Large Deformation Isotropic Elasticity - On the Correlation of Theory and Experiment for Incompressible Rubberlike Solids,Proceedings of theRoyal Society A: Mathematical, Physical and Engineering Sciences, 326(1567), 565-584.

See Also

find_initial_touch_point, used internally

Examples

```
data(sampleGelSmooth)
get_young_modulus(sampleGelSmooth)
```

---

ogden_model *ogden_model*

Description

Elastic stress as a function of the material stretch, according to the Ogden model

Usage

```
ogden_model(mu,alpha,stretch)
```

Arguments

`mu`	Vector of Young moduli at zero stretch (see details)
`alpha`	Vector of coefficients
`stretch`	Stretch at which the stress should be evaluated

Details

The Ogden model (Ogden 1972) is a power model for fitting Young moduli as a function of the material stretch (stretch = delta(L) / L = 1+strain). The formula can be expressed variously, we use here :

```
Stress=SUM mu*((stretch^alpha)-1)/alpha
```

The reason for this particular choice is that this causes the contribution of the different powers (`alpha`) to the Young modulus (d(stress)/d(strain)) for the undeformed material (`stretch=1`) to depend solely on the `mu` but not `alpha` values :
`lim(E,stretch=1)=SUM mu`
Further, `stress=0` for `stretch=0`.

Value

Stress at the given stretch values

Author(s)

Thomas Braschler

References

Ogden, R. W. (1972), Large Deformation Isotropic Elasticity - On the Correlation of Theory and Experiment for Incompressible Rubberlike Solids,Proceedings of theRoyal Society A: Mathematical, Physical and Engineering Sciences, 326(1567), 565-584.

Examples

```
stretch=seq(from=1,to=2,by=0.01)
stress=ogden_model(mu=c(1e3,1e3),alpha=c(6,-3),stretch=stretch)
# Secant Young modulus
E=stress/(stretch-1)

plot(E ~ stretch)
```

`ogden_model_energy` *ogden_model_energy*

Description

Elastic stress energy as a function of the material stretch, according to the Ogden model (Ogden 1972). This is the integral of the Ogden stress (as given by ogden_model).

Usage

```
ogden_model_energy(mu,alpha,stretch)
```

Arguments

mu	Vector of Young moduli at zero stretch (see details)
alpha	Vector of coefficients
stretch	Stretch at which the stress energy should be evaluated

Details

The integration constant for the stress energy to be 0 at the undeformed state (stretch=1 or equivalently stress=0).

Value

Stress energy at the given stretch values

Author(s)

Thomas Braschler

References

Ogden, R. W. (1972), Large Deformation Isotropic Elasticity - On the Correlation of Theory and Experiment for Incompressible Rubberlike Solids,Proceedings of theRoyal Society A: Mathematical, Physical and Engineering Sciences,326(1567), 565-584.

Examples

```
stretch=seq(from=1,to=2,by=0.01)
stress_energy=ogden_model_energy(mu=c(1e3,1e3),alpha=c(6,-3),stretch=stretch)


plot(stress_energy ~ stretch)
```

---

ogden_model_symmetric *ogden_model_symmetric*

Description

Convenience function, giving the elastic stress as a function of the material stretch, according to the Ogden model (Ogden 1972), by using symmetric pairs of mu and alpha. Uses ogden_model

Usage

```
ogden_model_symmetric(mu,alpha,stretch)
```

Arguments

mu	Vector of Young moduli at zero stretch (see details)
alpha	Vector of coefficients
stretch	Stretch at which the stress values should be evaluated

Value

Stress at the given stretch values, obtained by calling ogden_model via:

```
ogden_model(c(mu,mu),c(alpha,-alpha),stretch)
```

Author(s)

Thomas Braschler

References

Ogden, R. W. (1972), ?Large Deformation Isotropic Elasticity - On the Correlationof Theory and Experiment for Incompressible Rubberlike Solids?,Proceedings of theRoyal Society A: Mathematical, Physical and Engineering Sciences326(1567), 565?584.

Examples

```
stretch=seq(from=0.5,to=2,by=0.01)
stress=ogden_model_symmetric(mu=c(1e3),alpha=c(3),stretch=stretch)
# Sekant Young model
E=stress/(stretch-1)

plot(E ~ stretch)
```

---

ogden_model_symmetric_energy

ogden_model_symmetric_energy

Description

Convenience function, giving the elastic stress energy as a function of the material stretch, according to the integrated Ogden model (Ogden 1972), by using symmetric pairs of `mu` and `alpha`. Uses ogden_model_energy

Usage

```
ogden_model_symmetric_energy(mu,alpha,stretch)
```

Arguments

`mu`	Vector of Young moduli at zero stretch (see details)
`alpha`	Vector of coefficients
`stretch`	Stretch at which the stress energy should be evaluated

Value

Stress energy at the given stretch values, obtained by calling ogden_model_energy via:

```
ogden_model_energy(c(mu,mu),c(alpha,-alpha),stretch)
```

Author(s)

Thomas Braschler

References

Ogden, R. W. (1972), 'Large Deformation Isotropic Elasticity - On the Correlationof Theory and Experiment for Incompressible Rubberlike Solids',Proceedings of theRoyal Society A: Mathematical, Physical and Engineering Sciences326(1567), 565?584.

Examples

```
stretch=seq(from=0.5,to=2,by=0.01)
stress_energy=ogden_model_symmetric_energy(mu=c(1e3),alpha=c(3),stretch=stretch)



plot(stress_energy ~ stretch)
```

`plot_stress_strain` *plot_stress_strain*

Description

Draws a stress strain plot from compression data as read by read_texture_analyzer_tab and typically smoothened by smooth_texture_analyzer_data

Usage

```
plot_stress_strain(compression_data, analysis,start_height_mm, plot_evaluation=TRUE,pressure_colu
title="stress-strain")
```

Arguments

`compression_data`	Compression data as read by read_texture_analyzer_tab
`analysis`	Analyzed data as evaluated by foam_mechanical_analysis
`start_height_mm`	Starting height for the compression test, a priori in mm. Needs to be recorded when doing the test as the textureAnalyzer saves the data with starting Distance 0 at the beginning of the test, regardless of the absolute height
`plot_evaluation`	Indicates whether analysis data should be indicated directly in the plot
`pressure_column`	Possibility to indicate an alternative pressure column, for instance when correcting for buoyancy
`distance_column`	Possibility to indicate an alternative distance column
`direction_column`	Possibility to indicate an alternative direction column
`title`	Title of the graphic

Details

Converts the distance values to strain values by using the `start_height_mm` information and the touchpoint information contained in the `analysis` argument. Together with the pressure column in `compression_data`, a stress-strain plot is drawn. In addition, if `plot_evaluation` is `TRUE`, detailed data from the `analysis` variable is included. In text form, the Young moduli at compression (direction "down") and relaxation (direction "up") are provided, both for the elastic and plateau segement. The touchpoint, central elastic point and central plateau points are marked with red circles, and the relevant slopes for the Young modulus calculation are indicated as red lines.

Value

None

Author(s)

Thomas Braschler

Examples

```
path = system.file('sampleData', package = 'textureAnalyzerGels');
sampleGel = smooth_texture_analyzer_data(read_texture_analyzer_tab(paste(path,"sampleGel.tab",sep="/"),chuc
start_height_mm=2.5

analysis=foam_mechanical_analysis(
sampleGel,start_height=start_height_mm,approximate_gel_touch_point=0.15)

plot_stress_strain(sampleGel, analysis,start_height_mm, title="Sample plot stress-strain")
```

---

pore_radius_from_water_diffusion

pore_radius_from_water_diffusion

Description

Estimation of the pore size in a hydrogel from the apparent water diffusion coefficient

Usage

```
pore_radius_from_water_diffusion(D=1e-9,nu=1e-3,E=1e4)
```

Arguments

`D`	Apparent diffusion coefficient (can be obtained from stress relaxation experiments in covalently crosslinked gels)
`nu`	Viscosity of the pore fluid
`E`	Young modulus after stress relaxation (typically, half of the Young modulus before stress relaxation

Details

This function is based on a direct comparison of water diffusion under compression in a hydrogel and Poiseuille's law to obtain an order-of magnitude estimate for pore sizes in highly porous media. See also Braschler et al. 2015.

Covalently crosslinked hydrogel present viscoelastic behaviour which is almost exclusively due to water displacement in the pores. Indeed, under reasonable deformation, one does not expect plastic deformation of the covalently crosslinked gel skeleton itself, only displacement of water pore volume and local stretching or folding of the polymer chains. This gives rise to a characteristic stress relaxation behavior, where the gel is incompressible on short time scales, but compressible by loss of pore water at long time scales. On a fundamental level, the pore pressure is due to volume load with pore medium :

```
P_pore = E*phi
```

where `E` is the bulk elastic modulus (technically, given as the drained bulk modulus by `E/3/(1-2*poisson_ratio)`, but which is about the same as the Young modulus `E` for typical materials with a Poisson ratio of about 0.3) , whereas `phi=delta(V)/V` is the relative volume load with pore medium. In a relaxation experiment, `phi` is initially given by the rapid compression, but then drops to `phi=0` as the gel equilibrates by expulsion of pore solution. Darcy's law gives the flow velocity `v` due to the ensuing pressure gradient:

```
v = K/nu * (dP_pore/dx) = K*E/nu *(dphi/dx)
```

where `K` is the Darcy permeability (in m^2), and nu the viscosity of the pore medium (in Pa*s). Since `v` corresponds to the transport of excessive pore fluid, we can write `v= dphi/dt * 1/A`, such that the equivalence with a water diffusion law becomes evident:

```
dphi/dt = A*(K*E/nu)*(dphi/dx)
```

and therefore

D=K*E/nu

We can approximately relative the Darcy-permeability coefficient `K` to the pore size, by considering `N` parallel channels. The volume flow rate of `N` parallel channels, occupying a total cross-sectional area `A`, is given by Poiseuille's law:

```
dphi/dt = N*pi*r^4/8/nu*(dP_pore/dx)
```

If we set N=A/(pi*r^2), this becomes:

```
dphi/dt = A*r^2/8/nu*(dP_pore/dx) =>
v=r^2/8/nu*(dP_pore/dx)
```

which, by comparison with Darcy's law, relates the Darcy permeability coefficient to the pore size by:

```
K=r^2/8
```

Finally, if we now `D`,`E` and `nu`, we can estimate the pore size by:

`r=sqrt(8*K)=sqrt(8*D*nu/E)`

This is the formula used by the function to provide an estimate of the pore radius from the apparent water diffusion coefficient `D`.

Experimentally, a rapid applicaton of a uniaxial strain (`epsilon_z`) in a sample with homogeneous height and shape in the z-direction and with free lateral boundaries can be used to introduce an initially homogeneous volume loading `phi`. Typically, such a sample is a disk, placed between a substrate and a piston for loading on its flat faces. On a short time scale, hydrogels are typically nearly incompressible, such that the total volumetric strain is 0: `epsilon_z+epsilon_x+epsilon_y=0` and therefore `epsilon_x + epsilon_y = -epsilon_z`. The resulting stress in the z-direction is balanced by the loading piston, but the planar stress `sigma_xy=E*(epsilon_x+epsilon_y)=-E*epsilon_z` must be balanced by the pore pressure, such that we have: `P_pore = -E*epsilon_z` everywhere in the sample. In terms of volume loading, this can be interpreted as having a homogeneous, and controlled excess of pore medium `phi=-epsilon_z` everywhere. The apparent Young modulus under these conditions is `2*E`, since the piston must sustain both the stress due to `epsilon_z` and the pore pressure.

During the hydrogel relaxation, the excess pore water diffuses out of the gel, and ultimately, `P_pore=0`, with a drop of the apparent Young modulus by a factor of 2. Precise knowledge of the gel geometry can be used to estimate a diffusion coefficient for the disappearance of the excess pore water from the stress relaxation characteristics (for instance, by using diffusion_from_disk) for a disk-shaped sample. The present function can then be used to estimate the hydrogel pore size.

Value

Estimation of the pore radius

Author(s)

Thomas Braschler

References

Braschler T, Songmei W, Wildhaber F, Bencherif SA, Mooney DJ, "Soft nanofluidics governing minority ion exclusion in charged hydrogels", Soft Matter, Issue 20, 2015, specific details in Supplementary 2.

Examples

```
pore_radius_from_water_diffusion(D=1e-9,nu=1e-3,E=1e4)
```

`pore_size_from_hydrodynamic_pressure`

pore_size_from_hydrodynamic_pressure

Description

Order of magnitude estimation of the pore size from stress hysteresis. The idea is to estimate approximately an effective pore size that could explain higher stress by the pressure build-up due to lag in pore fluid evacuation as opposed to underpressure when relaxing the same gel. In general, such differences are large with micro- or nanoporous gels, but very small with macroporous gels and thus difficult to measure at all but very high compression rates in macroporous gels.

Usage

```
pore_size_from_hydrodynamic_pressure(P_up=500,P_down=1000,gel_thickness=1e-3,v_compression=1e-5,
```

Arguments

P_up	Pressure (stress) observed at a given strain during relaxation ("up" movement of the chuck)
P_down	Pressure (stress) observed during compression ("down" movement of the chuck), this is generally larger than P_up
gel_thickness	Height of the gel, in meters
v_compression	Speed of the chuck, identical except for direction in down and up movement. Units: m/s
viscosity	Viscosity of the pore fluid, in Pa*s
disk_radius	Radius of the disk sample, in meters

Details

This function is based on an order of magnitude estimation. For a single pore, Poiseuille's law of laminar flow resistance in a cylindrical tube gives:
Qsingle_pore = deltaP*pi*r^4/8/viscosity/disk_radius
where deltaP=P_down-P_up, and r is the characteristic pore radius we are looking for. There are many more or less warranted assumptions, such as all pores having the same length of disk_radius, all pores being cylindrical and of identical radius r and so forth, but one has to keep in mind that this is an order-of-magnitude estimation only.

For N pores in parallel, the flow rate is N times higher; it must at the same time match the pore fluid evacuation rate given by the volume displacement of the chuck:
Qtotal = v_compression*pi*disk_radius^2 = N*Qsingle_pore
The number of parallel pores can roughly be obtained from the cross section surface, roughly at half the disk radius:
N = disk_radius*pi*gel_thickness/pi/r^2
Assembling, we get: Qtotal = disk_radius*pi*gel_thickness/pi/r^2 * deltaP*pi*r^4/8/viscosity/disk_ra
= disk_radius/disk_radius * pi/pi*pi * gel_thickness * r^4/r^2 * deltaP/8/viscosity
=
Qtotal = pi*gel_thickness*r^2*deltaP/8/viscosity = v_compression*pi*disk_radius^2

The final expression, used in this function, for the pore radius estimation is thus:
=> r = disk_radius*sqrt(v_compression *8 *viscosity /gel_thickness/deltaP)

Value

Numerical estimation of the pore radius

Author(s)

Thomas Braschler

Examples

```
pore_size_from_hydrodynamic_pressure(P_up=500,P_down=1000,gel_thickness=1e-3,v_compression=1e-3,viscosity=
```

`read_mecmesin_tab` *read_mecmesin_tab*

Description

Reads a Mecmesin text file from the disk. This is an adaptation of the read_texture_analyzer_tab function to the specific format of Mecmesin text export (tested with Mecmesin MultiTest 2.5 dV)

Usage

```
read_mecmesin_tab(file, sample_diameter = 0.022, downup = TRUE)
```

Arguments

`file`	File path for the tabulated text file.
`sample_diameter`	Diameter of the chuck (or sample, whichever is smaller) used to compress the gel. Indication in meters, the chuck is assumed to be circular.
`downup`	Chuck displacement down (compression) followed by equal movement up (release)?

Details

Calls read_texture_analyzer_tab with specific parameters internally.

Value

Dataframe with the following columns:

`Distance`	Distance of movement of the chuck, as provided by the Mecmesin apparatus; the units are expected to be in mm
`Force`	Force as measured by the Texture Analyzer, in grams
`direction`	Interpretation in terms of downward and upward movement, possible values are "down" and "up". This is with respect to the phases of the program (assumed down first), not necessarily physical orientation.
`Time (s)`	Time of acquisition, as given by the Mecmesin apparatus
`pressure`	Calculated pressure (stress) exercised by the chuck; since pressure is given by force / area, the area is estimated from the `sample_diameter` argument

Author(s)

Thomas Braschler

References

Filippova A, Bonini F, Efremov L, Locatelli M, Preynat-Seauve O, Beduer A, Krause KH, Braschler T, 2021: Neurothreads: development of supportive carriers for mature dopaminergic neuron differentiation and implantation, Raw data set: https://doi.org/10.5281/zenodo.4441090 accompanying the corresponding scientific publication at Biomaterials (https://doi.org/10.1016/j.biomaterials.2

Examples

```
# The data for this example is taken from https://doi.org/10.5281/zenodo.4441090 (Filippova et al. 2021), Suppl
# From there, we copied the file "raw-TensileStrenght(peellike)(V1)-Modified Version-sample-2.txt" here to ser
path = system.file('sampleData/mecmesin/data/sample-2.txt', package = 'textureAnalyzerGels');
exampleTraction = read_mecmesin_tab(path,sample_diameter=4.76e-3,downup=FALSE)
plot(pressure ~ Distance, exampleTraction,type="b",xlab="Distance [mm]", ylab="Stress [Pa]")
```

---

read_mecmesin_tab_list *read_mecmesin_tab_list*

Description

Reads a series of texture analyzer tab files

Usage

```
read_mecmesin_tab_list(file_info,root_folder=getwd(),folder_column="Folder",file_column="File",
sample_diameter_column="diameter_mm",do_plot=TRUE,do_smoothing=TRUE,lines_to_read=NULL,downup=TR
```

Arguments

file_info	Dataframe describing the files to be read. Needs to contain at least the following columns: 1. A folder column, describing the sub-folder where the individual files are located. The name of this column is given by the argument folder_column. 2. A file column indicating the filename of the file to be read. The name of this column is given by file_column. 3. A sample or gel diameter column, where the diameter of the sample or gel is indicated (whichever is smaller, basically). The name of this column is given by the sample_diameter_column argument.
root_folder	Common root folder where all the tab-files to be read are located, defaults to the current working directory as given by getwd.
folder_column	Name of the column in file_info that contains the folder location.
file_column	Name of the column in file_info that contains the filename.
sample_diameter_column	Name of the column in file_info that contains the relevant sample/gel diameters. The sample or gel diameters should be in millimeters.
do_plot	Indicates whether or not plots should be drawn while reading the files
do_smoothing	Indicates whether or not smoothing by smooth_texture_analyzer_data should be performed when reading the files
lines_to_read	Possibility to restrict the lines to be read, a numerical vector indicating the lines to be read.
downup	sample displacement down (compression) followed by equal movement up (release)?
...	Additional arguments to be passed to smooth_texture_analyzer_data. These arguments should be named according to the arguments given in smooth_texture_analyzer_dat

Details

The function will run through all the lines of `file_info` and read the file specified by the file and folder columns (which need to be located within folder given by the `root_folder` argument). To do so, it uses read_mecmesin_tab, and, if `do_smoothing` is `TRUE`, smooth_texture_analyzer_data as well. If `do_plot` is true, it will plot the data for each line as well. The `lines_to_read` argument can be used to restrict the file reading to certain lines; this is primarily for debugging purposes, as the function may take quite a long time if there are many entries in `file_info`.

Value

A list with as many elements as there are rows in `file_info`. Each element in the list is the output of read_mecmesin_tab, optionally processed by smooth_texture_analyzer_data

Author(s)

Thomas Braschler, Patrick Burch

References

Filippova A, Bonini F, Efremov L, Locatelli M, Preynat-Seauve O, Beduer A, Krause KH, Braschler T, 2021: Neurothreads: development of supportive carriers for mature dopaminergic neuron differentiation and implantation, Raw data set: https://doi.org/10.5281/zenodo.4441090 accompanying the corresponding scientific publication at Biomaterials (https://doi.org/10.1016/j.biomaterials.2

Examples

```
# The data for this example is taken from https://doi.org/10.5281/zenodo.4441090 (Filippova et al. 2021), Suppl
path = system.file('sampleData/mecmesin', package = 'textureAnalyzerGels');
tensile_test_file_info = read.xls(file.path(path,"files_stretch.xlsx") )
tensile_test_data=read_mecmesin_tab_list(tensile_test_file_info,root_folder=path,do_smoothing=TRUE,do_plot=
```

`read_texture_analyzer_tab` *read_texture_analyzer_tab*

Description

Reads a texture analyzer file from the disk. The units are assumed to mm for the distance, and g for the force

Usage

```
read_texture_analyzer_tab(file, chuck_diameter = 0.004,aggregate_by="Distance",downup=TRUE,dec=".
```

Arguments

file	File path for the tabulated text file.
chuck_diameter	Diameter of the chuck used to compress the gel. Indication in meters, the chuck is assumed to be circular.
aggregate_by	Column to be used for aggregation. Provide FALSE if no aggregation is desired. If aggregate_by="Time" is indicated, the Time column will be rounded to integers prior to aggregation
downup	Chuck displacement down (compression) followed by equal movement up (release)?
dec	Decimal point when reading the text files
skip_lines	Possibility to skip lines in front of the actual to data to read
unit_line	Is there a line in the text file (after the variable names) that contains unit information? If yes pass 1, otherwise 0
sep	Separator to be passed to read.table
Force_column	Name of the column where the force is stored. The expected unit is grams for now
Distance_column	Name of the column where the distance travelled by the chuck is stored. The expected unit is mm for now
use_fill_for_read_table	Set the fill argument to read.table, used internally. Provide TRUE if there are incomplete lines in the .tab files that should be completed with empty columns

Details

The routine expects the file to contain force data from both the downward trajectory and the upward trajectory; it assigns the first half of the points to the down trajectory, and the second half to the up trajectory

Value

Dataframe with the following columns:

Distance	Distance of downward movement of the chuck, as given by the texture analyzer; the units are expected to be in mm
Force	Force as measured by the Texture Analyzer, in grams
direction	Interpretation in terms of downward and upward movement, possible values are "down" and "up"
Time	Time of acquisition, as given by the Texture Analyzer
pressure	Calculated pressure (stress) exercised by the chuck; since pressure is given by force / area, the area is estimated from the chuck_diameter argument

Author(s)

Thomas Braschler

Examples

```
path = system.file('sampleData', package = 'textureAnalyzerGels');
sampleGel = read_texture_analyzer_tab(paste(path,"sampleGel.tab",sep="/"),chuck_diameter=4e-3)
```

`read_texture_analyzer_tab_list` *read_texture_analyzer_tab_list*

Description

Reads a series of texture analyzer tab files

Usage

```
read_texture_analyzer_tab_list(file_info,root_folder,folder_column="Folder",file_column="File",
      chuck_diameter_column="diameter_mm",do_plot=TRUE,do_smoothing=TRUE,lines_to_read=NULL,
      aggregate_by="Distance",downup=TRUE,dec=".",skip_lines=1,unit_line=1,sep="",
      Force_column="Force",Distance_column="Distance",use_fill_for_read_table=FALSE,...)
```

Arguments

`file_info`	Dataframe describing the files to be read. Needs to contain at least the following columns: 1. A folder column, describing the sub-folder where the individual files are located. The name of this column is given by the argument `folder_column`. 2. A file column indicating the filename of the file to be read. The name of this column is given by `file_column`. 3. A chuck or gel diameter column, where the diameter of the chuck or gel is indicated (whichever is smaller, basically). The name of this column is given by the `chuck_diameter_column` argument.
`root_folder`	Common root folder where all the tab-files to be read are located.
`folder_column`	Name of the column in `file_info` that contains the folder location.
`file_column`	Name of the column in `file_info` that contains the filename.
`chuck_diameter_column`	Name of the column in `file_info` that contains the relevant chuck/gel diameters. The chuck or gel diameters should be in millimeters.
`do_plot`	Indicates whether or not plots should be drawn while reading the files
`do_smoothing`	Indicates whether or not smoothing by smooth_texture_analyzer_data should be performed when reading the files
`lines_to_read`	Possibility to restrict the lines to be read, a numerical vector indicating the lines to be read.
`aggregate_by`	Aggregation argument passed to read_texture_analyzer_tab, used internally
`downup`	Chuck displacement down (compression) followed by equal movement up (release)?
`dec`	Decimal point when reading the text files
`skip_lines`	Possibility to skip lines in front of the actual to data to read
`unit_line`	Is there a line in the text file (after the variable names) that contains unit information? If yes pass 1, otherwise 0
`Force_column`	Name of the column where the force is stored. The expected unit is grams for now

`Distance_column`
: Name of the column where the distance travelled by the chuck is stored. The expected unit is mm for now

`sep`
: Separator to be passed to read.table

`use_fill_for_read_table`
: Set the fill argument to read.table, used internally. Provide TRUE if there are incomplete lines in the .tab files that should be completed with empty columns

`...`
: Additional arguments to be passed to smooth_texture_analyzer_data. These arguments should be named according to the arguments given in smooth_texture_analyzer_dat

Details

The function will run through all the lines of `file_info` and read the file specified by the file and folder columns (which need to be located within folder given by the `root_folder` argument). To do so, it uses read_texture_analyzer_tab, and, if `do_smoothing` is `TRUE`, smooth_texture_analyzer_data as well. If `do_plot` is true, it will plot the data for each line as well. The `lines_to_read` argument can be used to restrict the file reading to certain lines; this is primarily for debugging purposes, as the function may take quite a long time if there are many entries in `file_info`.

Value

A list with as many elements as there are rows in `file_info`. Each element in the list is the output of read_texture_analyzer_tab, optionally processed by smooth_texture_analyzer_data

Author(s)

Thomas Braschler, Patrick Burch

Examples

```
path = system.file('sampleData/2016_08_30_Patrick_sample', package = 'textureAnalyzerGels');
CMC_concentration_file_info = read.xls(file.path(path,"file_listing.xlsx") )
CMC_concentration_data=read_texture_analyzer_tab_list(file_info=CMC_concentration_file_info,root_folder=pat
```

---

sampleGel *Sample Gel compression data*

Description

Compression curve for a 1.5 percent CMC (MW=90kDa) cryogel, crosslinked at every 10th residue by adipic dihydrazide (crosslinking pH is 6.7).

Usage

```
data(sampleGel)
```

Format

A data frame named `sampleGel` with the following variables

`Distance` Distance travalled by the chuck from its original height, in millimeters

`direction` Direction of movement (i.e. down, then up)

`Force` Force measured by the texture Analyzer. In grams.

`Time` Time elapsed since the beginning of the compression experiment. In seconds.

`pressure` Pressure (stress) as calculated form Force / gel cross section are. In Pascals

In addition a numerical value names `start_height_mm` that indicates the height of the chuck at the beginning of the experiment (where the `Distance` travelled is 0).

Author(s)

Patrick Burch
Thomas Braschler

Examples

```
# Generation of the data
path = system.file('sampleData', package = 'textureAnalyzerGels');
sampleGel = read_texture_analyzer_tab(paste(path,"sampleGel.tab",sep="/"),chuck_diameter=4e-3)
start_height_mm=2.5

# Loading of the data and elementary plotting
data(sampleGel)
plot(sampleGel$Distance, sampleGel$pressure/1e3,type="l",xlab="Distance [mm]",ylab="Pressure [kPa]")
```

---

sampleGelSmooth *Sample Gel compression data, smoothened*

Description

Compression curve for a 1.5 percent CMC (MW=90kDa) cryogel, crosslinked at every 10th residue by adipic dihydrazide (crosslinking pH is 6.7). Smoothened by smooth_texture_analyzer_data

Usage

```
data(sampleGelSmooth)
```

Format

A data frame names `sampleGelSmooth` with the following variables

`Distance` Distance travalled by the chuck from its original height, in millimeters

`direction` Direction of movement (i.e. down, then up)

`Force` Force measured by the texture Analyzer. In grams.

`Time` Time elapsed since the beginning of the compression experiment. In seconds.

`pressure` Pressure (stress) as calculated form Force / gel cross section are. In Pascals

`ForceSlope` Slope of the smoothened force curve, in gram/mm

`pressureSlope` Slope of the smoothened pressure curve, in Pa/mm

In addition a numerical value names `start_height_mm` that indicates the height of the chuck at the beginning of the experiment (where the `Distance` travelled is 0).

Author(s)

Patrick Burch
Thomas Braschler

Examples

```
# Generation of the data
data(sampleGel)
sampleGelSmooth=smooth_texture_analyzer_data(sampleGel, sd = 0.05,lm_region_upper_mm=0.05)
start_height_mm=2.5

# Loading of the data and elementary plotting
data(sampleGel)
data(sampleGelSmooth)
plot(sampleGel$Distance, sampleGel$pressure/1e3,type="l",xlab="Distance [mm]",ylab="Pressure [kPa]",ylim=c(0
plot.xy(xy.coords(sampleGelSmooth$Distance,sampleGelSmooth$pressure/1e3),type="l",col="red")
```

`smooth_texture_analyzer_data` *smooth_texture_analyzer_data*

Description

Smoothes the force and pressure curves using a gaussian kernel

Usage

```
smooth_texture_analyzer_data(theData, sd = 0.04,cycle_col="cycle",boundary_extension_mm=0.2,lm_re
```

Arguments

`theData`	Data for the force and distance relationship. Needs to have at least the columns "direction", "Distance", "Force" and "pressure"
`sd`	Standard deviation for the Gaussian kernel, in the same units as the Distance column
`cycle_col`	Column indicating compression cycles, for data analyzed by detect_compression_cycles. Defaults to a column named "cycle", but if this column cannot be found in `theData`, this argument is ignored
`boundary_extension_mm`	To correct for boundary effects, use linear extension beyond the measured region. This should be typically a few standard deviations `sd`

`lm_region_lower_mm`
: Region used to establish the linear regression for the extension at the lower `Distance` values. Should be within the region where the gel is not touched

`lm_region_upper_mm`
: Region used to establish the linear regression for the extension at the upper `Distance` values. This is a bit delicate to set, as a longer region is more precise, but the pressure vs. Distance line should still be reasonable straight.

`block_size`
: For optimization of calculation time in the smoothing calculation, it is possible to carry out the smoothing block-wise. See gaussianSmoothingBlock and gaussianSmoothingSlopeBlock for details. Provide `block_size=0` to avoid block-wise calculation during smoothing

Details

Calls gaussianSmoothing internally

Value

Variable of the same structure and values as the `theData` argument, with the following changes:
- Force column now contains smoothened values
- pressure column now contains smoothened values
-new column ForceSlope, which contains the smoothened, momentaneous slope (delta Force / delta Distance)
- new column pressureSlope, which contains the smoothened, momentaneous slope (delta pressure / delta distance)

Author(s)

Thomas Braschler

Examples

```
data(sampleGel)
plot(Force ~ Distance, sampleGel,type="l")
sampleGelSmooth=smooth_texture_analyzer_data(sampleGel, sd = 0.05,lm_region_upper_mm=0.05)
plot.xy(xy.coords(sampleGelSmooth$Distance,sampleGelSmooth$Force),type="l",col="red")
```

`tissue_mechanical_analysis`
tissue_mechanical_analysis

Description

Analysis of biological tissue compression curves. Stress-strain curves from biological tissues are usually non-linearly increasing. This function evaluates the Young modulus as the slope in the stress strain curves for small strain segments: strain 0.05-0.15, strain 0.15 to 0.25, then 0.25 to 0.35 and so forth. This gives a quick, discrete overview over how the Young modulus increases with increasing strain.

Usage

```
tissue_mechanical_analysis(theData,start_height,approximate_touch_point=NULL,stress_column="pres
```

Arguments

`theData`	Force-Distance data, typically from a "down"-"up" cycle of the chuck; the data needs to be smooth and contain already at the very least the columns `Distance` and `direction`, in addition to the colums described by the arguments `stress_column` and `stress_slope_column`. Such a variable is typically obtained by using read_texture_analyzer_tab and then smooth_texture_analyzer_data on the result. Care should be taken for the smoothing parameter `sd` in smooth_texture_analyzer_data, both the `pressure` and the `pressureSlope` should appear smooth in a diagram versus `distance`, while overall look of the curve with a flat part, a steeper elastic part, a less steep plateau and then again rising densitifcation in `pressure` vs. `Distance` should be conserved.
`start_height`	The texture Analyzer typically starts a file by `Distance=0`, so we loose the information on how high the chuck is above the substrate before starting compression. This needs to be noted separately and transmitted to this function in the `start_height` argument.
`approximate_touch_point`	Initial guess for the gel touch point (will be refined by a call to find_initial_touch_point)
`stress_column`	Name of the column in `theData` that contains the stress information. By default, this is the "pressure" column as produced by read_texture_analyzer_tab and smoothened by smooth_texture_analyzer_data. An alternative column can be used, for instance if chuck buoyancy is substracted with the creation of a new column (done manually).
`stress_slope_column`	Name of the column in `theData` that contains the slope of the stress. The units are the units of the stress column devided by the units of the Distance column. By default, this is the "pressureSlope" column as produced by read_texture_analyzer_tab and smoothened by smooth_texture_analyzer_data. An alternative column can be used, for instance if chuck buoyancy is substracted with the creation of a new column (done manually).

Details

The estimations of the Young modulus given in this function approximately corresponds to the local tangent modulus (ATSM E111); they are found by local linear regression (lm).

Value

A list with different geometric and mechanical measures:

touchpoint A numeric vector describing the touchpoint coordinates found for the different directions (usually, "down" and "up").

sample_thickness Estimation of the sample height

E A matrix with rows for the directions ("down", "up" in general) and columns for the different strains at which the Young modulus is estimated (i.e. 9 colums for Young moduli at strains 0.1, 0.2, 0.3, ... , 0.9

hysteresis An overall estimate of the hysteresis between compression ("down") and relaxation ("up"). This is the area between the "down" and "up" curve devided by the area under the "down" curve.

Author(s)

Thomas Braschler

References

ATSM standard E111: https://www.astm.org/Standards/E111.htm

See Also

find_initial_touch_point, used internally

Examples

```
# This is not necessarily the most appropriate use since with foam_mechanical_analysis, there is a more specific
# initial elastic Young modulus is high, followed by a plateau with lower Young moduli before a final increase

data(sampleGelSmooth)
results=tissue_mechanical_analysis(
sampleGelSmooth,start_height=start_height_mm)
results
```

---

tissue_mechanical_analysis_list

tissue_mechanical_analysis_list

Description

Analysis of list of compression curves, for example from biological tissues. This function applies tissue_mechanical_analysis to each compression curve and collects the results

Usage

```
tissue_mechanical_analysis_list(file_info,data_list,start_height_column="start_height_mm",approx
```

Arguments

`file_info`	Description of the compression curves. This is a dataframe, which needs to contain at least the columns with names given by the arguments `start_height_column` and `approximate_gel_touch_point_column`, but usually contains more information such as the fabrication conditions. `file_info` needs to contain as many lines as there are elements in `data_list`

`data_list` List of compression curves, typically produced by read_mecmesin_tab_list

`start_height_column`
Name of the starting height column in `file_info`. This column describes the starting height of the chuck at the beginning of compression, in millimeters

`approximate_gel_touch_point_column`
Name of the column in `file_info` that contains a manual estimation of the approximate gel touchpoint, in mm.

`...` Possibility to pass down further arguments to tissue_mechanical_analysis, used internally

Value

A list with as many elements as there are lines `file_info`. Each of these elements is the output of tissue_mechanical_analysis. As the output of tissue_mechanical_analysis is a list itself, the return value is a list of lists.

Author(s)

Thomas Braschler

See Also

tissue_mechanical_analysis, used internally

`write_foam_mechanical_analysis_list_to_xlsx`
write_foam_mechanical_analysis_list_to_xlsx

Description

Writes the output of `foam_mechanical_analysis_list`, which is a nested list (list of list), into an Excel file for further standalone processing.

Usage

```
write_foam_mechanical_analysis_list_to_xlsx(file_info,output_foam_mechanical_analysis_list,file_
```

Arguments

`file_info` Description of the compression curves. This is a dataframe, which needs to contain as many lines as there are elements in `data_list`

`output_foam_mechanical_analysis_list`
Output by foam_mechanical_analysis_list. This is a list of lists

`file_location` Path to the Excel file to be written. If it already exists, will be overwritten without further notice.

Details

This function uses the various spreadsheet creation, manipulation and saving functions provided by the xlsx package for generating and writing of the output Excel file.

Author(s)

Thomas Braschler

Examples

```
data(CMC_concentration_data)
foam_mechanical_analysis_list_result=foam_mechanical_analysis_list(file_info=CMC_concentration_file_info, c


write_foam_mechanical_analysis_list_to_xlsx(file_info=CMC_concentration_file_info,output_foam_mechanical_an
```

---

young_modulus_cryogel_theory

young_modulus_cryogel_theory

Description

Theoretical calculation of the Young modulus of a cryogel

Usage

```
young_modulus_cryogel_theory(polymer_volume_fraction=0.01,E_polymer=10e6,structural_factor=1)
```

Arguments

polymer_volume_fraction	Volume fraction occupied by the polymer phase (rather than the pore space)
E_polymer	Intrinsic Young modulus of the polymer phase
structural_factor	Factor relating the actual mechanical response to the idealized cantilever situation. On the order of 1 for structurally fully intact gels, and 0.01-0.1 for gels where some of the bridges are not formed due to too low viscosity of the monomer

Details

To obtain an order-of-magnitude estimation of the cryogel macroscopic modulus, a following cantilever-based model is used, as described in Beduer 2015

Each pore structure is represented by a single cantilever of the following dimensions:
- Thickness given by `polymer_volume_fraction*pore_size`
- Length of `pore_size/2`
- Width given by `pore_size/2`

Such a cantilever has a spring constant given by:
k=E_polymer*beam_thickness^3*beam_width/4/beam_length^3
=E_polymer*polymer_volume_fraction^3*pore_size^3*pore_size/2/4/(pore_size^3/8)=E_polymer*polymer_v

The Young modulus can be calculated by considering full compression (e.g., compression by pore_size), and then reporting it to the area of a pore (e.g., pore_size^2):
E_cryogel = k*pore_size/pore_size^2=E_polymer*polymer_volume_fraction^3

Maybe a bit surprisingly the global modulus of the cryogel is independent of the pore size.

In reality, the gels tend to be weaker than the idealized calculation because some of the "bridges" are actually interrupted, and maybe also because the sheet-cantilever model simply supposes a too high connectivity. To account for this, it is possible to supply structural_factor, which is used to multiply the cryogel Young modulus (E_cryogel = E_polymer*polymer_volume_fraction^3*structural_factor).

Value

Macroscopic Young modulus of the cryogel

Author(s)

Thomas Braschler

References

Beduer A, Braschler T, Peric O, Fantner GE, Mosser S, Fraering PC , Bencherif SA, Mooney DJ, Renaud P: A compressible scaffold for minimally invasive delivery of large intact neuronal networks, Adv Healthc Mater, 2015, 4(2):301-12, https://doi.org/10.1002/adhm.201400250

Examples

```
young_modulus_cryogel_theory()
```

---

young_modulus_thin_film_correction
young_modulus_thin_film_correction

Description

Correction factor for thin films in the Hertz model

Usage

```
young_modulus_thin_film_correction(R_indentor=20e-9,h_gel=100e-9,indentation_depth=40e-9)
```

Arguments

R_indentor	Radius of the spherical indentor
h_gel	Height of the gel
indentation_depth	Indentation depth

Details

This function calculates the no-slip case as proposed by Long 2011.

Value

The correction for the apparent Young modulus as calculated by the Hertz model. NA for inappropriate values of the argument

Author(s)

Thomas Braschler

References

Rong Long,Matthew S. Hall,Mingming Wu, and Chung-Yuen Hui: Effects of Gel Thickness on Microscopic Indentation Measurements of Gel Modulus, Biophysical Journal Volume 101, 2011, 643-650.

Examples

```
young_modulus_thin_film_correction()
```

Index