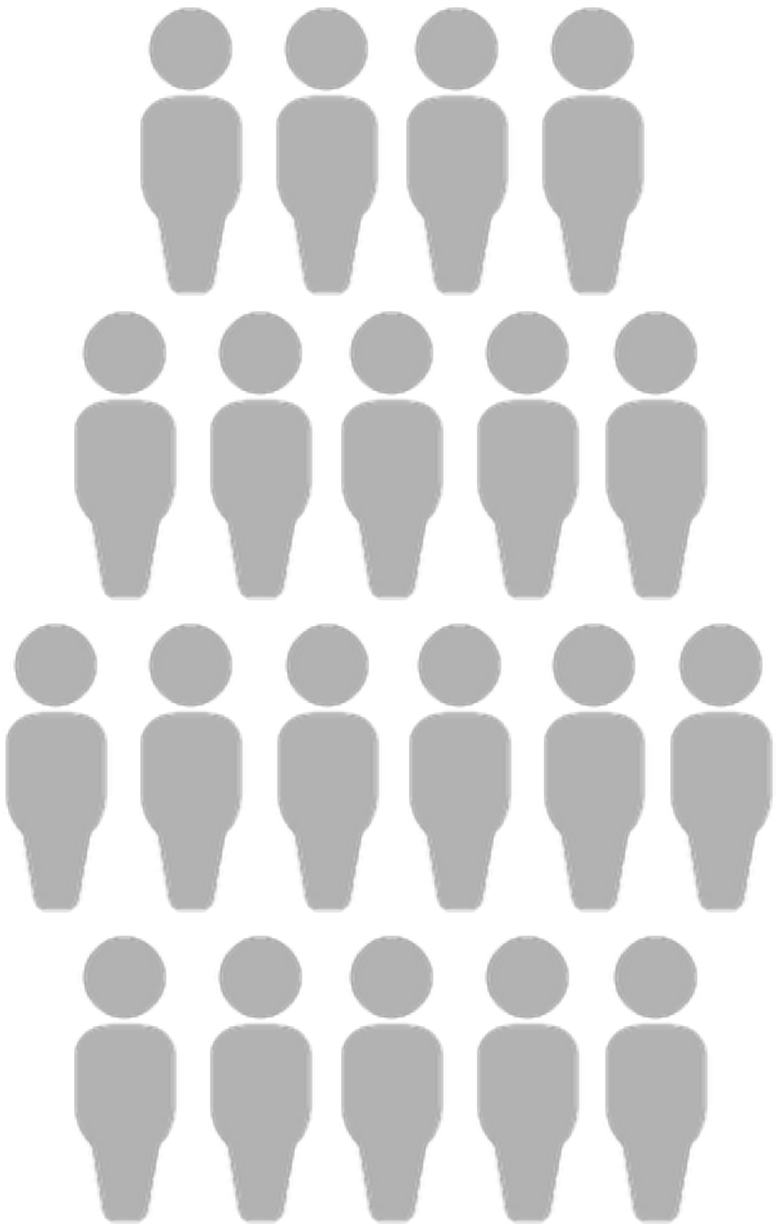
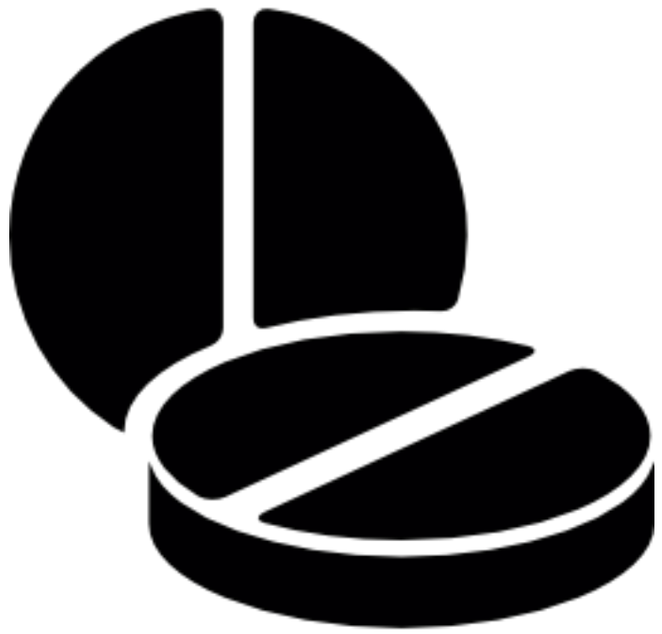
Understanding cancer behaviour with F#

Evelina Gabasova

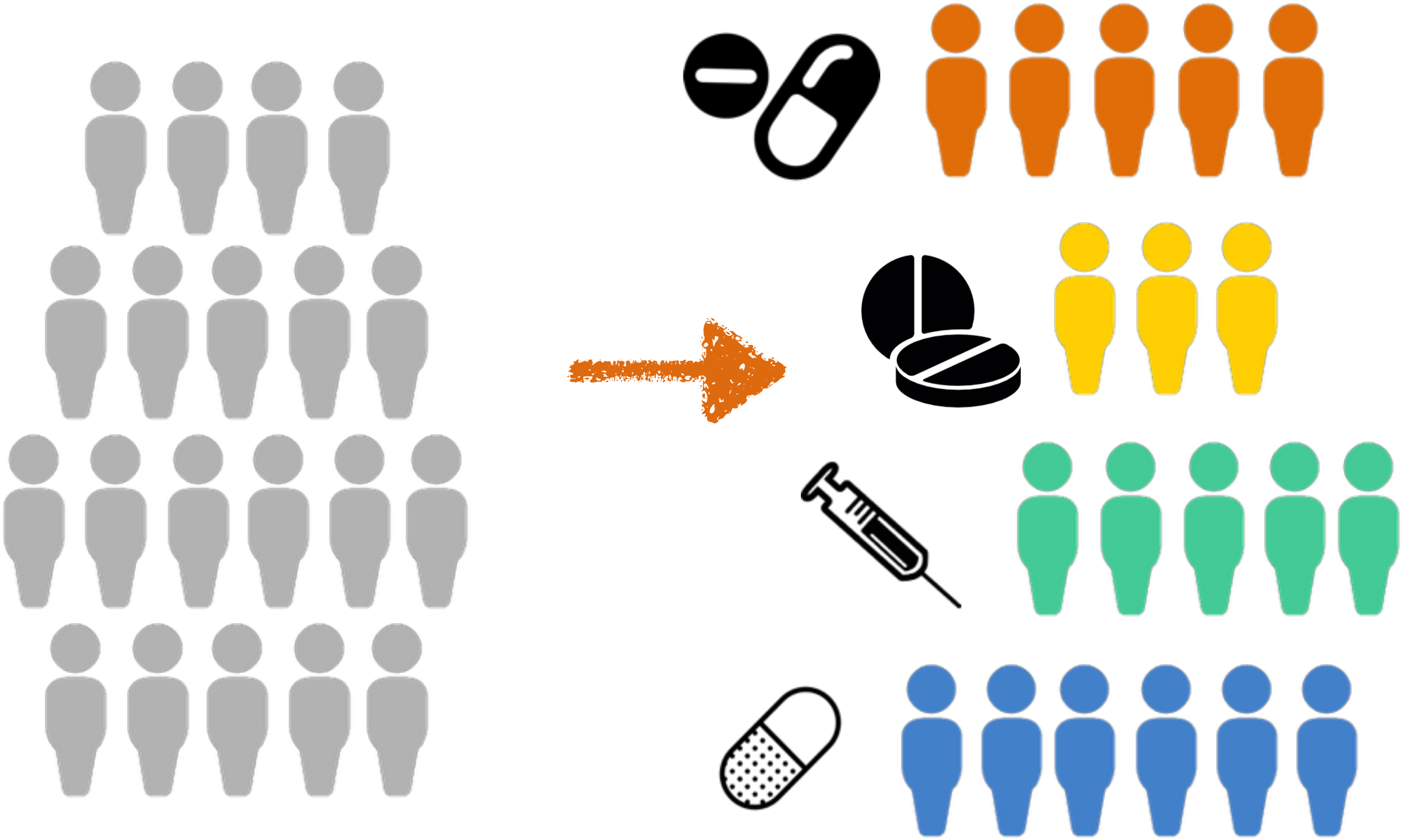
MRC Biostatistics Unit
University of Cambridge

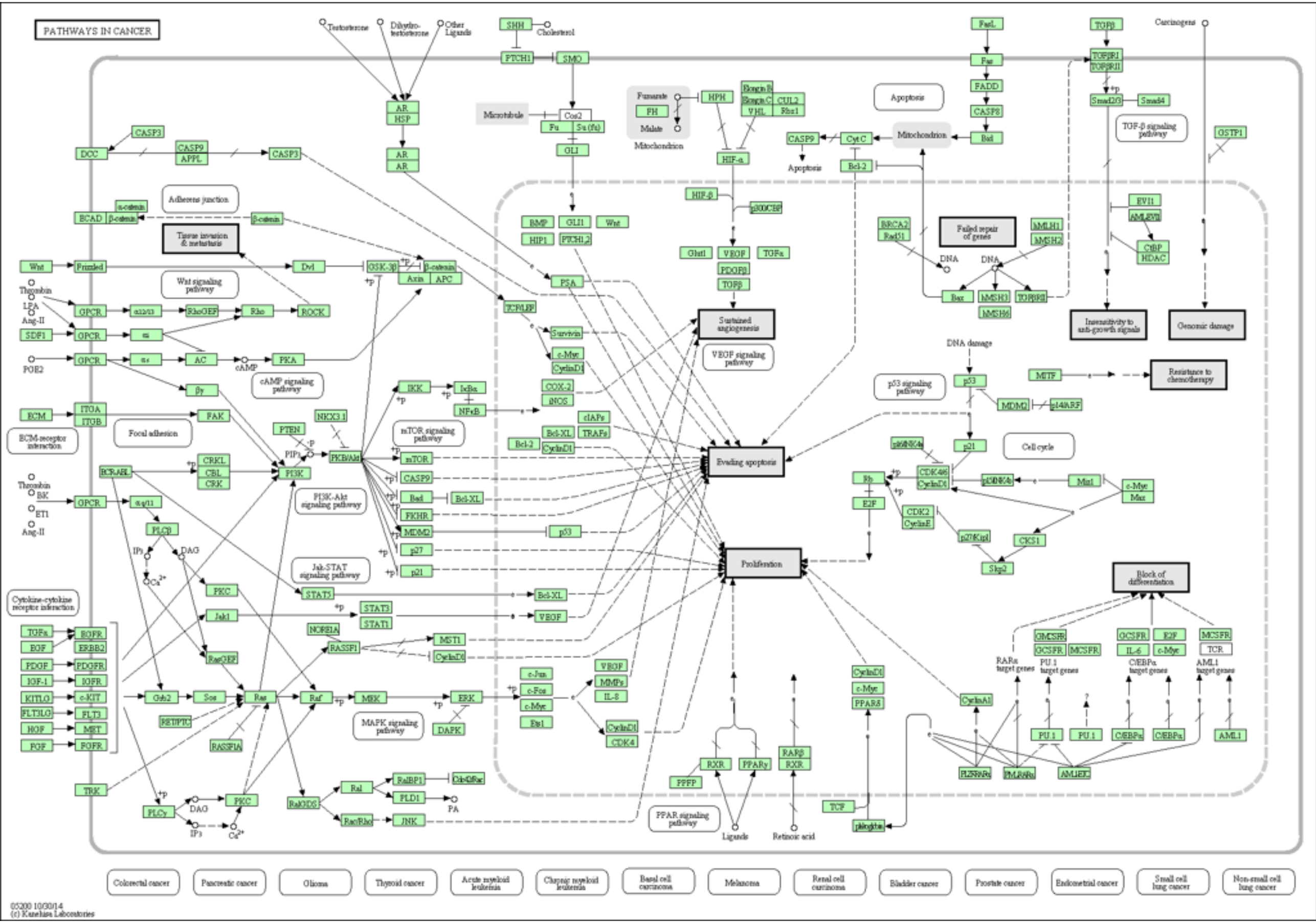
Precision medicine



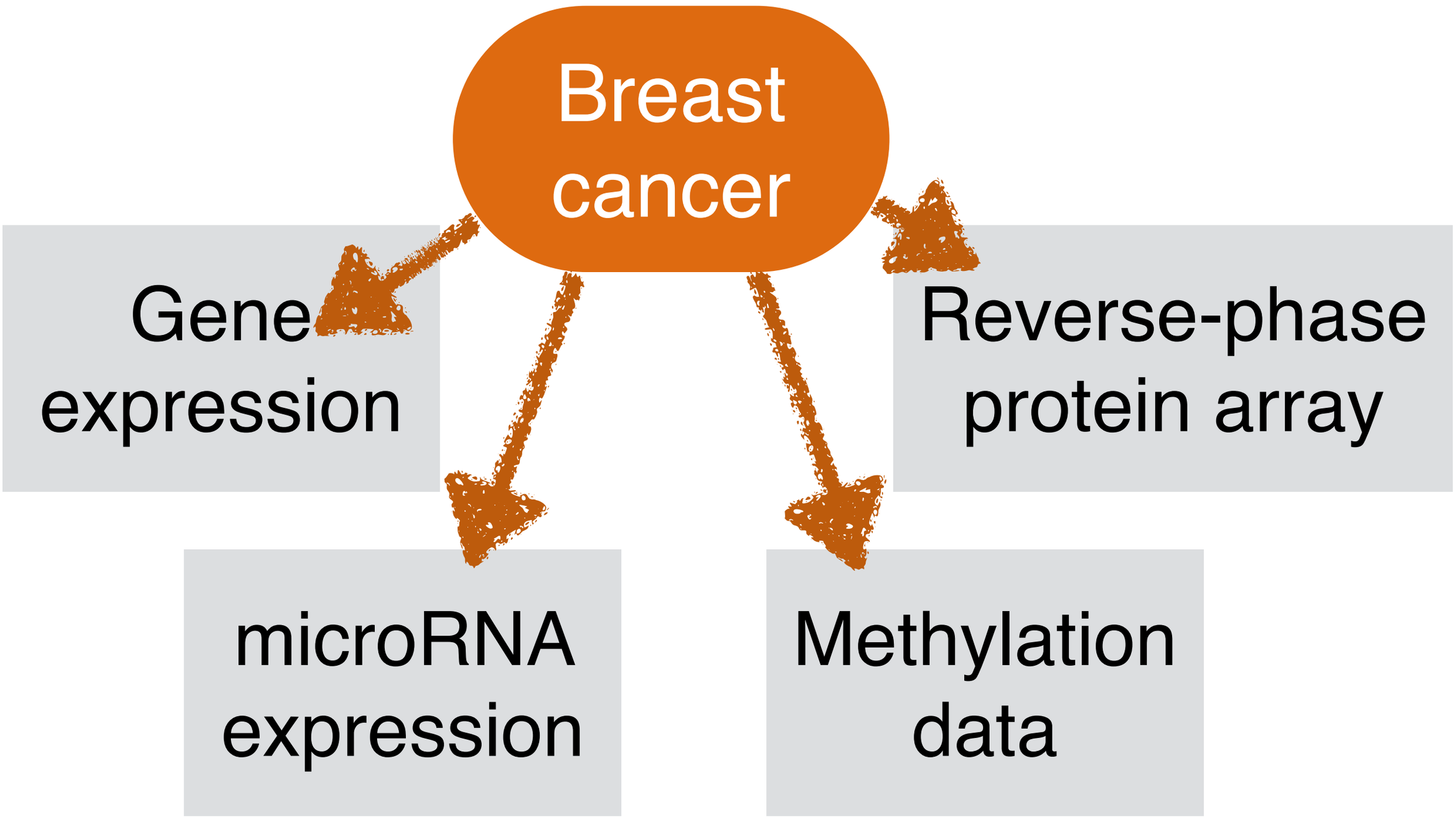


Precision medicine

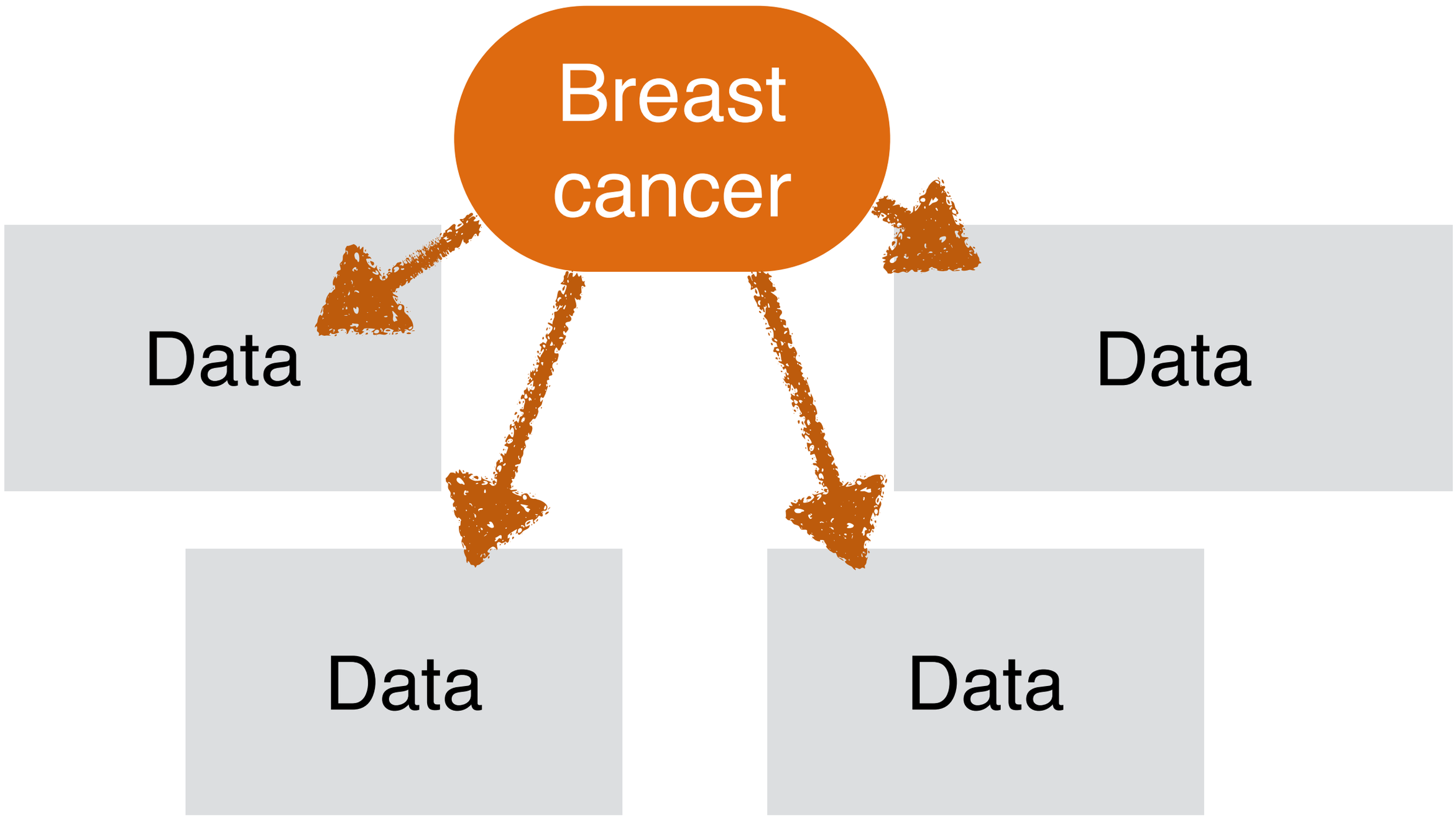




Integrative clustering



Integrative clustering



Scientific programming

Scripting languages

fast prototyping, easy to use

R, Python, Matlab

Why F#?

```
    Lp = lapply(logL, exp)
for(m in 1:M){
    for(i in 1:N) L[[m]][i,] = rmultinom(1,1,Lp[[m]][i,]) #Generate L from Lp
    if(w>1) L[[m]] = AlignClusters(C,L[[m]], type = 'mat') #Helps to align indices
    n[m,] = colSums(L[[m]])
    for(k in 1:K){ ###Update cluster parameters based on normal-gamma distribution
        if(d[m]==1&n[m,k]>1){
            S[[m]][,k] = sd(X[[m]][,L[[m]][,k]==1])^2
            PostMean = sum(X[[m]][,L[[m]][,k]==1])/(n[m,k]+1)
            B[[m]][,k] = b0[[m]]+0.5*(n[m,k]*S[[m]][,k]+n[m,k]*(mean(X[[m]][,L[[m]][,k]==1])-mu0
        if(d[m]>1&n[m,k]>1){
            PostMean = (mu0[[m]]+rowSums(X[[m]][,L[[m]][,k]==1]))/(n[m,k]+1)
            S[[m]][,k] = apply(X[[m]][,L[[m]][,k]==1],MARGIN=1,FUN='sd')^2
        B[[m]][,k] = b0[[m]]+0.5*(n[m,k]*S[[m]][,k]+n[m,k]*(rowMeans(X[[m]][,L[[m]][,k]==1])-mu0
        if(n[m,k]==1){
            PostMean = (mu0[[m]]+X[[m]][,L[[m]][,k]==1])/2
        B[[m]][,k] = b0[[m]]+0.5*(X[[m]][,L[[m]][,k]==1]-mu0[[m]])^2/2}
        if(n[m,k]==0){
            PostMean = mu0[[m]]
            B[[m]][,k] = b0[[m]]}
        Lambda = 1+n[m,k]
        A[[m]][,k] = a0[[m]]+n[m,k]/2
        Tau[[m]][,k] = rgamma(d[m],shape=A[[m]][,k],rate=B[[m]][,k])
        mu[[m]][,k] = rnorm(d[m],PostMean,sqrt(1/(Tau[[m]][,k]*Lambda)))
        Sigma[[m]][,k] = sqrt(1/Tau[[m]][,k])}
}
```

Why F#?

```
    Lp = lapply(logL, exp)
for(m in 1:M){
    for(i in 1:N) L[[m]][i,] = rmultinom(1,1,Lp[[m]][i,]) #Generate L from Lp
    if(w>1) L[[m]] = AlignClusters(C,L[[m]], type = 'mat') #Helps to align indices
    n[m,] = colSums(L[[m]])
    for(k in 1:K){ ###Update cluster parameters based on normal-gamma distribution
        if(d[m]==1&n[m,k]>1){
            S[[m]][,k] = sd(X[[m]][,L[[m]][,k]==1])^2
            PostMean = sum(X[[m]][,L[[m]][,k]==1])/(n[m,k]+1)
            B[[m]][,k] = b0[[m]]+0.5*(n[m,k]*S[[m]][,k]+n[m,k]*(mean(X[[m]][,L[[m]][,k]==1])-mu0
        if(d[m]>1&n[m,k]>1){
            PostMean = (mu0[[m]]+rowSums(X[[m]][,L[[m]][,k]==1]))/(n[m,k]+1)
            S[[m]][,k] = apply(X[[m]][,L[[m]][,k]==1],MARGIN=1,FUN='sd')^2
        B[[m]][,k] = b0[[m]]+0.5*(n[m,k]*S[[m]][,k]+n[m,k]*(rowMeans(X[[m]][,L[[m]][,k]==1])-mu0
        if(n[m,k]==1){
            PostMean = (mu0[[m]]+X[[m]][,L[[m]][,k]==1])/2
        B[[m]][,k] = b0[[m]]+0.5*(X[[m]][,L[[m]][,k]==1]-mu0[[m]])^2/2}
        if(n[m,k]==0){
            PostMean = mu0[[m]]
            B[[m]][,k] = b0[[m]]}
        Lambda = 1+n[m,k]
        A[[m]][,k] = a0[[m]]+n[m,k]/2
        Tau[[m]][,k] = rgamma(d[m],shape=A[[m]][,k],rate=B[[m]][,k])
        mu[[m]][,k] = rnorm(d[m],PostMean,sqrt(1/(Tau[[m]][,k]*Lambda)))
        Sigma[[m]][,k] = sqrt(1/Tau[[m]][,k])}
}
```

Vector? Matrix?
List? Array?
Data frame?

Why F#?

```
25          let newas =
26              data.Contexts
27              |> Array.map (fun context ->
28                  let pis = state.ContextWeights.[context]
29                  let rawPriorPis = Array.create pis.Length 1.0
30                  let value = sampleDirichletConcentration (hyperprior.AlphasPrior.[context])
31                  (
32              |> to
33          { state w                                                    as = newas}}
34
35      | FixedValue(
36          // do not
37          state
38
39  // Context-specif
40  // ***************
41
42  // Sample from conditional Dirichlet distribution with random walk Metropolis-Hastings
43  let sampleDirichlet_MetropolisHastings (currentValues: float[])
44      priorConcentration loglikFunction =
45      // 1. Add random walk proposal to current values
46      let randomWalkProposal = Normal(0.0, 1.0, rnd)
47      let proposal_unnorm =
48          currentValues
49          |> Array.map (fun x -> x + 0.1 * randomWalkProposal.Sample())
50
```

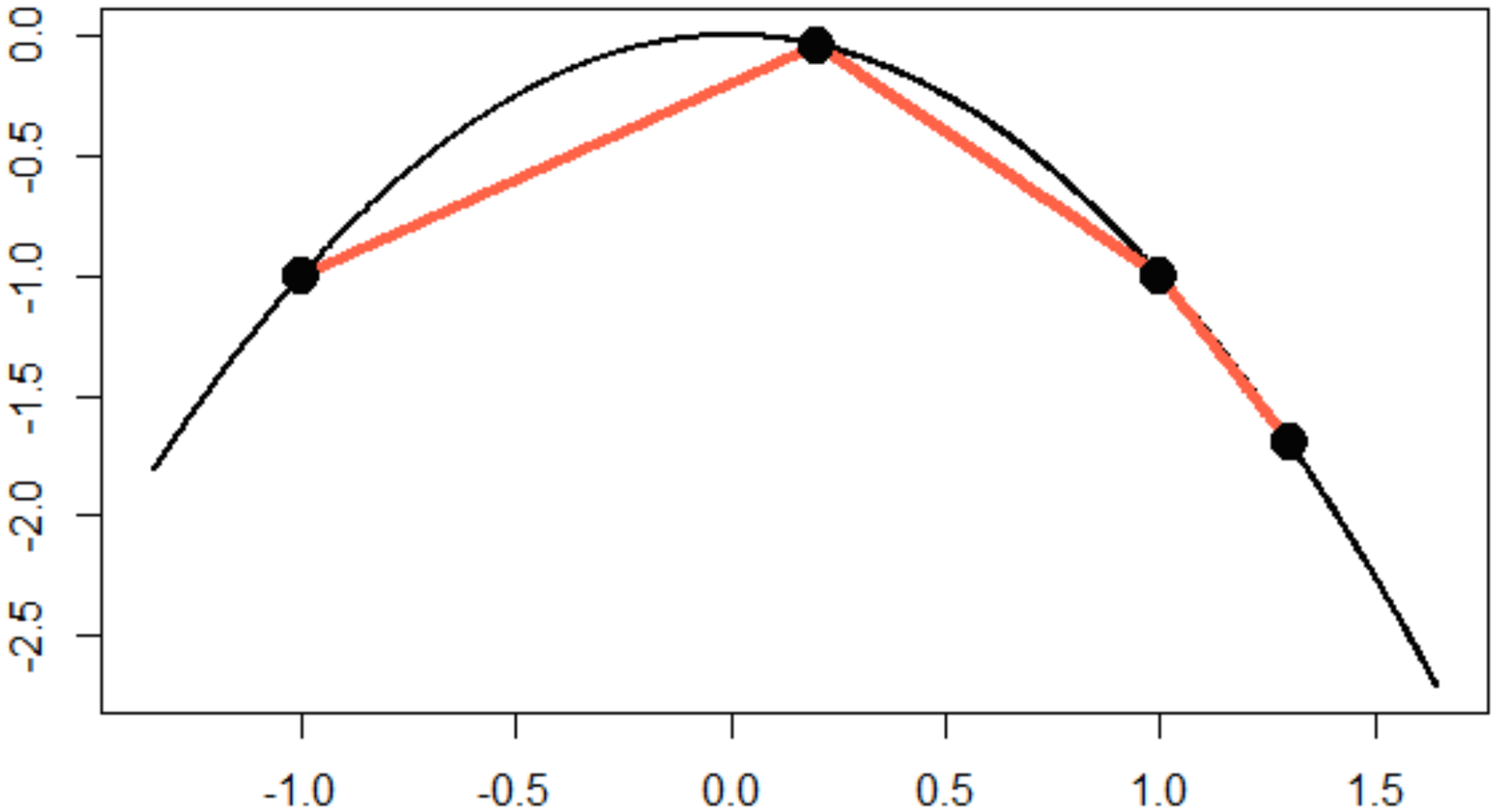
```
val sampleDirichletConcentration :
    gammaDist  : Gamma ->
    rawPriorPis: seq<float> ->
    pis        : seq<float> ->
    sampleType : SampleConcentrationParams
                 -> float

Summary
Adaptive rejection sampling (derivative-free)
```

Case study

Rewrite an algorithm from Matlab into F#

adaptive rejection sampling

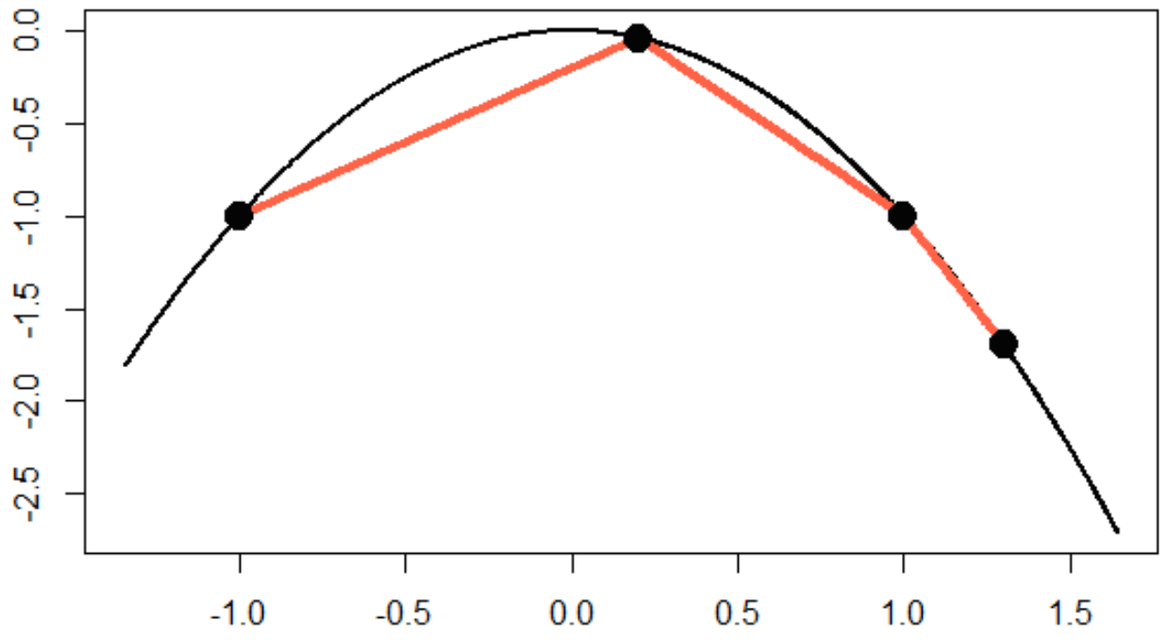


Case study

```
% value of the lower bound
if x<min([lowerHull.left])
  lhVal = -inf;
elseif x>max([lowerHull.right]);
  lhVal = -inf;
else
  for li=1:length(lowerHull)
    left = lowerHull(li).left;
    right = lowerHull(li).right;

    if x>=left && x<=right
      lhVal = lowerHull(li).m*x + lowerHull(li).b;
      break;
    end
  end
end
```

MatLab



Pattern matching

```
% value of the lower bound
if x<min([lowerHull.left])
  lhVal = -inf;
elseif x>max([lowerHull.right]);
  lhVal = -inf;
else
  for li=1:length(lowerHull)
    left = lowerHull(li).left;
    right = lowerHull(li).right;

    if x>=left && x<=right
      lhVal = lowerHull(li).m*x + lowerHull(li).b;
      break;
    end
  end
end
```

MatLab

```
// value of the lower bound
let lhVal =
    match (lowerHull, x) with
    | OutsideOnLeft -> -infinity
    | OutsideOnRight -> -infinity
    | InsideInterval lh -> lh.M * x + lh.B
```

F#

Pattern matching

```
% value of the lower bound
if x<min([lowerHull.left])
 lhVal = -inf;
elseif x>max([lowerHull.right]);
 lhVal = -inf;
else
  for li=1:length(lowerHull)
    left = lowerHull(li).left;
    right = lowerHull(li).right;

    if x>=left && x<=right
      lhVal = lowerHull(li).m*x + lowerHull(li).b;
      break;
    end
  end
end
```

MatLab

```
// value of the lower bound
let lhVal =
    match (lowerHull, x) with
     | OutsideOnLeft -> -infinity
     | OutsideOnRight -> -infinity
     | InsideInterval lh -> lh.M * x + lh.B
```

F#

Legibility

$$X + \log\left(\sum_{i=1}^{N} \exp\left\{x_i - X\right\}\right)$$

$$\text{where } X = \max\left\{x_i; i = 1, \ldots, N\right\}$$

```
let logSumExp xs =
    let maxValue = Array.max xs
    xs
    |> Array.map (fun x -> exp(x - maxValue))
    |> Array.sum
    |> log
    |> (+) maxValue
```

Legibility

$$X + \log\left(\sum_{i=1}^{N} \exp\{x_i - X\}\right)$$

$$\text{where } X = \max\{x_i; i = 1, \ldots, N\}$$

```
let logSumExp xs =
    let maxValue = Array.max xs
    xs
    |> Array.map (fun x -> exp(x - maxValue))
    |> Array.sum
    |> log
    |> (+) maxValue
```

Legibility

$$X + \log\left(\sum_{i=1}^{N} \exp\{x_i - X\}\right)$$

$$\text{where } X = \max\{x_i; i = 1, \ldots, N\}$$

```
let logSumExp xs =
    let maxValue = Array.max xs
    xs
    |> Array.map (fun x -> exp(x - maxValue))
    |> Array.sum
    |> log
    |> (+) maxValue
```

Typical workflow

Problem

Breast cancer

Gene expression

microRNA expression

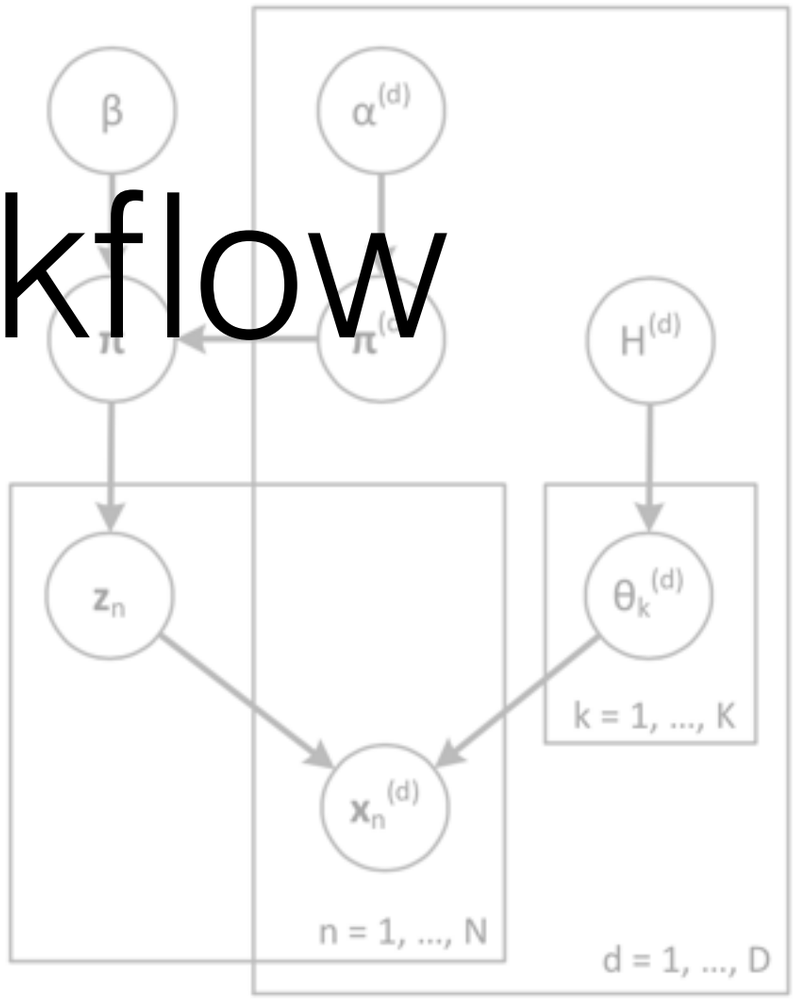
Methylation data

Revers
protei

Typical workflow

Problem

Model



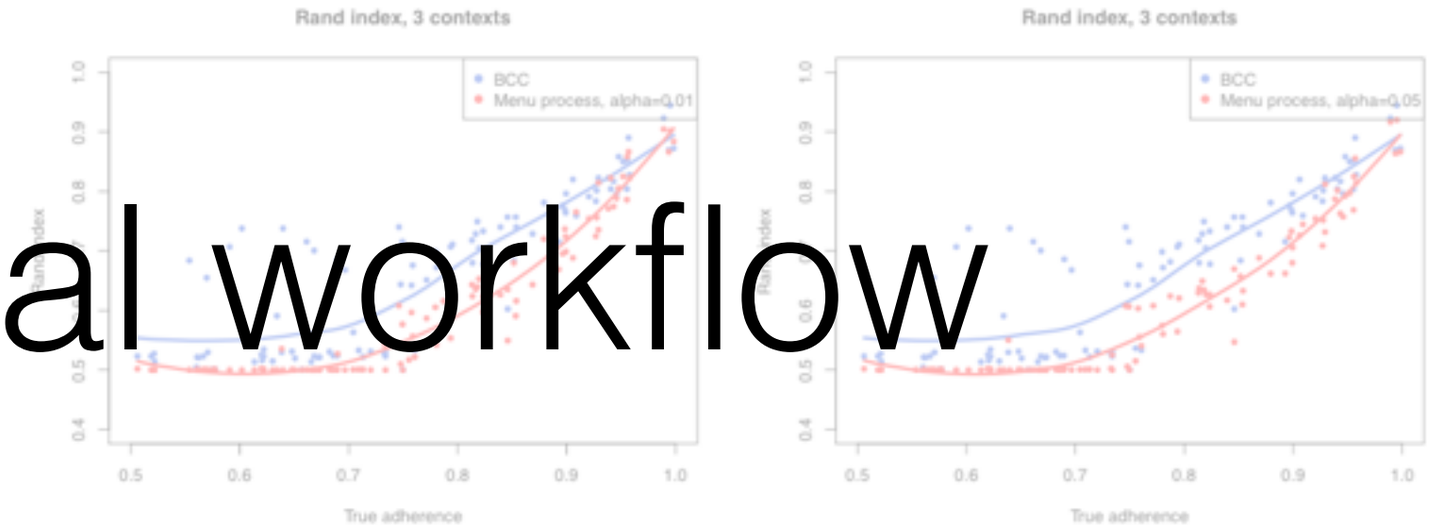
$$
\begin{aligned}
p\left(\pi^{(d)}\middle|\alpha_d,\beta,\pi\right) &\propto p\left(\pi^{(d)}\middle|\alpha_d\right)\; p\left(\pi\middle|\pi^{(1)},\ldots,\pi^{(D)},\beta\right) \\
&= \left\{\frac{\Gamma\left(K^{(d)}\alpha_d\right)}{\left(\Gamma(\alpha_d)\right)^{K^{(d)}}}\prod_{k=1}^{K^{(d)}}\left(\pi_k^{(d)}\right)^{\alpha_d-1}\right\}\times \\
&\quad\times\left\{\frac{\Gamma\left(\sum_{i_1=1}^{K^{(1)}}\cdots\sum_{i_D=1}^{K^{(D)}}\left[\beta\times\pi_{i_1}^{(1)}\times\cdots\times\pi_{i_D}^{(D)}\right]\right)}{\prod_{i_1=1}^{K^{(1)}}\cdots\prod_{i_D=1}^{K^{(D)}}\Gamma\left(\beta\times\pi_{i_1}^{(1)}\times\cdots\times\pi_{i_D}^{(D)}\right)}\prod_{i_1=1}^{K^{(1)}}\cdots\prod_{i_D=1}^{K^{(D)}}\left(\pi_{i_1,\ldots,i_D}\right)^{\beta\pi_{i_1}^{(1)}\ldots\pi_{i_D}^{(D)}-1}\right\} \\
&\quad\times\frac{\Gamma(\beta)}{\prod_{i_1=1}^{K^{(1)}}\cdots\prod_{i_D=1}^{K^{(D)}}\Gamma\left(\beta\times\pi_{i_1}^{(1)}\times\cdots\times\pi_{i_D}^{(D)}\right)}\times \\
&\quad\times\left\{\prod_{k=1}^{K^{(d)}}\left(\pi_k^{(d)}\right)^{\alpha_d-1}\right\}\left\{\prod_{i_1=1}^{K^{(1)}}\cdots\prod_{i_D=1}^{K^{(D)}}\left(\pi_{i_1,\ldots,i_D}\right)^{\beta\pi_{i_1}^{(1)}\ldots\pi_{i_D}^{(D)}-1}\right\} \\
&\propto \frac{\Gamma\left(K^{(d)}\alpha_d\right)}{\left(\Gamma(\alpha_d)\right)^{K^{(d)}}}\left\{\prod_{k=1}^{K^{(d)}}\left(\pi_k^{(d)}\right)^{\alpha_d-1}\right\}\left\{\prod_{i_1=1}^{K^{(1)}}\cdots\prod_{i_D=1}^{K^{(D)}}\frac{1}{\Gamma\left(\beta\times\pi_{i_1}^{(1)}\times\cdots\times\pi_{i_D}^{(D)}\right)}\left(\pi_{i_1,\ldots,i_D}\right)^{\beta\pi_{i_1}^{(1)}\ldots\pi_{i_D}^{(D)}-1}\right\}
\end{aligned}
$$

Typical workflow

Problem

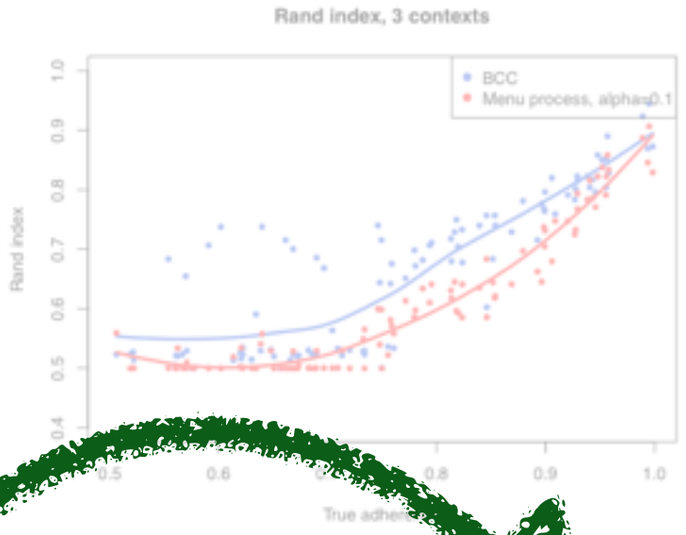
Model

Evaluation

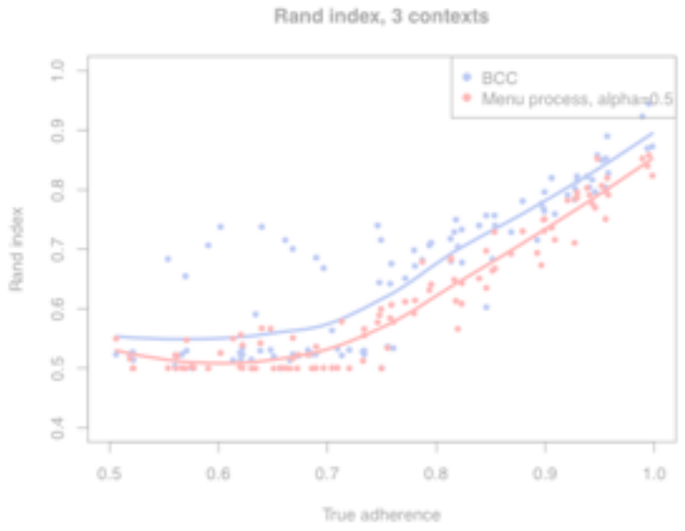


(a) $\alpha = 0.01, \gamma = 0.5$

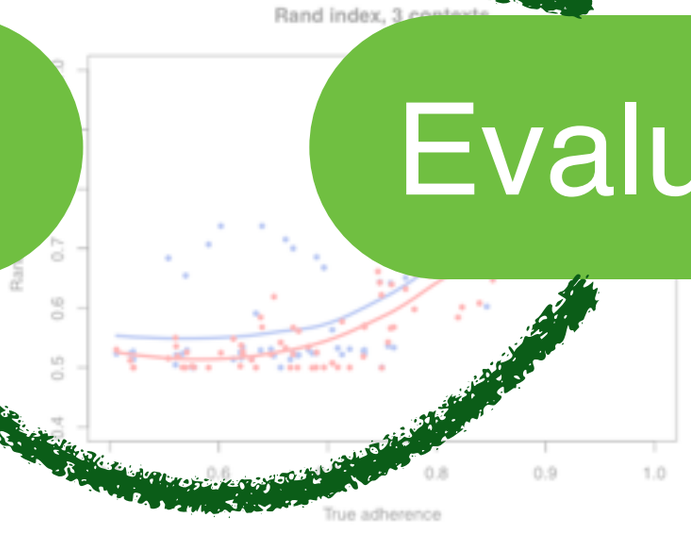
(b) $\alpha = 0.05, \gamma = 0.5$



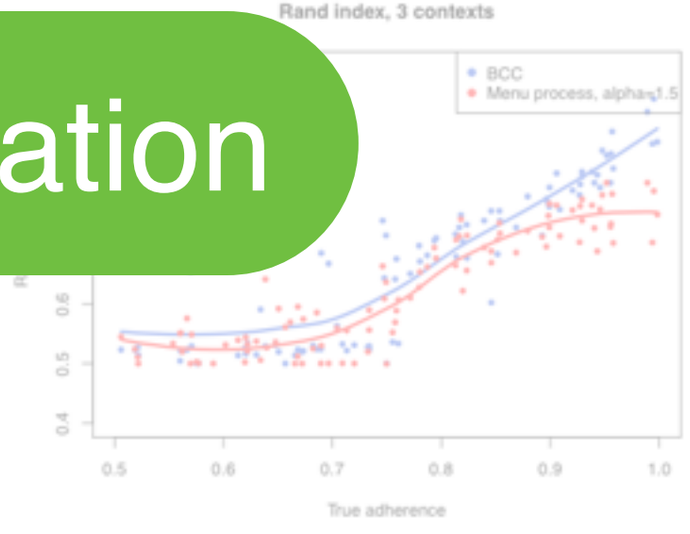
(c) $\alpha = 0.1, \gamma =$



(d) $\alpha = 0.5, \gamma = 1$

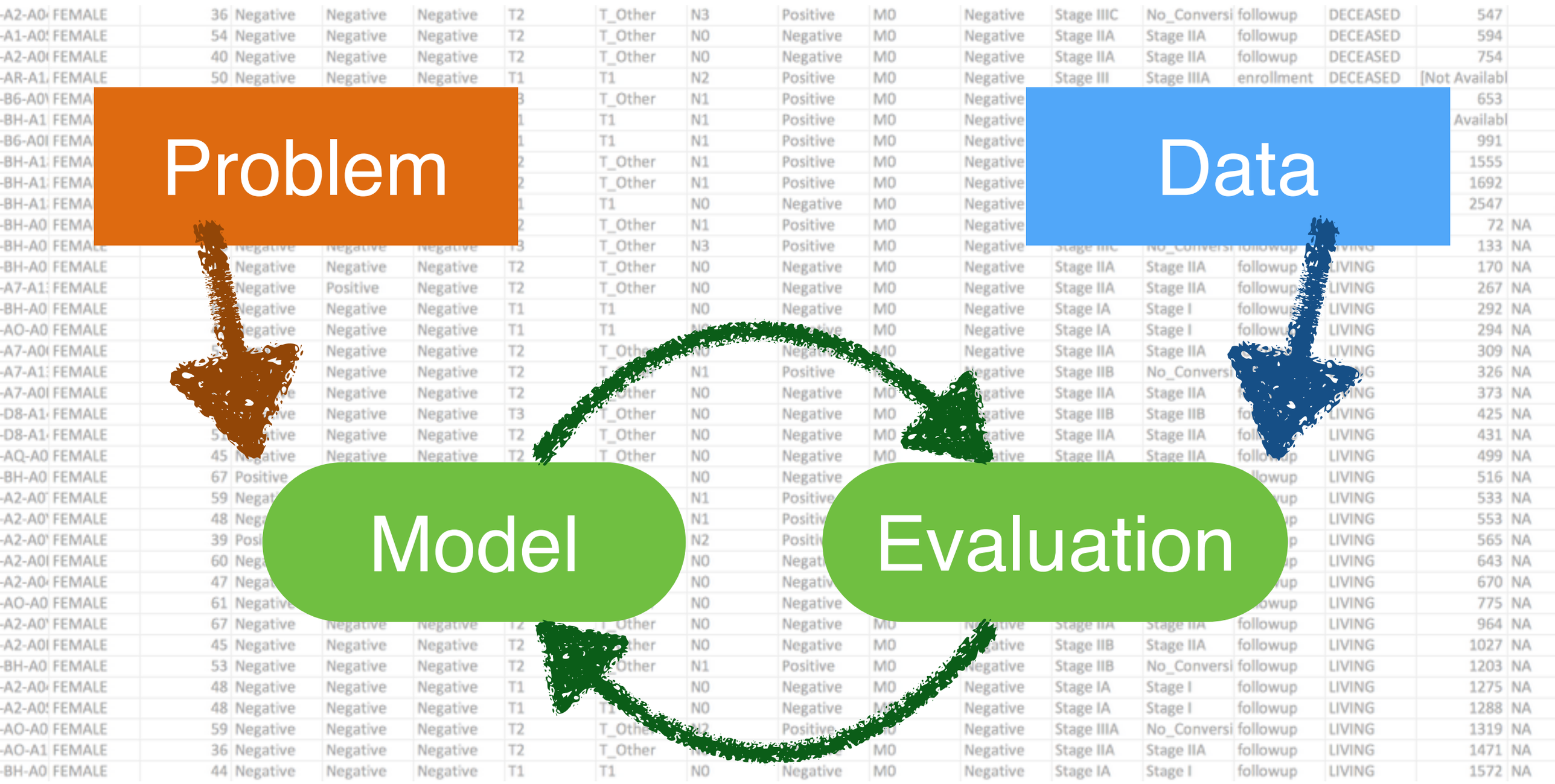


(e) $\alpha = 1, \gamma = 1$

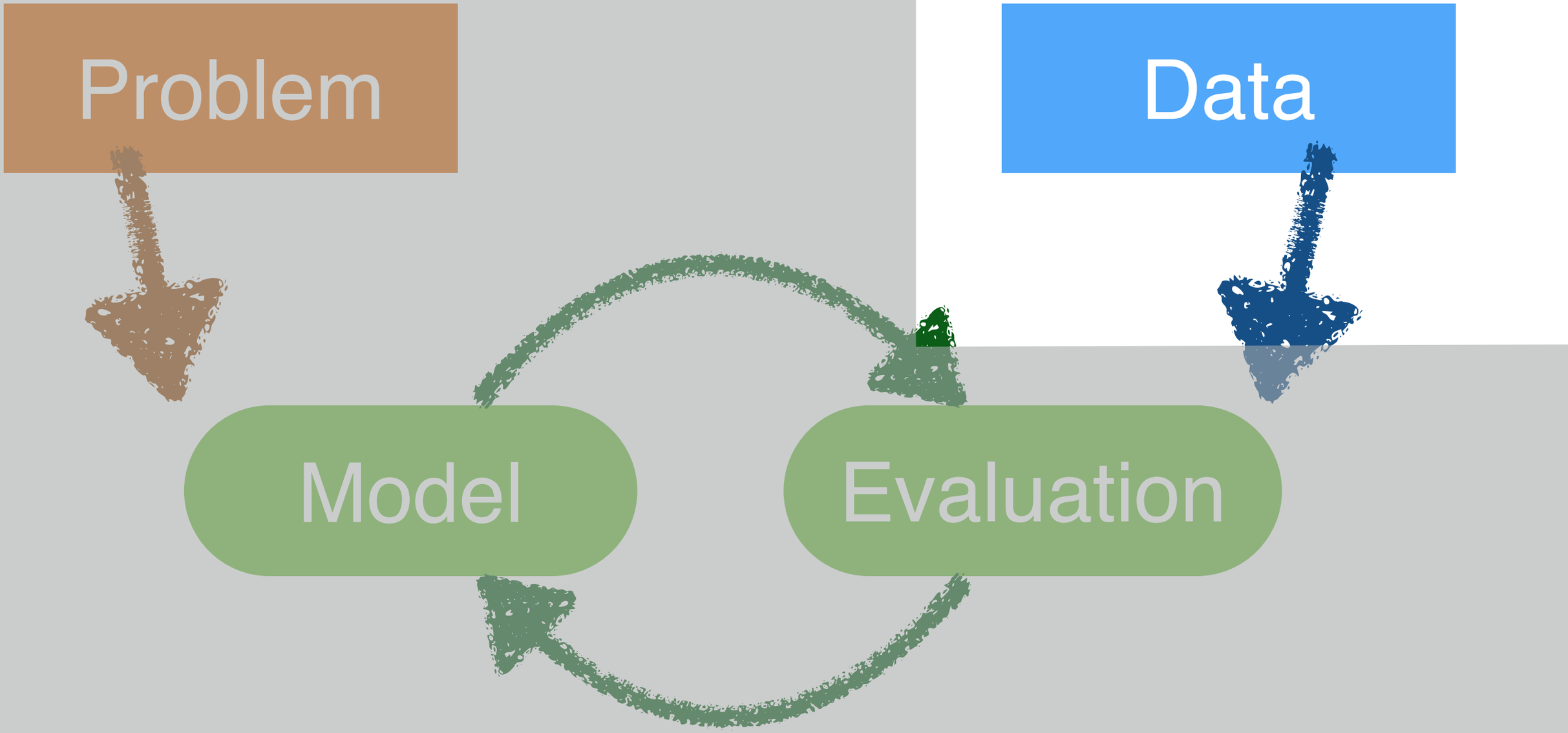


(f) $\alpha = 1.5, \gamma = 1$

Typical workflow



Typical workflow



Typical workflow

Usually takes much more time than it should

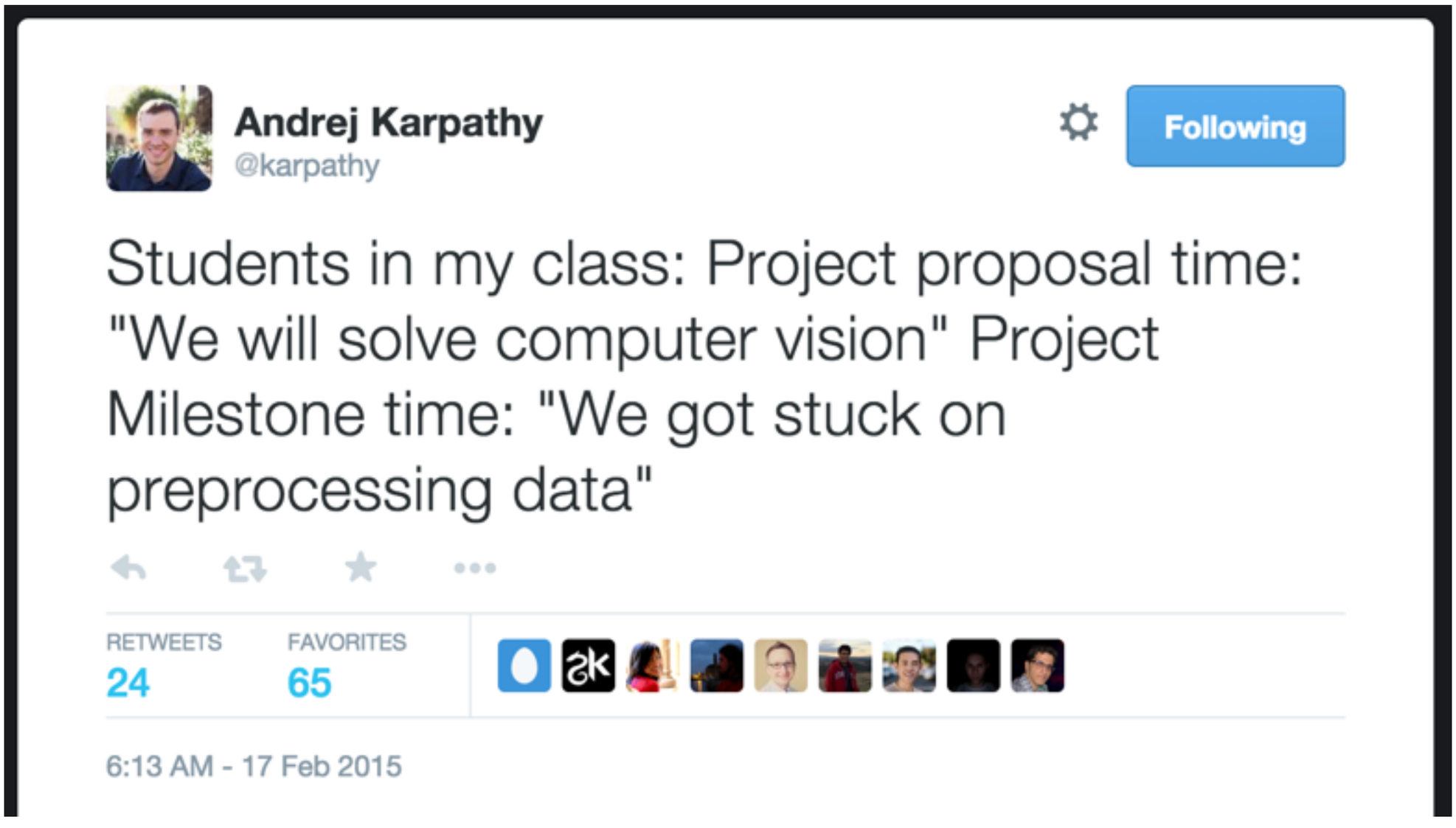
Problem

Data

Model

Evaluation

Dealing with real-world data



Type providers

F# Data - typed access to external data sources

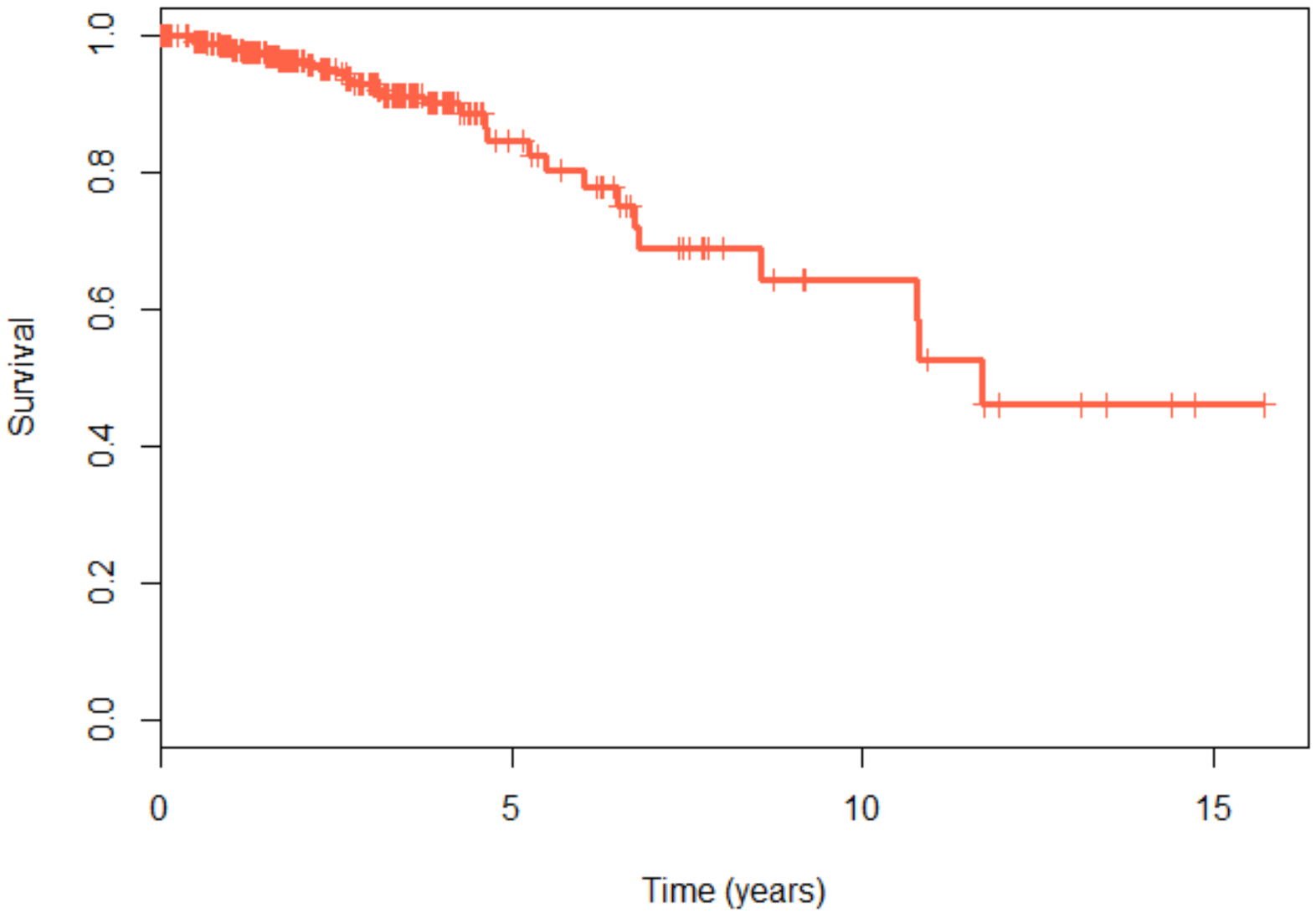
Csv files

JSON, XML, HTML

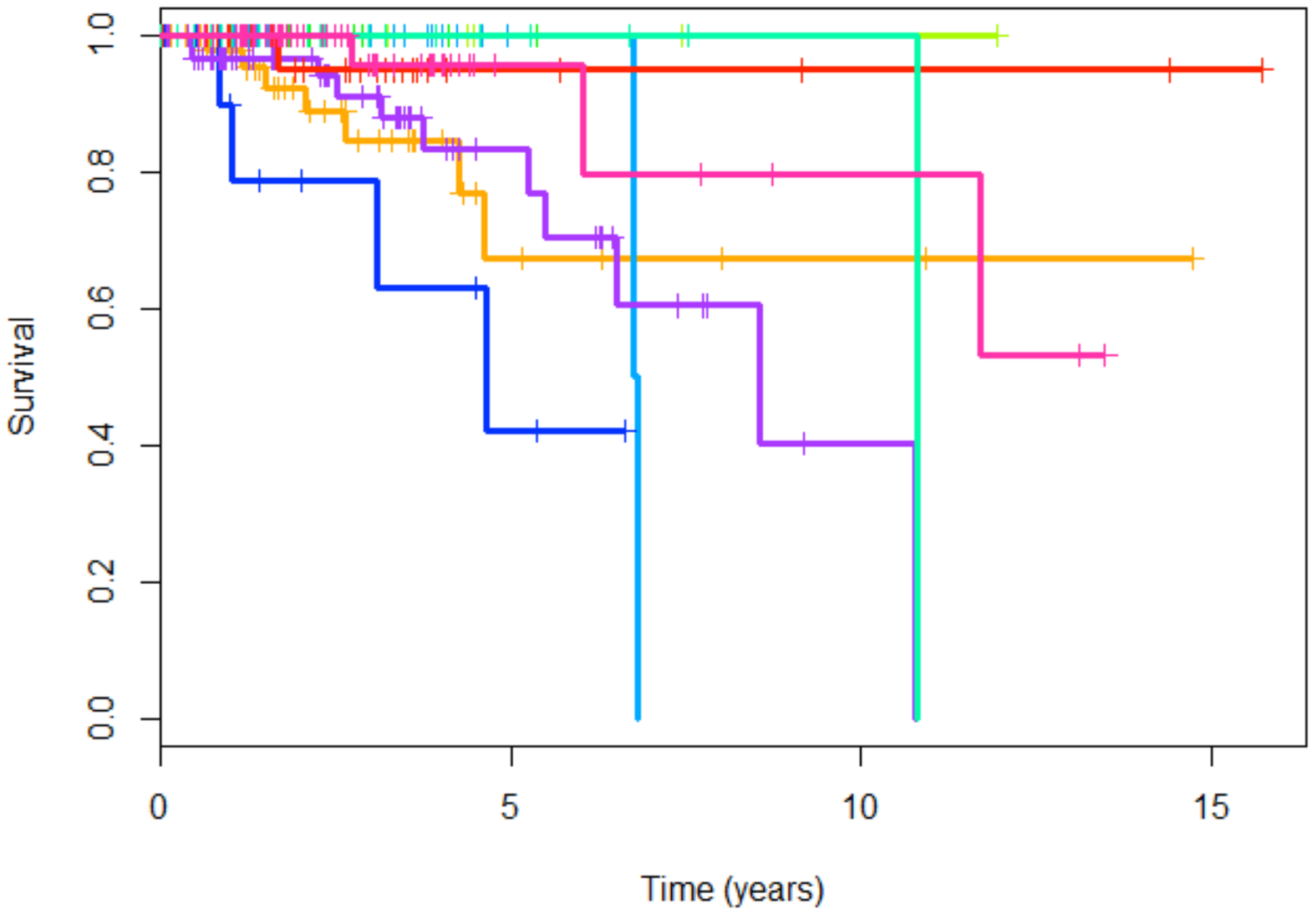
RProvider

Bioconductor
OPEN SOURCE SOFTWARE FOR BIOINFORMATICS

Results



Results



Thank you

@evelgab

evelinag.com

evelina@evelinag.com

F# eXchange 17 April in London